Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 09, No. 02, June 2023.

STATEFUL LAYERED CHAIN MODEL TO IMPROVE THE SCALABILITY OF BITCOIN

Dalia Elwi[1, *], Osama Abu-Elnasr[1], A. S. Tolba[1, 2] and Samir Elmougy[1]

(Received: 19-Jan.-2023, Revised: 26-Mar.-2023, Accepted: 1-Apr.-2023)

ABSTRACT

Bitcoin becomes the focus of scientific research in the modern era. Blockchain is the underlying technology of Bitcoin because of its decentralization, transparency, trust-less and immutability features. However, blockchain can be considered the cause of Bitcoin scalability issues, especially storage. Nodes in the Bitcoin network need to store the full blockchain to validate transactions. Over time, the blockchain size will be bulky. So, the full nodes will prefer to leave the network. This leads to the blockchain being centralized and trusted and the security will be adversely affected. This paper proposes a Stateful Layered Chain Model based on storing accounts' balances to reduce the Bitcoin blockchain size. This model changes the structure of the traditional blockchain from blocks to layers. The experimental results demonstrated that the proposed model reduces the blockchain size by about 50.6 %. Implicitly, the transaction throughput can also be doubled.

KEYWORDS

Bitcoin, Cryptocurrency, Blockchain, Stateful, Layered, Chain, Scalability, Storage, Throughput.

1. INTRODUCTION

Blockchain is a distributed, secured, scalable and transparent ledger that permanently records transactions among decentralized network participants [1]. It avoids relying on a trusted third party to validate, verify and process the transactions. Besides cryptocurrency, blockchain is a base technology for many other fields, such as medicine, social, the internet of things, supply chain, voting, information sharing and file storage.

The digital asset, which is cryptographically secured, is called cryptocurrency. Cryptocurrency is a blockchain-based technology to transfer funds between users in a secure and transparent mode [2]. Currently, there are thousands of cryptocurrencies, but according to the coin market cap [3], the best cryptocurrency is Bitcoin [4].

Bitcoin depends on blockchain technology to store transactions between users in a distributed network. Full nodes in Bitcoin store the entire blockchain starting from the genesis block, so that they can retrieve historical activities, search transactions, validate new transactions and calculate balances [5]. Over time, the size of the blockchain will grow exponentially and it will be difficult for full nodes to download and store the complete Bitcoin blockchain. Figure 1 shows the growth of Bitcoin blockchain size from 2009 to 2022. According to Bitcoin visuals' statistics [6], In March 2022, the number of Bitcoin blocks is over 71×10^4, the chain size is 366 GB and in the last two years, the annual growth rate is around 15.8 %. As a result of this huge storage, there is a big reduction in the number of full nodes that have enough resource capabilities to store the entire blockchain. Therefore, the blockchain network will be more centralized.

Bitcoin transactions are structured to make the blockchain stateless, which means that it does not keep the addresses' balances. However, the balances can be retrieved by tracking the address transactions through the entire blockchain. So, we propose a Stateful Layered Chain Model based on storing the addresses' balances. In the proposed model, the transaction and block structures are completely changed. The block is replaced by a layer that is smaller in size. Our model reduces the chain size to solve the storage scalability problem, keeps transactions easy to be verified and implicitly increases the transaction throughput. Our model allows full nodes with limited storage capabilities to process new transactions without the need for storing the entire ledger. Therefore, it maintains the decentralization of the network and at the same time keeps the transaction verification simple as in

1. D. Elwi (Corresponding Author), O. Abu-Elnasr, A.S. Tolba and S. Elmougy are with Computer Science Department, Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt. Email: dalia_elwi@mans.edu.eg
2. A.S. Tolba is with New Heliopolis Engineering Institute, Cairo 11829, Egypt.

traditional Bitcoin. Also, our model can increase the transaction throughput implicitly if we make the layer size similar to the block size, allowing the storage of more transactions.

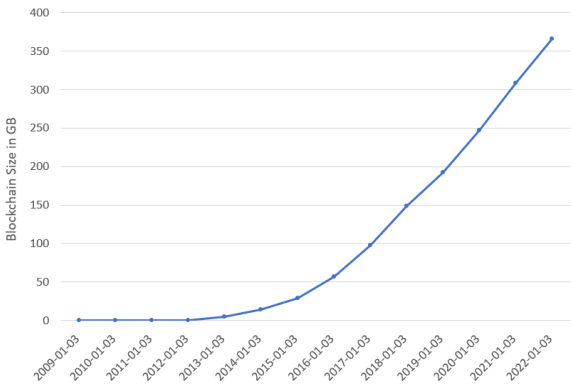

Figure 1. The cumulative size of Bitcoin blocks [6].

The paper is organized as follows: Section 2 gives a simple background about Bitcoin blockchain structure. Section 3 introduces the related work. Section 4 explains the proposed Stateful Layered Chain Model. Section 5 shows the implementation and experimental results. Section 6 provides the conclusion and future work.

2. BACKGROUND

A peer-to-peer network is used in Bitcoin for block and transaction exchange. Peers have three types: Simplified Payment Verification (SPV) node, Light-weight Client node and Full node. SPV node, which is called a *thin client*, is a client that keeps a copy of the block headers. It uses Merkle root to ensure that the block is containing the required transactions [7]. The light-weight Client node does not store the blockchain. Instead, it creates purchase or money transfer transactions and sends them to full nodes [8]. Full nodes, which are called *miners*, store and serve the entire blockchain to guarantee the highest level of security by verifying all the blocks from the genesis block [5]. They are responsible for transaction authentication, the mining process to create new blocks and getting rewards.

2.1 Blockchain Structure

Blockchain is a linked list of blocks. Each block has a pointer to the previous block. The block is divided into two parts: block header and block body [9]. Block header contains information about the block, such as its version, its hash, previous block hash, Merkle root, timestamp, difficulty and Nonce, as illustrated in Table 1. On the other hand, transactions are recorded in the block body [10]. Figure 2 shows the Bitcoin blockchain structure.

Table 1. Fields of the block header.

Fields	Description
Version	The software version number is used for upgrading the protocol.
Hash	It is a 256-bit unique binary number to represent the block.
Previous Hash	It is a hash number of the previous block.
Merkle Root	The root of the Merkle Tree [11] is generated from the transactions in the block.
Timestamp	The time of the block creation is presented in UNIX time.
Difficulty	The difficulty value is used in the mining process to create a new block.
Nonce	It is a counter-value that makes the block hash less than or equal to the target value.

2.2 Transaction Structure

The transaction is a transfer of value from a source address *(input)* to a destination address *(output)*. It has six fields: version, input counter, inputs, output counter, outputs and lock time [12], as illustrated in Table 2. One of the most important concepts of Bitcoin is privacy, which means that the user's balance should be unknown to the network's miners. Bitcoin maintains this concept by relying on a stateless blockchain ledger that does not record any balances.

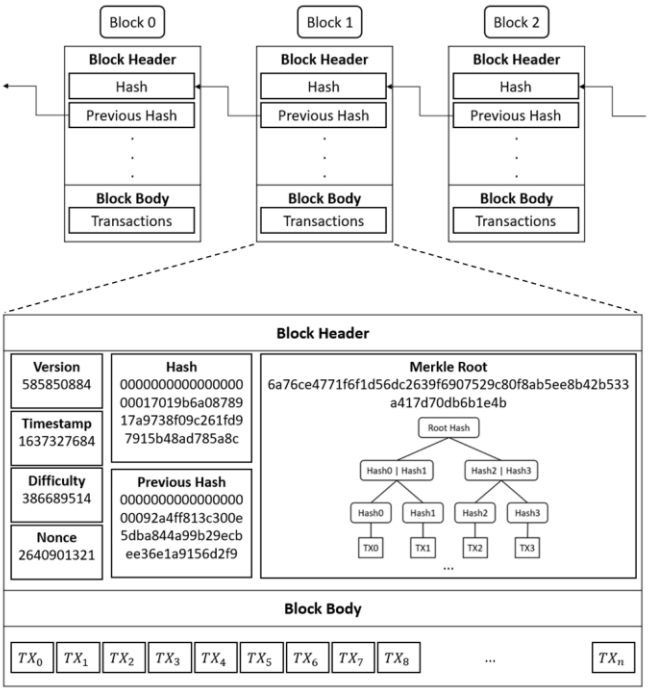

Figure 2. Bitcoin blockchain structure.

Table 2. Fields of transaction structure.

Fields	Description
Version	It is a version number of the rules that a transaction follows.
Input Counter	The number of inputs.
Inputs	One or more transaction inputs.
Output Counter	The number of outputs.
Outputs	One or more transaction outputs.
Lock Time	The locked duration is where the outputs of the transaction cannot be spent. It can be UNIX time or block number.

The user's balance is implicitly divided into many Unspent Transaction Outputs ($UTXO_s$) which can be used as inputs in other transactions, provided that the user can spend only the $UTXO_s$ belonging to him/her. Despite that, the user's balance can be calculated by scanning the entire blockchain and accumulating all $UTXO_s$ belonging to him/her. In addition to the entire blockchain, a set of $UTXO_s$ is stored in full nodes which are used in transaction authentication. Figure 3 shows the mechanism of Bitcoin transactions. Transaction input is composed of four fields: outpoint, unlocking script size, unlocking script and sequence number. Transaction output is composed of three fields: amount, locking script size and locking script [12]. Table 3 and Table 4 illustrate both transaction input and output, respectively.

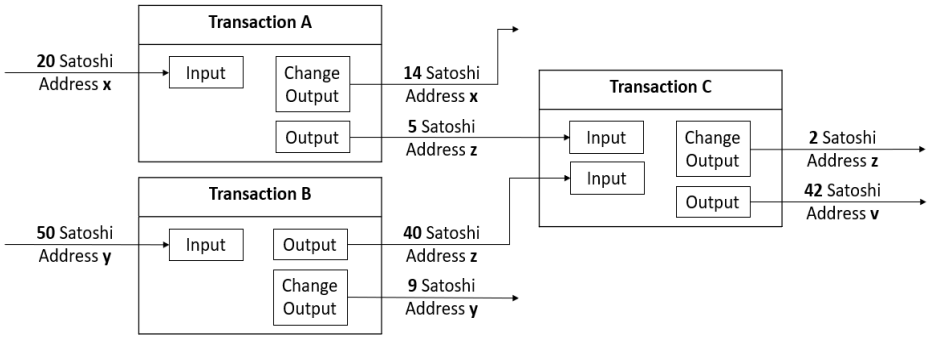

Figure 3. Bitcoin transactions*.

**Transaction A transform 5 Satoshi from address x to address z, return 14 Satoshi to himself and the fee is only 1 Satoshi. Transaction B transforms 40 Satoshi from address y to address z, returns 9 Satoshi to himself and the fee is only 1 Satoshi. Transaction C transforms 42 Satoshi from address z to address v, returns 2 Satoshi to himself and the fee is only 1 Satoshi.*

Table 3. Fields of the transaction input structure.

Fields	Description
Outpoint	It is a pointer to a previous transaction containing the UTXO to be spent. It is composed of the previous transaction hash and UTXO index.
Unlocking Script Size	The length of unlocking scrip in bytes.
Unlocking Script	It is used to unlock the locking script of UTXO. It is a proof of the ownership of the locking script.
Sequence Number	It is a number used to verify the lock time.

Table 4. Fields of the transaction output structure.

Fields	Description
Amount	The transferred value in Satoshi.
Locking Script Size	The length of locking scrip in bytes.
Locking Script	It is used to lock the transferred amount, so that only those with the unlocking script can open the lock.

2.3 Transaction Authentication

When the new transaction is created, miners search the entire blockchain to find the $UTXO_s$ that are used as inputs for this new transaction. So, miners need to store the entire blockchain. Then, UTXO must be authenticated to ensure the sender's ownership of the transferred funds.

2.3.1 Keys and Addresses

Public and private keys are the main requirements for creating any transaction. The private key is a 256-bit binary number that is generated randomly using Secure Hash Algorithm SHA256 [13]. On the other hand, a public key is a unique number that is calculated from a private key using a one-way function called Elliptic Curve Multiplication function [14]. So, the private key cannot be calculated from the public key, but the connection between them can be proved using the digital signature without having to reveal the private key [15]. Bitcoin addresses are calculated by hashing public keys.

2.3.2 Digital Signature

A digital signature is a number that is calculated from the private key and the transaction data. It is used to prove the ownership of the outputs, which are used in the transaction as inputs [16]. The authentication process has two steps: *(1) signing* for digital signature creation and *(2) verifying* to prove that the digital signature and the public key were generated from the same private key [17]. The signing and verifying steps are shown in Figure 4.

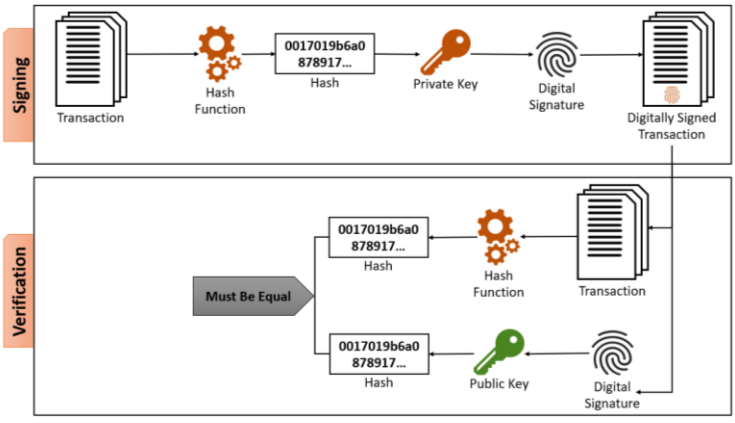

Figure 4. Authentication process.

2.3.3 Scripts

Execution of Bitcoin scripts is based on Reverse Polish Notation which uses the stack to perform this execution [18]. Every transaction contains locking and unlocking scripts, which are called *scritPubKey* and *scriptSig*, respectively. They are executed as one script. If the execution result is true, then the transaction is valid. Standard types of transaction scripts are: Pay to Public Key Hash *(P2PKH)*, Public

Key, Multi-signature, Pay to Script Hash *(P2SH)* and Data Output *(Op-Return)*. Most Bitcoin transactions are based on the *P2PKH* script [19]. The locking and unlocking scripts for *P2PKH* are shown in Figure 5.

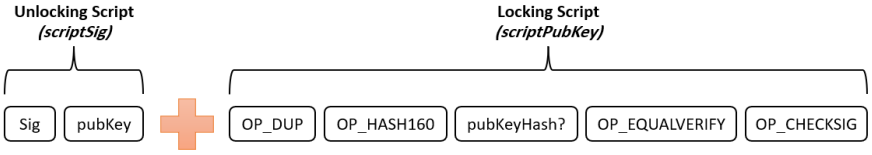

Figure 5. *P2PKH* locking and unlocking scripts [20]*.

**scriptSig is provided by the transaction inputs to unlock the scritPubKey of some previous UTXOs.*

2.3.4 Transaction Fee

The miner creating a new block and recording it into the blockchain should take a fee for each transaction in this new block. The fee can be determined as a transaction output during the creation of the transaction. Also, if it is not determined directly in the transaction, it can be calculated by the miner during the creation of the block. The fee is calculated by subtracting the amount of transaction outputs from the total value of transaction inputs, where the remainder would be the fee [12]. If the value of inputs exceeds the amount of the output, the difference in value is considered the transaction fee. Therefore, the output, which is called "Change Output", must be added to the transaction outputs to force the miner to get a specific value of fee [21]. The change output amount is returned to the sender's address. So, the fee can be calculated as in the following equation:

$$Fee = Inputs\ Value - (Outputs\ Amount + Change\ Output\ Amount)$$

The value of the transaction fee is calculated automatically by the wallet. Miners choose the transactions with the highest fee to be added to the block.

3. RELATED WORK

Based on the view of the blockchain storage expansion problem, the existing suggested models to solve this problem are divided into two categories: *off-chain* solutions and *on-chain* solutions.

3.1 Off-chain Solutions

Off-chain solution is based on storing a large amount of block data in the off-chain storage to reduce the size of the main blockchain. The off-chain storage is maintained by a secondary network. So, there are two networks: one for the off-chain and the second for the main chain. Intercommunication between the two networks adds an overhead to the system's efficiency.

Guy et al. [22] used the Distributed Hash Table (DHT) to store the essential data. On the other hand, the SHA-256 hashes are recorded in the main blockchain to reference DHT data. DHT is maintained by special nodes that differ from the miner nodes. So, this method requires some amount of third-party trust. Qiuhong et al. [23] used a distributed storage called InterPlanetary File System (IPFS) to store the main transaction data. On the other hand, the block contains the transaction hashes as a reference to the IPFS transactions. In this model, a little reduction in the system efficiency is caused as a result of the transaction requests from IPFS nodes. Also, it needs to trust another network that stores IPFS data. Soharab et al. [24] proposed two types of blocks: raw block to store transaction data and hash block to store transaction hashes. Both are connected by a Content Identifier (CID). Raw blocks are stored in the IPFS network as a secondary blockchain and hash blocks are stored in the main blockchain. Raw and hash blocks are created by the miners. So, this model is more complex than the traditional Bitcoin blockchain model. Joseph and Thaddeus [25] proposed a network of micropayment channels to process massive transactions off-chain. The channel is created between two users to exchange funds between them more than once. After the restriction on the timestamp is satisfied, only one transaction is created with the state of the users' balances and broadcast to the main chain. Xiaoqing et al. [26] proposed an Efficient Storage Scheme (ESS) based on the distribution of $UTXO_s$ in Bitcoin blocks. ESS assigns a UTXO weight for each block according to the number of its $UTXO_s$ and its height. Blocks with higher UTXO weight will keep the complete block information, while

blocks with lower UTXO weight will only save the block header and delete the block body. The deleted $UTXO_s$ are stored in a new database in the form of snapshots called *deleteutxo*. When the new transaction used UTXO from a pruned block as input, the *deleteutxo* database is searched to determine whether the input has an unused record. This scheme causes an additional cost, but it is still in an acceptable range.

3.2 On-chain Solutions

On-chain solution is based on changing the structure of the blockchain to reduce its size. So, there is only one network to store the main blockchain in a different structure. Some solutions are just by pruning the block data and some of them are based on completely changing the blockchain structure. Roman et al. [27] presented a pruning scheme called Coin-Prune. It is based on having honest miners to create a snapshot of the current blockchain state using the UTXO set, periodically. Miners announce and reaffirm the snapshots publicly. Therefore, they do not have to store all historical data. Instead, they rely on the recent snapshot. This scheme has a security limitation; e.g. the potential for removing illicit content from the UTXO set. Also, it needs trust assumptions. Martin et al. [28] demonstrated Functionality-Preserving Local Erasure (FPLE) to erase undesirable transaction outputs from the blockchain and store their references in an erasure database. The proposed FPLE ensures that all nodes are synchronized with the network even when the erased outputs are later spent. It assumes that nodes ignore the unconfirmed transaction the validation of which depends on erased data. A new strategy for transaction verification is proposed to deal with transactions that are relying on erased data. Xiaohai et al. [29] proposed a new jigsaw-like data-reduction approach called Jidar. Each node in Jidar does not need to store the complete block. Instead, it only stores the transactions of interest and the relevant Merkle branches. So, the proof is needed from the transaction proposer to verify the new transactions. Jidar also provided a mechanism to get a complete block based on cohering all the data fragments from the other nodes just like stitching all the pieces into a complete jigsaw puzzle picture. There are not any trust assumptions in Jidar, but it does not maintain the decentralization concepts of the traditional Bitcoin system. Serguei [30] suggested changing the structure of the blockchain from the linked list of blocks to a Directed Acyclic Graph (DAG) ledger. He used this structure to implement IOTA cryptocurrency. DAG ledger is consisting of transaction units. Each unit presents a single transaction from a single user. The new transactions need to approve the legitimacy of two previous transactions. In this structure, nodes do not require a large storage to store the complete DAG. In contrast, they need a higher coding requirement to load the DAG ledger. Li et al. [31] proposed an algorithm to downsample the block body to reduce the blockchain size. This algorithm is based on the distribution of the interval between the approval of a UTXO and its use. So, based on this analysis, transactions are discarded after a period decided by the algorithm. This method loses very little broadcast accuracy to reduce storage. Ryunosuke et al. [32] proposed a new architecture called Trail to reduce the size of the blockchain. The trail allows nodes to store the block without transactions. In contrast, a client who creates a transaction needs to send proof of its assets. So, unlike traditional blockchains, the clients store additional data. Ulfah et al. [33] summarized the block and compressed the result to reduce the blockchain size. At the same time, they make transaction verification easier than traditional Bitcoin. They developed the summarization by rearranging the block headers and adding parameters according to this arrangement. After that, the summary block is compressed with the deflate compression algorithm which is a composite of LZ77 and Huffman algorithms. The deflate compression algorithm is used to overcome the problems that occurred due to the summarization.

4. PROPOSED SOLUTION

4.1 Stateful Layered Chain Model

A Stateful Layered Chain Model is proposed to reduce the size of the Bitcoin blockchain by changing the transaction structure and the block structure. Like a blockchain, a Stateful Layered Chain is a distributed ledger that is recorded in different miners in a peer-to-peer network. All miners must have the same replica of the ledger to avoid reliability. So, our model is not relying on third parties. $UTXO_s$ set is no longer stored. Instead, accounts with their balances are recorded. Transactions are no longer based on inputs and outputs. However, they are based on receivers and senders. Table 5 presents the proposed transaction structure.

In our model, there are two types of layers: *Last State Layer* (LSL) and *Default State Layer* (DSL). LSL stores the state of all accounts, while DSL stores the transactions. LSL is only one layer that is

Table 5. Fields of transaction structure.

Fields	Description
Sender Address	It is a sender account public key in a short format.
Sender Unlocking Script	It is a script used to unlock the locking script of the sender account balance to authenticate the ownership.
Receiver Address	It is a receiver account public key in a short format.
Receiver Locking Address	It is a script used to lock the transferred value, so that only those with the unlocking script can open the lock.
Value	It is the transferred value in Satoshi.

updated every time a DSL is created to save the last state for each account. DSL is the alternative form of a block to store transactions, knowing that DSL reduces the size of the block and at the same time keeps the number of transactions as in the block. Stateful Layered Chain is composed of a linked list of DSL layers. So, each DSL layer points to the previous one. Figure 6 and Figure 7 show the proposed LSL and DSL structures in JavaScript Object Notation (JSON) format, respectively. Table 6 and Table 7 illustrate the DSL and LSL structures.

Figure 6. The proposed LSL structure in JSON format.

Table 6. Fields of LSL structure.

Fields	Description
Hash	It is a 256-bit unique binary number to represent the LSL layer.
n Accounts	The number of client accounts.
Size	The length of the LSL in bytes.
Accounts	Client accounts.

Table 7. Fields of DSL structure.

Fields	Description
Hash	Hash number of the updated LSL layer.
Previous Hash	Hash number of the previous DSL layer.
Fee	It is the amount of fee that the miner gets.
n Receivers	The number of receiver accounts.
Nonce	It is a counter value that makes the block hash less than or equal to the target value.
Size	The length of the DSL in bytes.
Height	The height of the DSL in the chain.
Receivers	Receiver accounts.

After transactions are broadcast through the network, they must be authenticated. The transaction is authenticated by using the sender *unlocking script* to unlock the *locking script* of his/her account contained in LSL to guarantee the ownership of the transferred funds. As in the traditional Bitcoin, the

Proof of Work (PoW) consensus algorithm is used in the mining process. PoW is used to ensure network consistency, ensure a high level of security and obtain approval on the new DSL from all miners. DSL transactions must guarantee the following conditions:

1) There are no two receivers with the same address.
2) Receiver address is not the same as one of the senders' addresses.
3) Sender cannot send funds greater than or equal to his/her balance.
4) 0Sender account must be included in the LSL.


```
"Hash": "00000000000000000000017019b6a0878917a9738f09c261fd97915b48ad785…",
"Previous Layer Hash": "00000000000000000000092a4ff813c300e5dba844a99b29ecb…",
"Fee": 1000,
"n Receivers": 3,
"Nonce": 2640901321,
"Size": 500,
"Height": 150001,
"Receivers": [
{
        "Address": "1Hc5hmiZyx9FYY9hAV4MMBgsjJzwLoMUmP",
        "Locking Script": "76a914536ffa992491508dca0354e52f32a3a7a679a53a88ac",
        "n Senders": 2,
        "Senders": [
        {
                "Address": "35iMHbUZeTssxBodiHwEEkb32jpBfVueEL",
                "Unlocking Script": "0315d70a082f5669614254432f2cfabe6d6da926…",
                "Value": "100000"
        },
        ……//other Senders //……
        ]
},
……//other Receivers//……
]
```


Figure 7. The proposed DSL structure in JSON format.

Table 8, Table 9 and Table 10 illustrate LSL account, DSL receiver and DSL sender structures, respectively. Stateful Layered Chain Model provides some concepts, including:

1) **Layer Reduction:** Since the transaction authentication process needs only the LSL to check the balance and to ensure the sender's ownership of the transferred funds, miners can remove the earliest DSL layers from the chain.
2) **Node Bootstrap:** New full nodes just need to download the LSL to begin the mining process.
3) **User Privacy:** For more privacy, users can split their balance into many accounts with different addresses.

Table 8. Fields of LSL account structure.

Fields	Description
Address	It is an account public key in a short format.
Locking Script	It is a script used to lock the balance value, so that only those with the unlocking script (owner account) can open the lock.
Balance	It is the amount value in Satoshi owned by the account.

Table 9. Fields of DSL receiver structure.

Fields	Description
Address	It is a receiver account public key in a short format.
Locking Script	It is a script used to lock the transferred amount, so that only those with the unlocking script can open the lock.
n Senders	It is the number of senders sending funds to the receiver account.
Senders	Sender accounts.

Table 10. Fields of DSL sender structure.

Fields	Description
Address	It is a sender account public key in a short format.
Unlocking Script	It is a script used to unlock the locking script of the account balance to authenticate the ownership.
Value	It is the transferred value in Satoshi.

4.2 Stateful Layered Chain Creation Methodology

Stateful Layered Chain starts with the genesis layer which includes the first transaction. The LSL layer is created to present the state of the accounts in the genesis layer. After that, DSL can be created as shown in Figure 8. Miners in the network choose transactions from the Memory Pool for validation and authentication. Then, the valid transactions are added to DSL as the receiver/sender structure and LSL is updated. The sender balance is decreased and the receiver balance is increased. At the end, a miner who calculates the hash number of LSL firstly broadcasts the DSL to the rest of the network.

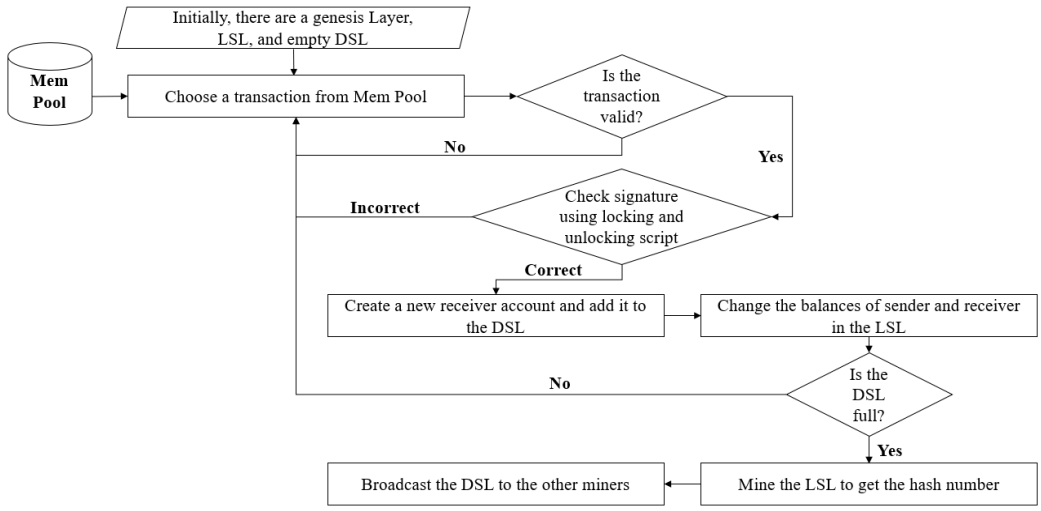

Figure 8. DSL creation flowchart.

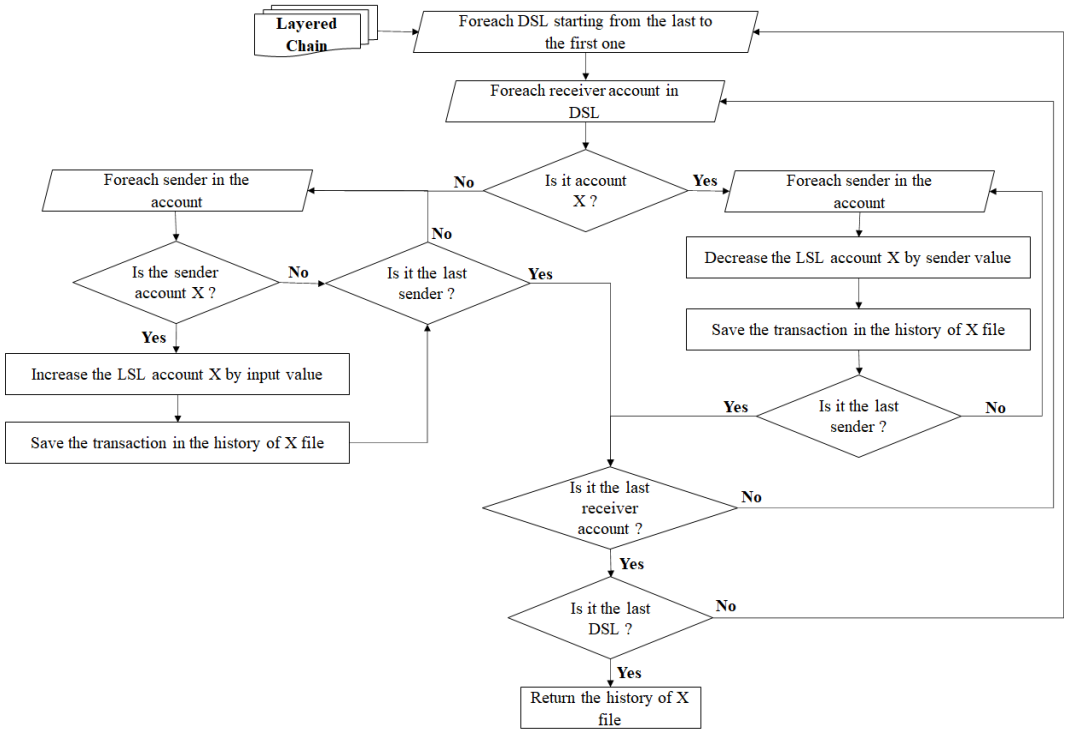

Figure 9. History of account x flowchart.

Stateful Layered Chain Model provides the ability to retrieve the history of LSL accounts. Figure 9 shows the flowchart of how to retrieve the history balances of account x. Also, Stateful Layered Chain Model provides a method to retrieve the LSL at any height of the chain. This method is shown in Figure 10.

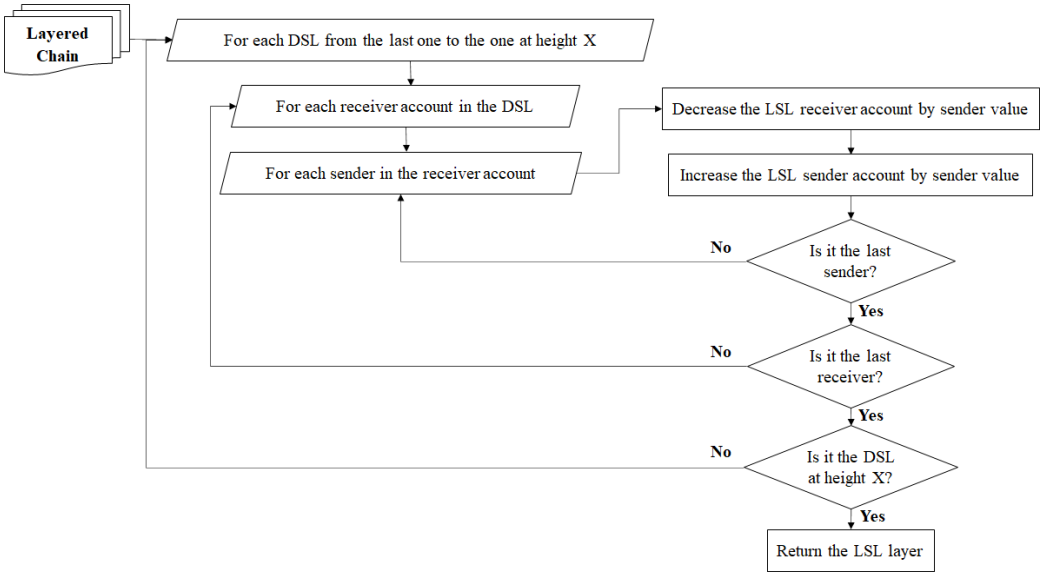

Figure 10. Deriving LSL at height x flowchart.

4.3 Example

The following example demonstrates the main concepts of the proposed model. It supposes that:

1) Initially LSL contains *six* accounts (A, B, C, D, E, F) with a balance equal to 100 Satoshi.
2) There are nine valid transactions in the Memory Pool, as illustrated in Table 11.
3) DSL can record only three transactions.

Figure 11 shows the Stateful Layered Chain before and after adding the transactions in DSL layers.

Table 11. Valid transactions in the memory pool.

Transaction number	Transaction with Amount = 10
1	A to B
2	B to C
3	C to D
4	A to C
5	A to F
6	F to E
7	B to F
8	F to B
9	F to A

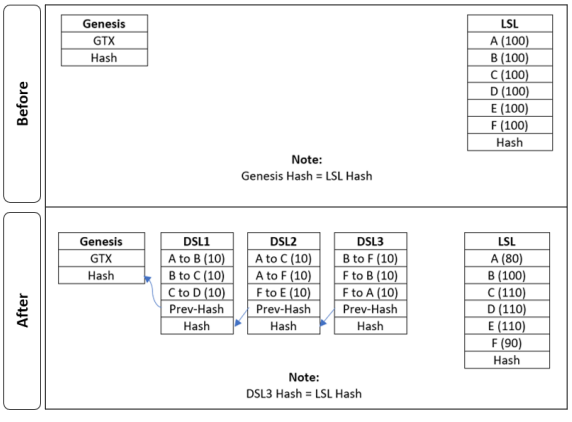

Figure 11. Stateful layered chain before and after DSL layers' creation.

5. IMPLEMENTATION AND EVALUATION RESULTS

5.1 Implementation Environment

The Stateful Layered Chain Model is implemented using 100 blocks from the Bitcoin system. The blocks are from a height of 710327 to 710426 which are the latest blocks at that time. We developed a transformation process using the C# .Net framework to transform the blocks into DSL layers. The transformation process is divided into four stages, as shown in Figure 12.

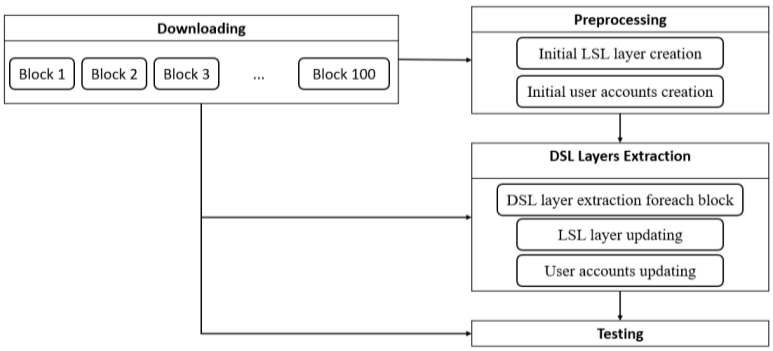

Figure 12. Transformation process stages.

1) **Downloading:** Blockchain data API [34] is used to download a single block in JSON format. Blocks are downloaded individually.
2) **Pre-processing:** In this stage, the initial LSL layer and initial user accounts are derived from the existing blocks. Algorithm 1 and Algorithm 2 are used to create initial LSL and initial user accounts, respectively.
3) **DSL Layer Extraction:** The DSL Layer for each block is extracted from the existing blocks, as presented in Algorithm 3. After the extraction of each DSL layer, the LSL layer and user accounts are updated. Algorithm 4 and Algorithm 5 are used to update LSL and user accounts.
4) **Testing:** User accounts are updated for each block and compared with the corresponding user accounts for each DSL layer to ensure that they have the same balances.

Algorithm 1. Initial LSL layer creation

Input: TXPOOL, Blocks
Variable:
TXPOOl: Pool of all transaction indexes in the existing blocks
Account: Account to be added in LSL
Output: Initial LSL Layer
BEGIN
LSL layer = empty
FOR EACH Block from the oldest to the newest
FOR EACH Transaction in the Block
FOR EACH Input in the Transaction
IF it is not coin base transaction && the input index does not exist in *TXPOOL*
IF there is *Account* with the same input address in the layer
Account value = *Account* value + input value
ELSE
Create a new *Account* with the input address, value and script.
Add *Account* to the LSL layer
END IF
END IF
END FOR EACH
END FOR EACH
END FOR EACH
Calculate the hash number of the Initial LSL Layer
Save the Initial LSL Layer as a file.
END

Algorithm 2. Initial user accounts' creation

Input: Blocks
Variable:
UID: A number to identify the user
User: User account including addresses list and the balance for each address
Output: List of Users
BEGIN
UID= 1
List of Users = empty
FOR EACH Block from the oldest to the newest
FOR EACH Transaction in the Block
Create a new $User$ with empty addresses list
FOR EACH Input in the Transaction
IF it is not coin base transaction && an input address does not exist in the *User* addresses list
Add the address to the *User* addresses list
END IF
END FOR EACH
IF the *User* addresses list is not empty
the user number = *UID*
UID = *UID* + 1
Add the *User* to the List of Users
END IF
END FOR EACH
END FOR EACH
Remove redundant users
Define users' balances for their addresses
Save the List of Users as a file
END

Algorithm 3. DSL layer extraction

Input: Block, List of Users, LSL Layer
Variable:
RAccount: Receiver Account to be added in DSL Layer
SAccount: Sender Account that sends funds to the *RAccount*
Output: New DSL Layer, Updated LSL Layer, Updated List of Users
BEGIN
Create New Empty DSL Layer with Zero *RAccount*
FOR EACH Transaction in the Block
IF the Transaction is not coin base transaction
Take a fee from the sender.
END IF
FOR EACH Output in Transaction Output List
IF Output address does not exist in Transaction Inputs addresses
Create New DSL Layer *RAccount*
RAccount Address = Output Address
RAccount Script = Output Script
RAccount Height = Layer Height
IF the Transaction is not coin base transaction
List of Enough Addresses = get enough addresses from the LSL layer
Current Value = 0
FOR EACH Address in the List of Enough Addresses
Create New *SAccount* with the same Address
IF the Address is not the last in the List
Current Value = Current Value + Address Value
SAccount Value = Address Value
ELSE IF the Address is the last in the List
SAccount Value = Output Value – Current Value
END IF

```
                    END FOR EACH
                ELSE IF the Transaction is a coin base transaction
                    Create New SAccount without Address
                    SAccount Value = Output Value
                END IF
                IF DSL Layer contains the RAccount
                    Add the new SAccount to the RAccount
                ELSE IF Layer does not contain the RAccount
                    Add the RAccount to the DSL Layer
                END IF
            Update LSL Layer
            Update User Accounts in the List of Users
            END IF
        END FOR EACH
END FOR EACH
Calculate the hash number of the LSL layer
Save the DSL Layer, LSL Layer and List of Users as files
END
```

Algorithm 4. LSL layer updating

```
Input: LSL Layer, DSL Layer RAccount
Variable:
    RAccount: Receiver Account to be added in DSL Layer
    SAccount: Sender Account that sends funds to the RAccount
    Value: Total amount transferred from SAccounts to RAccount
Output: Updated LSL Layer
BEGIN
Value = 0
FOR EACH SAccount in RAccount
    Value = Value + SAccount Value
    IF SAccount Address is not Empty
        LSL Layer Account = Get LSL Layer Account with the same SAccount Address
        LSL Layer Account Value = LSL Layer Account Value – SAccount Value
        IF LSL Layer Account Value = 0
            Remove LSL Layer Account from the LSL Layer
        END IF
    END IF
END FOR EACH
LSL Layer Account = Get LSL Layer Account with the same RAccount Address
IF there is LSL Layer Account
    LSL Layer Account Value = LSL Layer Account Value + Value
ELSE IF there is no LSL Layer Account
    Create a new LSL Layer Account with the same address, script and value of the RAccount
    Add it to the LSL Layer
END IF
END
```

Algorithm 5. User accounts' updating

```
Input: Sender, List of Users, RAccount
Variables:
    RAccount: Receiver Account to be added in DSL Layer
    SAccount: Sender Account that sends funds to the RAccount
    Value: Total amount transferred from SAccounts to RAccount
    User: User account including addresses list and the balance for each address
Output: Updated List of Users
BEGIN
Value = 0
FOR EACH SAccount in RAccount
    Value = Value + SAccount Value
    IF SAccount Address is not Empty
        Sender Account Value = Sender Account Value – Value
    END IF
```

END FOR EACH
User = Get *User* from the List of Users whose addresses list contains *RAccount* Address
User Account Value = *User* Account Value + *Value*
END

5.2 Experimental Results

5.2.1 Block *vs.* DSL Layer

After testing, we found that user accounts from blocks and layers are identical. At the same time, the DSL layer size is about 51.2 % of the block size and the complete Stateful Layered Chain size is 50.6 % of the complete blockchain size. Figure 13 shows the size of those 100 blocks and the size of the corresponding 100 DSL layers. Figure 14 shows the size of the complete Stateful Layered Chain compared to the blockchain size.

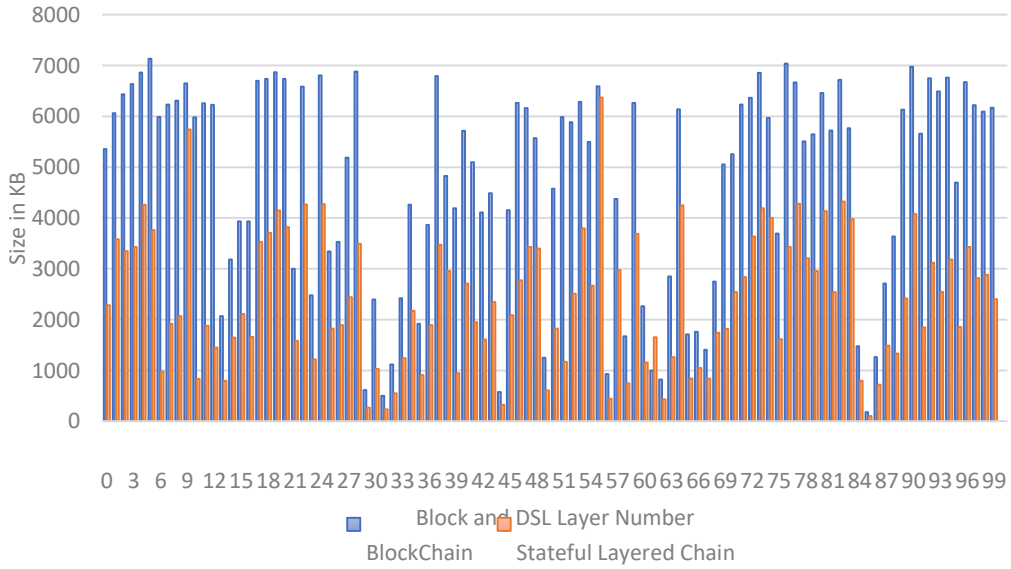

Figure 13. Size of block *vs.* size of DSL layer.

5.2.2 UTXO Pool *vs.* LSL Layer

UTXO Pool is not enough used in the Stateful Layered Chain Model. Instead, there is an LSL layer to present the accounts' balances. Figure 15 shows the UTXO pool size *versus* the LSL layer size after the last block and DSL layer. We observed that the LSL size is 53.2 % of the UTXO pool size.

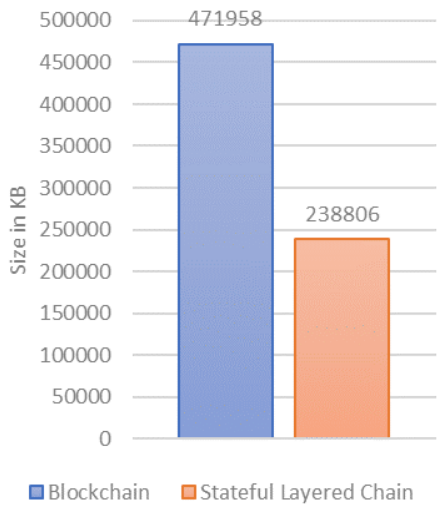

Figure 14. Stateful layered chain size *vs.* blockchain size.

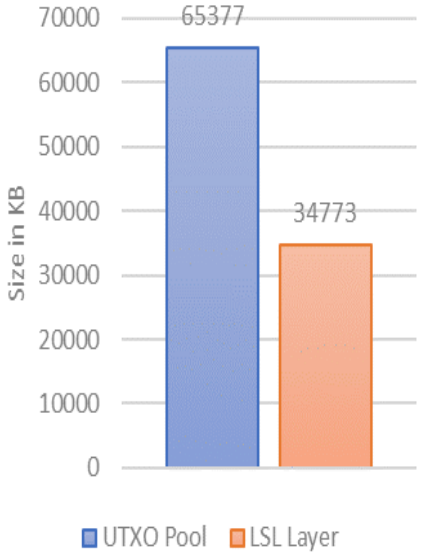

Figure 15. UTXO pool size *vs.* LSL layer size.

5.2.3 Transaction Throughput

According to the concept of layers reduction that is provided by the proposed model, the earliest DSL layers can be removed and only the LSL layer is preserved. So, the size of the DSL layer can be the same as the block size to increase the transaction throughput. Figure 16 shows that transaction throughput is increased if we keep the DSL layer size equal to the block size.

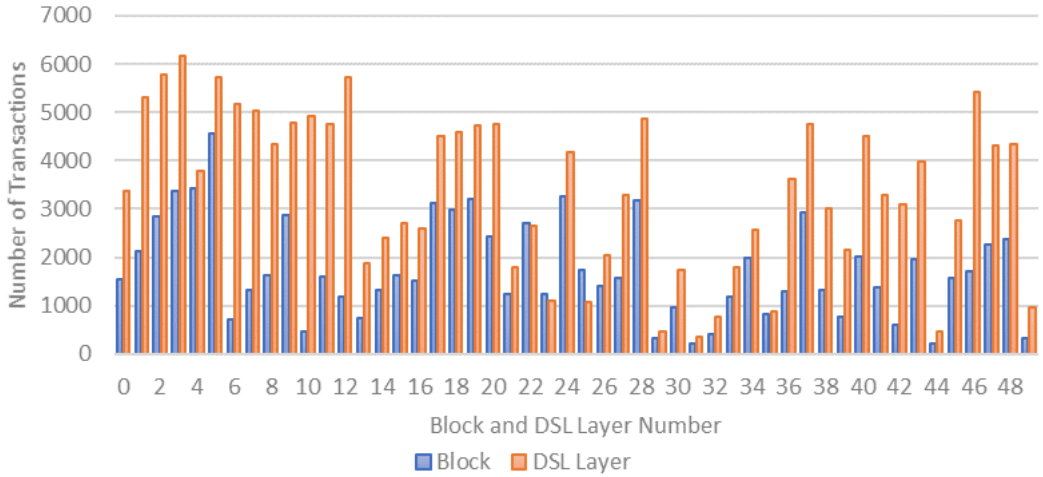

Figure 16. Number of transactions in blocks and DSL layers.

The total number of transactions in a Stateful Layered Chain with 50 DSL layers is shown in Figure 17 compared to those in a blockchain with 50 blocks of the same size as the DSL layers. We noticed that the number of transactions in the Stateful Layered Chain is 1.93 more than those in the blockchain.

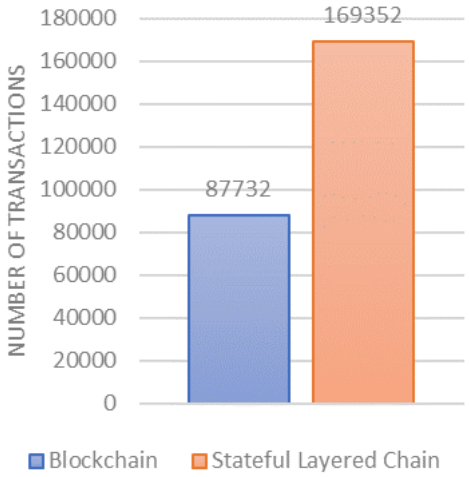

Figure 17. The total number of transactions in the stateful layered chain compared to those in the blockchain.

The results show that:

1. Using a stateful chain rather than a stateless chain causes the size of the stateful layered chain to be 50.6 % smaller than that of the Bitcoin blockchain.
2. The LSL layer size is 53.2 % smaller than the UTXO pool size, whereas the LSL layer records accounts rather than transactions.
3. The transaction throughput of a stateful layered chain can be 1.93 more than that of the Bitcoin blockchain if we keep the DSL layer size equal to the block size.

The Stateful Layered Chain Model shows success in storage scalability and transaction throughput. At the same time, the proposed model had not affected the security and decentralization. However, many of the traditional Bitcoin aspects still exist in the proposed model, like decentralization using distributed network, mining using PoW and authentication using digital signature.

6. CONCLUSION

In this paper, we proposed a Stateful Layered Chain Model which is a completely changed structure of the Bitcoin blockchain. It is no longer based on UTXOs. Instead, it relies on users' balances to send and receive funds. The proposed model enhances Bitcoin's scalability in respect of its storage. It saves approximately a half of the storage, since the Stateful Layered Chain is 50.6 % smaller than the Bitcoin blockchain. Also, the UTXO pool has been replaced with the LSL layer which is about a half the size of the UTXO pool. As the results show, the LSL size is 53.2 % smaller than the UTXO pool size. In addition, the proposed model improves the transaction throughput by nearly twice that of Bitcoin if we keep the DSL layer size equal to the block size.

One of our future research directions is to increase the transaction throughput and decrease the transaction latency by adding the parallel-mining process to the proposed model.

REFERENCES

[1] K. Salah, M. H. U. Rehman, N. Nizamuddin and A. Al-Fuqaha, "Blockchain for AI: Review and Open Research Challenges," IEEE Access, vol. 7, pp. 10127-10149, 2019.

[2] S. P. Yadav, K. K. Agrawal, B. S. Bhati, F. Al-Turjman and L. Mostarda, "Blockchain-based Cryptocurrency Regulation: An Overview," Computational Economics, vol. 59, pp. 1659–1675, 2022.

[3] "CoinMarketCap," [Online], Available: https://coinmarketcap.com/ (accessed).

[4] "Bitcoin," [Online], Available: https://bitcoin.org/en/ (accessed).

[5] F. Tschorsch and B. Scheuermann, "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies," IEEE Communications Surveys & Tutorials, vol. 18, no. 3, pp. 2084-2123, 2016.

[6] "Bitcoin Visuals," [Online], Available: https://bitcoinvisuals.com/ (accessed).

[7] S. Nakamoto, "Bitcoin: A Peer-to-peer Electronic Cash System," Decentralized Business Review, p. 21260, [Online], Available: https://bitcoin.org/bitcoin.pdf, 2008.

[8] S. Park, S. Im, Y. Seol and J. Paek, "Nodes in the Bitcoin Network: Comparative Measurement Study and Survey," IEEE Access, vol. 7, pp. 57009-57022, 2019.

[9] S. Zhao and D. O'Mahony, "Applying Blockchain Layer2 Technology to Mass E-commerce," Cryptology ePrint Archive, [Online], Available: https://eprint.iacr.org/2020/502.pdf, 2020.

[10] S.-W. Chae, J.-I. Kim and Y. Park, "Practical Time-release Blockchain," Electronics, vol. 9, no. 4, p. 672, 2020.

[11] B. Bailey and S. Sankagiri, "Merkle Trees Optimized for Stateless Clients in Bitcoin," Proc. of the Int. Conf. on Financial Cryptography and Data Security, pp. 451-466, Springer, 2021.

[12] V. Vallois and F. A. Guenane, "Bitcoin Transaction: From the Creation to Validation, a Protocol Overview," Proc. of the 2017 1st IEEE Cyber Security in Networking Conf. (CSNet), pp. 1-7, Rio de Janeiro, Brazil, 2017.

[13] N. T. Courtois, M. Grajek and R. Naik, "Optimizing sha256 in Bitcoin Mining," Proc. of the Int. Conf. on Cryptography and Security Systems, pp. 131-144, Springer, 2014.

[14] E. H. Umucu, "Elliptic Curve Cryptography in Blockchain Technology," SSRN, DOI: 10.2139/ssrn. 4033934, 2022.

[15] H. Hellani, A. E. Samhat, M. Chamoun, H. El Ghor and A. Serhrouchni, "On Blockchain Technology: Overview of Bitcoin and Future Insights," Proc. of the 2018 IEEE Int. Multidisciplinary Conf. on Engineering Technology (IMCET), pp. 1-8, Beirut, Lebanon, 2018.

[16] W. Fang, W. Chen, W. Zhang, J. Pei, W. Gao and G. Wang, "Digital Signature Scheme for Information Non-repudiation in Blockchain: A State of the Art Review," EURASIP Journal on Wireless Communications and Networking, vol. 2020, no. 1, pp. 1-15, 2020.

[17] A. I. Badev and M. Chen, "Bitcoin: Technical Background and Data Analysis," [Online], Available: https://www.federalreserve.gov/econresdata/feds/2014/files/2014104pap.pdf, 2014.

[18] P. V. Krtolica and P. S. Stanimirović, "Reverse Polish Notation Method," International Journal of Computer Mathematics, vol. 81, no. 3, pp. 273-284, 2004.

[19] H. Brakmić, Bitcoin and Lightning Network on Raspberry Pi, 1st Ed., ISBN: 978-1-4842-5522-3, Apress Berkeley, CA, pp. XIII, 364, 2019.

[20] "Bitcoindeveloper," [Online], Available: https://developer.bitcoin.org/devguide/transactions.html#.

[21] P. Müller, S. Bergsträßer, A. Rizk and R. Steinmetz, "The Bitcoin Universe: An Architectural Overview of the Bitcoin Blockchain," 11. DFN-Forum Kommunikationstechnologien, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn, 2018.

[22] G. Zyskind and O. Nathan, "Decentralizing Privacy: Using Blockchain to Protect Personal Data," Proc. of the 2015 IEEE Security and Privacy Workshops, pp. 180-184, San Jose, USA, 2015.

[23] Q. Zheng et al., "An Innovative IPFS-based Storage Model for Blockchain," Proc. of the 2018 IEEE/WIC/ACM Int. Conf. on Web Intelligence (WI), pp. 704-708, Santiago, Chile, 2018.

[24] M. S. H. Sohan et al., "Increasing Throughput and Reducing Storage Bloating Problem Using IPFS and Dual-blockchain Method," Proc. of the 2021 2nd IEEE Int. Conf. on Robotics, Electrical and Signal Processing Techniques (ICREST), pp. 732-736, DHAKA, Bangladesh, 2021.

[25] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-chain Instant Payments," [Online], Available: https://lightning.network/lightning-network-paper.pdf, 2016.

[26] X. Wang, C. Wang, K. Zhou and H. Cheng, "ESS: An Efficient Storage Scheme for Improving the Scalability of Bitcoin Network," IEEE Transactions on Network and Service Management, vol. 19, no. 2, pp. 1191-1202, 2021.

[27] R. Matzutt et al., "How to Securely Prune Bitcoin's Blockchain," Proc. of the IEEE 2020 IFIP Networking Conf. (Networking), pp. 298-306, Paris, France, 2020.

[28] M. Florian, S. Henningsen, S. Beaucamp and B. Scheuermann, "Erasing Data from Blockchain Nodes," Proc. 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 367-376, Stockholm, Sweden, 2019.

[29] X. Dai, J. Xiao, W. Yang, C. Wang and H. Jin, "Jidar: A Jigsaw-like Data Reduction Approach without Trust Assumptions for Bitcoin System," Proc. of the IEEE 39th Int. Conf. on Distributed Computing Systems (ICDCS), 2019: IEEE, pp. 1317-1326, Dallas, USA, 2019.

[30] S. Popov, "The Tangle," *White Paper*, vol. 1, no. 3, 2018.

[31] L. Quan, Q. Huang, S. Zhang and Z. Wang, "Downsampling Blockchain Algorithm," Proc. of the IEEE INFOCOM 2019 - IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS), pp. 342-347, Paris, France, 2019.

[32] R. Nagayama, R. Banno and K. Shudo, "Trail: An Architecture for Compact UTXO-based Blockchain and Smart Contract," IEICE TRANSACTIONS on Information and Systems, vol. 105, no. 2, pp. 333-343, 2022.

[33] U. Nadiya, K. Mutijarsa and C. Y. Rizqi, "Block Summarization and Compression in Bitcoin Blockchain," Proc. of the 2018 IEEE Int. Symposium on Electronics and Smart Devices (ISESD), pp. 1-4, Bandung, Indonesia, 2018.

[34] "Blockchain Data API," [Online], Available: https://www.blockchain.com/api/blockchain_api.

ملخص البحث:

أصبحت العُملة الرّقمية مجالاً للبحث في أيامنا هذه. وتعدّ تكنولوجيا سلاسل الكُتَل الأساس الّذي تقوم عليه العملة الرّقمية (بِتكوين) نظراً لتمتّعها بعدد من الخصائص، مثل اللامركزية والشّفافية والحَصانة. ومع ذلك، يمكن اعتبار تكنولوجيا سلاسل الكُتَل هي السّبب وراء المسائل المتعلّقة بتوسيع استخدام البتكويْن، وخصوصاً فيما يتعلّق بالتّخزين. وتحتاج العُقَد في شبكة البتكويْن الى تخزين كامل سلسلة الكُتَل لإجراء التّعامُلات. لكن بمرور الوقت، يصبح حجم سلسلة الكُتَل ضخماً. وهكذا فإنّ العُقَد الممتلئة ستفضّل ترْك الشبكة. وهذا يؤدي الى زيادة مركزية سلسلة الكُتَل، ويتأثر أمان الشّبكة بصورةٍ سلبية.

تبحث هذه الورقة في اقتراح نظام سلاسل ذي طبقاتٍ يستند على تخزين أرْصِدة الحسابات من أجْل التّقليل من حجْم سلسلة الكُتَل الخاصّة بالبتكويْن. ويغير النّموذج المقترح البنية التقليدية لسلسلة الكُتَل من كُتَلٍ الى طبقات. وتُظهر نتائج التّجارب أنّ النّموذج المقترح يقلّل حجم سلسلة الكُتَل بنسبة تقرُب من (50.6%)، ويتضمّن ذلك مضاعفة التّعاملات التي يمكن إجراؤها عبر النّظام.