Detecting Multi-Block Double Spent Transaction Based On B-Tree Indexing

A. Murugan, J. Vijayalakshmi

Abstract: The emergence of cryptocurrency as a new payment method gives rise to various security threats related to transaction malleability like privacy leakage, loss of amount, doing illicit activities, and double-spending of the same money more than once. Double-spending of money is quite easier to implement in cryptocurrency management rather than fiat currency management because data replication can be easily done here. The rise of double-spending may degrade the performance of the Bitcoin network. In Blockchain, there may be lakhs of records and thousands of blocks available, of this detecting double-spending data in multiple blocks takes more time. To address the time management, of detecting double-spent data in multiple blocks of blockchain this paper had proposed Multi-Block Double spent Transaction Detection (MBDTD) architecture using B-tree indexing and Cognizant Merkle. The combination of Cognizant Merkle and B-tree indexing supports the rapid verification of transaction data in multiple blocks. B-tree indexing supports speedy retrieval of Merkle value among multiple blocks and Cognizant Merkle supports quick searching of transactions in each block.

Index Terms: Bitcoin, Blockchain, B-tree index, Cognizant Merkle, Double-spend attack, Peer-to-Peer network, Transactions,

1. INTRODUCTION

Blockchain is an organized data structure that links the blocks [3] sequentially from genesis block to current block which holds the records of financial transactions. This uses cryptography and digital signatures method to prove authenticity, identity and access rights for reading or writing operations. The data is written by some nodes and it is read by multiple participants around the network [2]. The Blockchain is a renovate technology that enables users to execute the financial transactions in a guaranteed manner and it can be audited by everyone without the need for a trusted third party. Each transaction history is indexed in a decentralized ledger and redistributed to all users in the network. The complexity of transactions like Proof-Of-Work (POW) makes these transactions impossible to falsify [1]. Blockchain works on the principle of distributed ledger system which means the data record is not kept by any one central authority rather it is available at each and every node on the Peer-to-Peer network (P2P). Every participant node keeps the updated copy of the data record in the blockchain. The record doesn't specify individual transaction rather it shows the transaction that took place between any two nodes on the network. This ensures that the entire network is fraud-proof because everyone has an authentic copy of records and no one can falsify the transaction or double-spend the same bitcoins [3]. The architecture of blockchain is shown in Figure 1 which consists of three layers namely Application layer, Decentralized ledger layer, and Peer-to-Peer network layer. The Application layer is built based on the underlying decentralized ledger layer maintained on a peer-to-peer network. In the case of the Bitcoin network, the application layer supports the exchange of bitcoins for e-commerce transactions through a Bitcoin Wallet. A Bitcoin Wallet represents the user's unspent bitcoins. The decentralized ledger layer consists of multiple components which ensure that the single, global ledger remains consistent and tamper-proof. The decentralized ledger layer is called the blockchain where transactions are arranged into blocks and each block cryptographically linked to its parent block to form a chain.

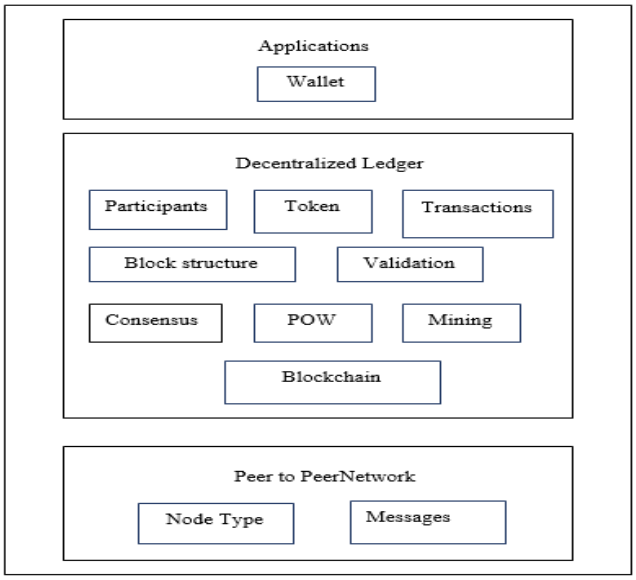


Figure 1: *Blockchain architecture*

A transaction represents a token exchange between participants. In the case of the Bitcoin network, a transaction represents the movement of bitcoins between different addresses. The validation section performs the transaction validation which is done by all nodes on the network. Mining is the process of grouping transactions into block that is added to the end of current blockchain [4]. Proof-Of-Work (POW) algorithm is used by blockchain to ensure that there is a consensus (mutual agreement) among all nodes on the network regarding transaction validation. The bottom layer represents the P2P network with different roles played by multiple nodes in the system. These nodes are useful for exchanging various messages for updating and maintaining the decentralized ledger [4].

- *A. Murugan, Associate Professor & Head, PG & Research Department of Computer Science, Dr. Ambedkar Government Arts College (Autonomous), Affiliated to University of Madras, Chennai, India, PH- 9444340011. E-mail: amurugan1972@gmail.com*
- *J. Vijayalakshmi, Research Scholar, PG & Research Department of Computer Science, Dr. Ambedkar Government Arts College (Autonomous), Affiliated to University of Madras, Chennai, India, PH-8124682620. E-mail: jeyamaha2002@gmail.com*

2. TRANSACTION VERIFICATION THROUGH BITCOIN MERKLE

Ralph C. Merkle [7] introduces the Merkle tree concept in the year 1979. The Bitcoin network maintains the history of all transactions in blockchain by using the Merkle tree. In P2P network, data consistency and verification are very important because some data may be cloned in multiple locations. Data verification, synchronization, and consistency can be easily achieved with the help of the Merkle tree. Merkle tree serves two purposes namely i) it ensures data authenticity by verifying the data before downloading a file from Peer-to-Peer nodes and ii) to verify information from doubtful resource. This Merkle tree is used by lightweight bitcoin nodes. Lightweight nodes verify transactions using a method called Simplified Payment Verification (SPV). SPV allows a node to verify if a transaction has been included or not without downloading the entire blockchain [6]. SPV nodes are special bitcoin nodes that are designed to run on space and power-constrained devices like smart phones, tablets or embedded systems. SPV clients allow them to operate a Bitcoin network without storing the full blockchain [10]. SPV nodes will only download the transactions and block headers in each block rather than storing full blockchain network. This type is 1000 times smaller than full blockchain [5]. Present bitcoin system uses Binary Hash Tree (BHT) techniques for implementing the Merkle tree [11]. Merkle provides biometric security like a digital fingerprint of the entire set of transactions in a particular block. In Bitcoin, Merkle tree is determined by repeatedly hashing the pair of nodes until there is one hash called Merkle root. The Merkle tree cost is estimated based on 2^h leaves and 2^h-1 hash where h is the height of the Merkle tree [6]. Merkle tree is designed based on a BHT tree where every leaf node is labeled with a data block and every non leaf nodes is labeled with a cryptographic hash of the labels of its child nodes. Frequently accessed nodes are moved close to the root. For increasing security and efficiency of bitcoin transaction double SHA256 cryptographic hash is used. Since Merkle tree is based on the BHT principle it follows the principle of even the number of nodes for representing transactions in leaf nodes. In case of odd number of transactions, the last transaction will be replicated to achieve counterfeit [11]. Consider there are four transactions T0, T1, T2, and T3 which is represented as leaves of Merkle tree. Hashes of each transaction are stored in each leaf node of BHT as H0, H1, H2 and H3 instead of direct storage of transaction. Hash values of transactions are determined by H0=SHA256 (SHA256 (T0)) and H01=SHA256 (SHA256 (H0+H1)) for next level and so on. Figure 2 denotes the Merkle tree based on an even number of nodes construction [6]. Here the Merkle value is constructed by applying hashes to the transactions T0, T1, T2, and T3 as H0, H1, H2, and H3. The intermediate hashes like H01 and H23 is constructed and finally, the Merkle root hash is constructed based on intermediate hashes H01 and H23 as H0123.

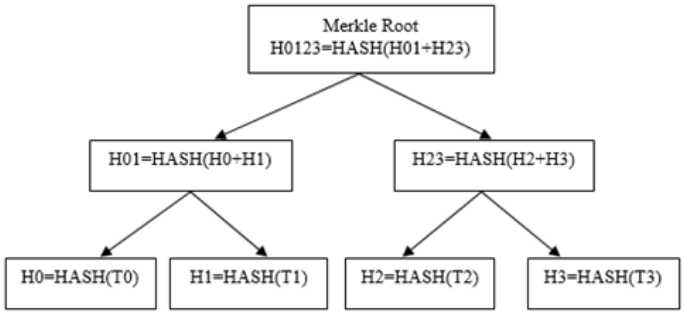


Figure 2: *Merkle tree construction based on even nodes count*

In Blockchain, there may be thousands of blocks and each block may contain thousands of records. To check whether the transaction is included or not is a tedious process. For this, the Bitcoin network uses light client mode verification that uses the SPV method. SPV retrieves only the block header and the transaction section to check whether the particular transaction that exists in the particular block or not. The transaction existence is verified by checking all the hashes from the bottom to the top of the tree. Starting from bottom, produce the hash of a particular transaction, then the intermediate hash is calculated at the middle level. Finally, the bitcoin Merkle root is computed and checked for its presence. In Blockchain, there may be thousands of blocks and each block may contain thousands of records. To check whether the transaction is included or not is a tedious process. For this, the Bitcoin network uses light client mode verification that uses the SPV method. SPV retrieves only the block header and the transaction section to check whether the particular transaction that exists in the particular block or not. The transaction existence is verified by checking all the hashes from the bottom to the top of the tree. Starting from bottom, produce the hash of a particular transaction, then the intermediate hash is calculated at the middle level. Finally, the bitcoin Merkle root is computed and checked for its presence.

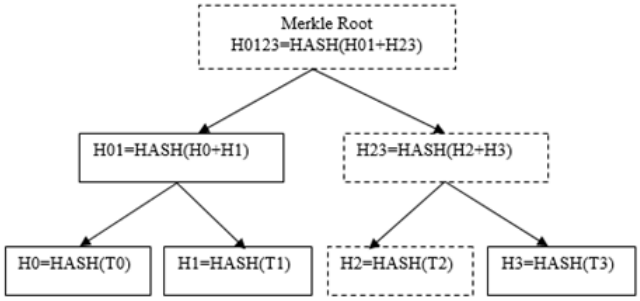


Figure 3: Merkle path proving transaction data inclusion

Consider an example if a transaction says T2, needs to be checked for its presence in the Merkle tree. The SPV client will apply the hash for transaction T2 which gets a result as H2 at the bottom of the tree. Then the intermediate hashes like H23 are calculated and finally, Merkle root H0123 is calculated. Now it compares the computed Merkle root value with the block header Merkle root in the block. If it matches then it

examines the Merkle path of the particular transaction as H2, H23, and H0123. From this technique, the SPV node can prove whether a particular transaction exists in that block or not. The above process of inclusion testing is depicted in Figure 3 where the testing section is represented as dotted one [6]. Here the Merkle root is constructed by applying SHA256 functions as binate. The following Figure 4 shows the calculation process of Bitcoin Merkle.

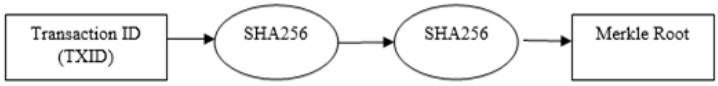


Figure 4: *Determining Bitcoin Merkle root using SHA256*

3. TRANSACTION VERIFICATION THROUGH COGNIZANT MERKLE

Cognizant Merkle is a new Merkle tree structure that is constructed based on HAT trie which achieves more speed compared to Bitcoin Merkle. This HAT trie [8] is represented by two alternative constructions namely hybrid and pure HAT trie. These variants are achieved using bucket management and splitting. The pure HAT trie employs the Burst-trie technique to achieve faster communication but it requires more space for application. In this technique, a full bucket is burst into almost n buckets where n denotes the size of the alphabet that are parented by a new trie. Here the bucket contains strings that share only one prefix which is removed and referenced by a single parent pointer. According to the leading character the strings are stored in pure buckets. The hybrid HAT trie applies the B-trie splitting algorithm that reduces the bucket cost and allows fast access by using multiple pointers for a single bucket. This hybrid HAT trie shares a single prefix. Here the prefix part is not removed rather it is stored along with the last part of the string. Consider a case where the string 'Eight' is to be stored in hybrid HAT trie the prefix part of the string (E) is separated from the original string and the remaining part of the string(ight) along with total characters(4ight) are stored in the container which is shown in Figure 5. The hybrid HAT trie may contain pure buckets also. The building and searching of HAT trie are based on top-down fashion. Initially, it starts with a single empty hybrid bucket which is occupied until full. The property of bucket is, it won't allow duplicate records. Hence when the bucket is full it is burst based on HAT trie type. After bursting, a hybrid bucket is created based on new parent trie along with pointers assigned to it. The bursting procedure continues as hybrid and old trie is moved as a grandparent. The bursting takes place based on a split point to achieve even distribution. Strings can be accessed in a linguist manner from top to bottom by accessing every leading character. The split point only decides the type of HAT trie. Parent pointers are set to empty in case of the empty bucket which is deleted afterward. The above process is represented in Figure 5 which contains both Pure and hybrid HAT trie implementation based on split point.

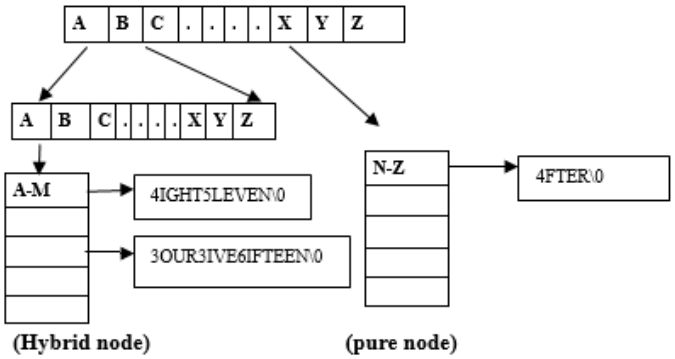


Figure 5: *Pure and Hybrid HAT-trie node representation for the values of Eight, Eleven, Four, Five, Fifteen and water.*

In Cognizant Merkle, the Merkle root is constructed based on the combination of base 64 encoding and fast bitwise hashing. Base 64 encoding [12] is used for transaction encoding because it can map all types of characters like ASCII, UTF 16 and others. This is useful in multiprotocol communication like Peer-to-Peer network that supports data loss prevention. The usage of fast bitwise hash prevents the adversarial attack. The following Figure 6 shows the Merkle tree implementation based on Cognizant Merkle

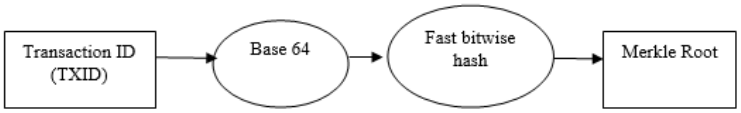


Figure 6: *Determining Cognizant Merkle root using Base64 and fast bitwise hash*

4. DOUBLE-SPEND DETECTION USING COGNIZANT MERKLE

The basic part of the Bitcoin network is a transaction. A transaction is a way of transferring bitcoins from one owner to another owner in the Bitcoin network electronically. The owners are represented by bitcoin addresses that are generated by a wallet. Transaction fundamental part is Unspent Transaction Output (UTXO). Consumed UTXO transaction is called transaction inputs and Produced UTXO transaction is called transaction outputs. A valid transaction is one whose input and output are accepted and stored in public ledger called blockchain [9]. The following Figure 7 shows the general transaction execution process.

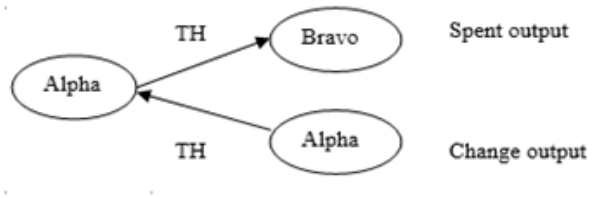


Figure 7: *General Transaction Executions*

The way of spending the same bitcoin more than once in multiple transactions is called as double-spending transaction or multi-spend transactions or irreversible transactions. Double-spending is easier in the Bitcoin network because the Bitcoin network uses digital money in the form of bits. Bits are easier to copy and replicated across multiple situations. So multi spending of coins is easier in Bitcoin network. The following Figure 8 shows the situation of double-spend attack. Here the same Transaction Hash (TH) value is replicated and spent in more than one recipient namely Bravo and Charlie where Bravo is the original recipient. This type of spending is called a double-spend attack. This attack arises due to dishonest miners or vendors or client or network propagation delay or through bitcoin exchange services.

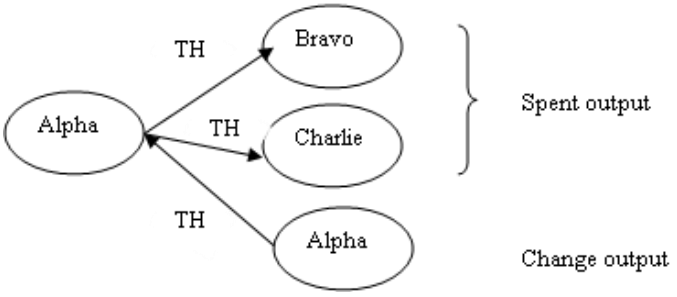
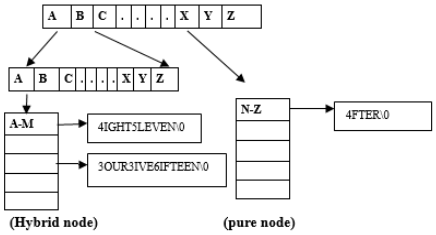


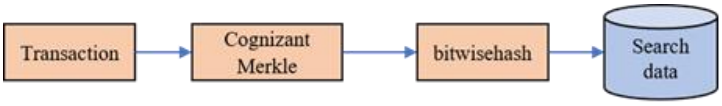
Figure 8: *Double-spend attack*

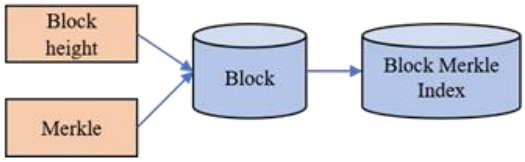
The following attacks like race attack, brute force attack, gold finger attack, selfish mining attack, Finney attack, block withholding attack, and fork after withholding attack will create double-spend attacks easily. Double-spend attack is a serious threat in the security of Bitcoin network. Any node can make faulty transactions by creating offline chain which in turn creates double-spent transactions in various block of the blockchain. This degrades the property of Peer-to-Peer network like data validation and integration. The purpose of blockchain implementation will also fail in this case because the records in blockchain are immutable and irreversible. Once the transaction like double-spend occurs it can't be removed or deleted in the previous block of the Blockchain. The tamper-proof property of blockchain will also fail due to the double-spending problem. Since the blockchain follows the principle of distributed ledger technology the entire network nodes will get fault information of blockchain and the whole network fails. If a double-spend transaction occurs in multiple blocks then it is difficult for identifying and analyzing the original transaction from the multi-spend transaction list. The user anonymity property of blockchain will facilitate the simulation of double-spend attacks further. The following proposed architecture Multi-Block Double spent Transaction Detection (MBDTD) which is depicted in Figure 9 will detect the double-spent transaction in multiple blocks efficiently. Here the multi-block double-spend detection of bitcoin transaction was easily identified by the combination of tempMerkle value, B-tree indexing, and crypto hashing algorithm.

The hybrid HAT trie appl ies the B-trie splitting algorithm allows fast access by using multiple pointers for asingle bucket. T prefix. Here the prefix part is not removed rather it is stored alor Consider a case where the string 'Eight' is to be stored in hybrid H (E) is separated from the original string and the remaining part of characters(4ight) are stored in the container which is shown in F contain pure buckets also. The building and searching of HAT tri Initially, it starts with a single empty hybrid bucket which is occu

The property of bucket is, it won't allow duplicate records. burst based on HAT trie type.After bursting, a hybrid bucket is cre with pointers assigned to it. The bursting procedure continues as grandparent. The bursting takes place based on a split point to ach be accessed in a linguist manner from top to bottom by accessing point only decides the type of HAT trie. Parent pointers are set to which is deleted afterward. The above process is represented in F and hybrid HAT trie implementation based on split point.







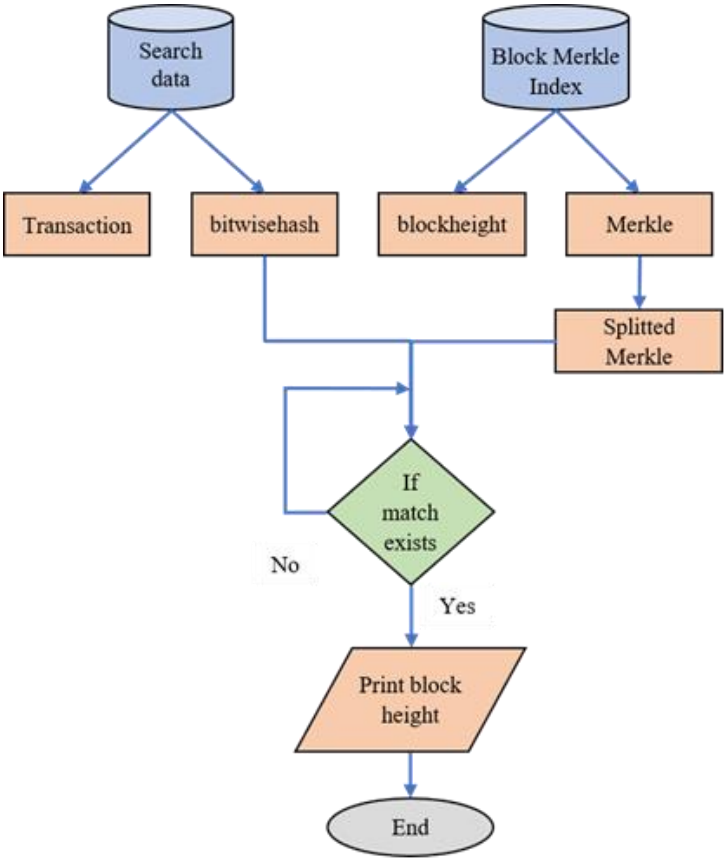


Figure 9: *Multi-Block Double-spent Transaction Detection (MBDTD) architecture*

Initially, the B-tree indexing method is used to construct blockMerkleindex where block height and blockMerkle values are arranged like B-tree. The reason for choosing B-tree indexing is that it provides a fast way of accessing Merkle value. Each node on the network uses B-tree keys to store the data (blockheight, Merkle) in ascending order. The property of B-tree is such that each of these B-tree keys has two references to their child nodes. The left side child node keys are always less than the current key value and right child node key values are always greater than the current key value. The time required for insertion, deletion and searching time is O (log n) [13]. Suppose if a transaction is to be tested for its existence or not in the blockchain, first the transaction is applied with Cognizant Merkle construction to construct temporary Merkle value. This temporary Merkle value is compared with the Merkle values of each and every block starting from genesis block to the current block using the cryptographic hash method. If a match found in any one of the block Merkle values then that corresponding block height is retrieved using B-tree indexing. Through this method the same transaction that exists in multiple blocks are identified. These above processes of checking are depicted in Figure 9. Table 1 denotes the table structure of MBDTD architecture. Here the block table attributes are used to construct blockMerkleindex table using B-tree index. Search data table contains the searching transaction list along with bitwisehash field which contain temp Merkle value determined using the technique of Cognizant Merkle.

Table 1: *Table structure followed in MBDTD architecture*

block

block height	transactions	transaction counter	previous hash	timestamp	blockhash	Nonce	Merkle

searchdata

transaction	bitwisehash

transbitwisehash

transaction	bitwisehash

blockdsdata

transaction	bitwisehash	block Merkle	blockheight

4.1 MULTI-BLOCK DOUBLE-SPENT TRANSACTION DETECTION(MBDTD) ALGORITHM

The following algorithm blockdoublespendcheck() detects the double-spent transaction in multiple blocks using blockMerkleindex table and searchdata table. Here for the given searching transaction, the temporary Merkle value is constructed using Cognizant Merkle. The blockMerkleindex is constructed based on blockheight and Merkle properties of the block table. These two tables are compared to get the final block result.

```
Algorithm blockdoublespendcheck()
{
    //getting a transaction that is repeated in multiple blocks of blockchain

     for every value of transaction in searchdata
     {
         // reading every transaction in the search data list
         for every matching value of bitwisehash of particular transaction in searchdata
         {
              //finding the tempMerkle value of transaction using Cognizant Merkle

              for each value of block height in blockMerkleindex
              {
                   // comparing tempMerkle value with all Merkle values of blockchain
                   // to find match. If match exists it returns the corresponding blockheight
                    // retrieving Merkle value of each block

                   temp = Merkle value of blockMerkleindex

                   // if there is no Merkle value skip the checking part

                   if temp=”None”
```

```
                    continue

                // assign the split point of Merkle value

                 n = 3

                // split the Merkle values into set of 3 digits and stored in array

                 out = splitted Merkle value based on 3 digits

                // compare the tempMerkle value with Merkle splitted values

                 s = hashcheck(out, bitwisehash)

                // if match exists return the corresponding blockheight of the Merkle value else
                // repeat the process of checking for next value

                 if (s==1)
                     print blockheight of blockMerkleindex
                 else
                     continue
             }
         }
     }
}


 Algorithm hashcheck( lys , element)
{
     S=SHA256(element)
     for each value of lys
     {
         S1=SHA256(lys[i])
        if (S == S1)
            return 1
     }
     return 0
}
```

5. RESULTS AND DISCUSSION

The proposed work was implemented in the Intel i3 Core processor with 2 GB RAM and Ubuntu 32 bit operating system. Python is used as a front end and PostgreSQL database as a backend server. The work was successfully carried out with 10 blocks of totally 736 transaction records. A total of 83 transaction records were used for searching the double-spend attempt in multiple blocks. Each and every search transaction record was taken and successfully checked along with 10 blocks of Merkle values. Totally 1826 records of comparison were done and the double spend identification result is stored in a blockdsdata table. This table includes details like searching transaction identifier, their corresponding calculated temp Merkle value, comparing Merkle value of each block and the resultant blockchain height. The results were collected and tested which provides 100% accuracy.

6. CONCLUSION AND FUTURE WORK

This paper has discussed the detection of double-spending of transaction in multiple blocks of the Blockchain. First, it starts with the discussion about the blockchain architecture. Next it discussed the transaction verification process that is carried out by Bitcoin Merkle. The process is moved on to the discussion of the transaction verification process carried by Cognizant Merkle. Finally, this paper examines the serious issue of double-spend problem in multiple blocks and propose a new architecture named Multi-Block Double-spent Transaction Detection (MBDTD). Next, the paper has deliberated about how it handles the multi-block double-spend detection using MBDTD architecture effectively. Finally, this concluded that if MBDTD method is applied before transaction inclusion into the block, then the multi-block double-spend attempt will definitely fail. The future work focuses on handling the performance improvements on security issues of double-spend based on selfish mining.

REFERENCES

[1] The blockchain or Distributed trust. "https://euagenda.eu/publications/the-blockchain-or distributed-trust". [online; accessed 25-oct-2019]

[2] A. Murugan and J. Vijayalakshmi. "Discovering the Bitcoin Double-spend using Lost Agreement Amount", International Journal of Recent Technology and Engineering, Vol. No. 8(3), PP 3764-3770, 2019.

[3] Understanding the basics of blockchain. "https://data-flair.training/blogs/basics-of-blockchain-technology/" [online; accessed 24-oct-2019]

[4] Understanding Blockchains and Bitcoin part 2 Technology, "https://luxsci.com/blog/understanding-blockchains-and-bitcoin-technology.html".[Online; accessed 24-oct-2019]

[5] Andreas M. Antonopoulos." Mastering Bitcoin: unlocking digital cryptocurrencies", O'Reilly Media, 2014.

[6] Vijayalakshmi J. and Murugan, A. "Revamp Perception of Bitcoin Using Cognizant Merkle", In Proceeding of Emerging Research in Computing, Information, Communication and Application, Springer,PP 141-150, 2019

[7] Merkle R.C. "A digital signature based on a conventional encryption function.", Advances in Cryptology, Springer, Berlin, Heidelberg, PP 369-378, 1987.

[8] Askitis Nikolas, and Ranjan Sinha. "HAT-trie: a cache-conscious trie-based data structure for strings." In Proceedings of the thirtieth Australasian conference on Computer science, Australian Computer Society, Vol.No.62, PP 97-105, 2007.

[9] A. Murugan and J. Vijayalakshmi. "Preventing the bitcoin Double-spend using Transaction Hash and Unspent Transaction Output." International Journal of Recent Technology and Engineering, Vol.No.8 (3), PP 3771-3776, 2019.

[10] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System","https://bitcoin.org/bitcoin.pdf", 2008.

[11] Knecht Markus, Willi Meier, and Carlo Nicola. "A space-and time-efficient Implementation of the Merkle Tree Traversal Algorithm", arXiv, Vol.No. 1409(4081), 2014.

[12] Stackoverflow questions, ”https://stackoverflow.com/questions/4070693/what-is-the-purpose-of-base-64-encoding-and-why-it-used-in-http-basic-authentica” [online;access date: 13- feb-2018]
[13] How database B-tree indexing works,”https://dzone.com/articles/database-btree-indexing-in-sqlite”.[Online; accessed 11-jan-2019]