International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Preventing the bitcoin Double Spend using Transaction Hash and Unspent Transaction Output

A. Murugan, J. Vijayalakshmi

ABSTRACT: ***The red-hot crypto currency is a bitcoin which occupies first position in the capital investment of financial world which is assaulted by various factors like wallet attacks, network attacks, mining attacks and double spending attacks. Double spending is the major attack in which the attacker tries to cheat the network nodes and use the same coin for more than one set of transactions. Of this the original transaction identification from the set of transactions is a challenging one. In this paper we propose a solution for identifying the primary transaction from the set of double spended or multi spended transactions. The proposed approach finds the authentic transaction from the list of double spended transactions using transaction hash value, which is primarily used for every transaction in the Bitcoin network. Transaction hash value is used as transaction identifier for each bitcoin transaction. By comparing the transaction hash value with the existing pool of unconfirmed input pool, transhash pool and utxopool one can identify the genuine transaction from the flawed transaction list. The firsthand transaction is then added to the Confirmed input pool which is then entered into the newly added block of the blockchain. This architecture will prevent the double spend of bitcoin further in the network which facilitates the network nodes as well as minimize the miners task for verification and validation of transaction.***

Keywords: Blockchain, Double spending, UTXOpool, Inputpool, Transactionhash

I. INTRODUCTION

In 2008 the evolution of global financial crisis made the advent of Digital currency management systems. Satoshi Nakamoto was the the father for the arrival of crypto currency in the digital world. Crypto currency is a form of virtual currency which supports high level of security and prevents the reproduction of using same digital currency for multiple times in transactions. Crypto currencies follow the principle of permanent ledger management, globally distributed and constant verification by powered computers [2].

Manuscript published on 30 September 2019
* Correspondence Author
Dr. A. Murugan*, Associate Professor and Head of the Department of PG and Research in Computer Science.
J. Vijayalakshmi, research scholar of PG & Research Department of Computer Science in Dr. Ambedkar Govt. Arts College. She obtained her Master of Philosophy(M.Phil) degree in Computer Science at Bharathidasan college, Tiruchirappalli

Crypto currency is a decentralized currency which provides open and self-regulating features and support transaction which is alternative to banks [3]. Crypto currency is analogous to fiat currency which is widely accepted by various countries for merchandise. The first crypto currency was Bitcoin which was introduced in 2009 with various security features compared to centralized computing system like banks and financial institutions. Bitcoin is a "Gold standard" crypto currency which is widely accepted by all countries around the world like United States, Canada, Australia and European Union like Finland, Cyprus, UK, Germany and Bulgaria [2]. Bitcoin is a type of virtual currency with none intermediary trusted parties and none participant identities. The construction of Bitcoin system includes "proof-of-work" amaze and "public ledger "called blockchain which encounter double spending occurrence effectively. A Blockchain is a shared public ledger which holds all bitcoin transactions which is shared among the network peers. The main segments of Bitcoin system include transactions, consensus and communication. Bitcoin state is identified based on the list of transactions [4]. Transaction is a way of transferring of bitcoin (BTCs) among peers in Bitcoin network through electronic mode. These peers are referred it as bitcoin addresses which are used in each transaction occurrence [5]. The basic part of transaction includes Unspent Transaction Output (UTXO) which is the fundamental building block of crypto currency transaction. The transactions which consume UTXO are called transaction inputs and the transaction which produce UTXO are called transaction outputs. The transactions are valid once it gets accepted into the public history of transactions called blockchain or public ledger [3]. The following Figure 1 shows the regular transaction process in which Alpha is the sender and Bravo is the receiver and TH refers to transaction hash or transaction identifier (TXID) which is used as primary identifier for every transaction. If the TH value is used for spending numerous times then it is called it as *double spending transaction* or *multi spending transaction*. Normally the transaction produces two outputs the one that sent to recipient are called spent output and the one that sent to the sender itself as balance amount are called change output. Here Alpha and Bravo are peers.

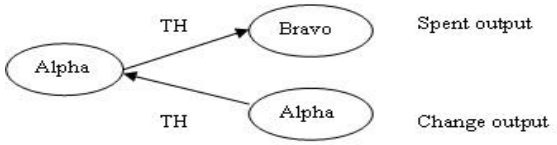

Figure 1: Normal transaction process

Peers contain multiple addresses which are stored and managed by its wallet along with its public/private key pairs. These key pairs are used for transferring and verifying ownership of BTCs among addresses. These public/private keys are used for generating and verifying the authenticity of the ownership by checking the chain of signatures.

The verified transactions are admitted inside the blocks which are broadcasted to the entire network of nodes. To prevent bitcoin from double spending of the same BTC for more than once, hash-based proof-of-work (POW) strategy is used [5]. Double spending or irreversible transaction is a process by which the user could generate two or more transactions with the same set of digital coins to two or more different receivers in a rapid sequence. This type of attack is often called it as race attack. If all the recipient transaction was successfully verified and included in the local blockchain then it might create double spending problem which leads the blockchain to conflicting state. The double spend attack arises due to various factors like honest miners, vendors, client, network propagation delay and bitcoin exchange services. The following attacks like race attack, Finney attack, brute force attack, vector 76 attacks, gold finger attack, selfish mining, block withholding and fork after withholding attacks also facilitate double spending attacks [6]. In 51% attack the attacker gets 51% of the computational power over the network control which may use to reverse the original transaction and make his private blockchain as real blockchain. In race attack the attacker sends the same coin in rapid succession to two different addresses. This attack happens if the seller doesn't wait for the confirmations of payment. This happens in fast payment scenarios like fast food payments, online services, vending machine payments and ATM withdrawals where the exchange time is less between transfer of currency and products [5]. The following Figure 2 shows the emanate of double spending problem due to race attack. Here the sender Alpha tries to use the same transaction hash (TH) which spends same bitcoin amount to Bravo and Charlie at the rapid succession. Based on the current blockchain rule, the miners accept the longer blockchain as current chain for further processing and the original recipient Bravo's transaction is rejected due to shorter blockchain (local blockchain 1) and Charlie's transaction is accepted due to longer confirmation chain (local blockchain 2) which leads the blockchain to inconsistent state.

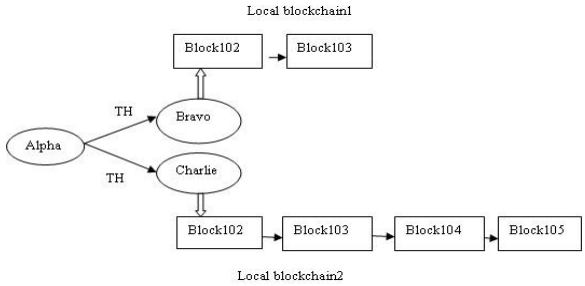

Figure 2: Double spend attack

To avoid this type of erratic status bitcoin uses the conception of POW and consensus agreement. If double spending situation arise then the sellers will lose their products and income. To forbid this action either the blockchain management can use observers in the network or ban the incoming connections [6].

II. EFFECT OF DOUBLE SPENDING

Once the miner grasps the 51% of computing power, one can take the control of the blockchain and make the double spending attack easily by altering the transaction data, stop verifying the transaction inside the block and stop the miner mining any available block. The fork problems arise due to arrival of consensus rules. Consensus is a mechanism in which all blockchain nodes have to agree to the same message and assure that the latest block have been added to the blockchain correctly without any fork attacks or malicious attacks. Fork problem arises due to the upgradation of software from old version to newer version in decentralized network [7]. The race attack like double spending the same crypto currency multiple times for transactions was created by fork, which is relatively easy in POW based blockchain. Here the attacker achieved the intermediate time between transaction initiation and transaction confirmation for quickly launching the double spending attack [8]. The double spending attack creates various issues like governance issue and accountability issue. The governance issue degrades the digital currency's value relative to general price level and the accountability issues may create loop hole for the attackers to attack digital information [9].

III.REVIEW OF LITERATURE

Satoshi [1] propound the solution for double spending of bitcoin by using timestamp server, proof-of-work and through consensus protocol. The timestamp proves the existence of transaction to achieve hash which is used for forming a chain.Proof of work involves auditing the hashing value with the number of zero bits. This computation was impractical for an attacker to attack when honest mining nodes are controlling the network. Consensus mechanism was used to enforce rules and incentives for the network nodes.

Mandar et al. [10] proposed a solution for double spending using timeout period of transaction. During the transaction time period no other transaction is allowed to access particular database until the initiated transaction was completed. If user attempts to create double spend of same bitcoin by initiating a transaction, the transaction would be blocked. The author also restricts the user to login into his already logged in account and also prevent him to make another transaction for next three times.

Martijin [11] provided the solution for 51% attack for reducing the conflicting honor for miners. The author used two phase proof-of-work model for solving this issue by preserving the existing blockchain with surviving bitcoin balances. The system also preserves the large investments made by the miners and also provides transition from working system to new one with adjustable knobs which can be suitable for community needs.

IV. PROPOSED WORK

The present blockchain follows the standard of adding all incoming transactions to the utxopool which is then validated and added to the existing blockchain routine, once it gets verified and confirmed by group of participants in the bitcoin network.

The proposed architecture uses inputpool for adding transaction to the block in which all the incoming transactions were added to the Unconfirmed input pool. Once it was verified and validated without double spending attack it got added to the confirmed inputpool which was then added to the block of the blockchain. To prevent degeneration of double spend attack in the block chain the present identified the double spended transaction from the Unconfirmed input pool of transactions using the architecture of *Dual payout based on Lost Agreement Amount* (DPL2A) which was depicted on Figure 3. DPL2A architecture identified the double spended transactions which were taken as input for the subsequent process for identifying the original transaction. To diagnose the original transaction from the double spended transaction, this paper proposed a suitable solution based on prearranged *transhash* table, input pool table reference and predetermined *utxopool* table reference which was depicted on Figure 4.

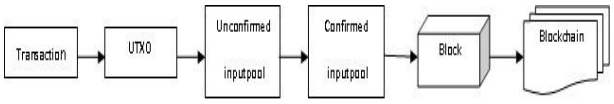

Figure 3: Dual Payout based on Lost Agreement Amount (DPL2A) Architecture

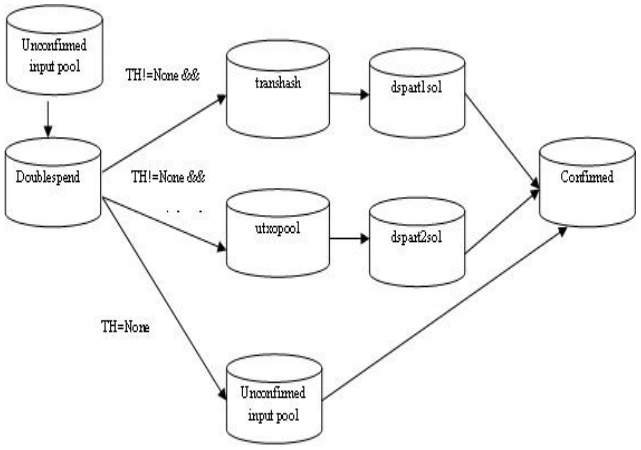

Figure 4: Authentic Contract Recognition based on Trans_UTXO_Input(ACRT) architecture

The *Dual payout based on Lost Agreement Amount* (DPL2A) architecture includes various pools (databases) for identifying the double spend transactions from the *unconfirmed inputpool* and the current *Authentic Contract Recognition based on Trans_UTXO_Input* (ACRT) architecture was used for identifying the original transaction from the set of double spended transaction which was added to the *confirmed inputpool* which in turn added to the block of the blockchain. The proposed work was implemented and tested based on four transactors namely *Alpha, Bravo, Charlie* and *Delta* who played the main role of the experimental transactions process. The framework consists of various pools which include *transaction, block, blockchain, inputpool, outputpool, utxopool, alphastxopool, bravostxopool, charliestxopool, deltastxopool, alphautxopool, bravoutxopool, charlieutxopool, deltautxopool, alphapool, bravopool, charliepool, deltapool, doublespend, confirmedinput, transhash, dspart1sol and dspart2sol*. All these pools follow the corresponding structure which was depicted in following Table 1.

Table 1: Common structure followed in DPL2A and ACRT architecture

transaction

sender	recipient	inputs	outputs	hash	value	time	size

block

block height	transactions	transaction counter	previous hash	timestamp	blockhash	nonce	merkle

blockchain

blockchain	utxopool	pool	stxopool

inputpool,outputpool,utxopool

sender name	transactionhash	output index	value	time	used flag	used counter	recipient name

alphastxopool,bravostxopool,charliestxopool,deltastxopool

sender name	transactionhash	output index	value	time	used flag	used counter	recipient name

alphautxopool, bravoutxopool, charlieutxopool, deltautxopool

sender name	transactionhash	output index	value	time	used flag	used counter	recipient name

alphapool, bravopool, charliepool, deltapool

sender name	transactionhash	output index	value	time	used flag	used counter	recipient name

doublespend

sender name	transactionhash	output index	value	time	used flag	used counter	recipient name

confirmedinput

sender name	transactionhash	output index	value	time	used flag	used counter	recipient name

transhash

sender	recipient	value	time	encode	serialise

dspart1sol

ds sender	dstrans hash	ds output index	ds value	ds time	ds ufla g	ds ucounter	ds recipient	ts sender	ts recipient	ts value	ts time	ts serialise	ts encode

dspart2sol

ds sender	dstrans hash	dsoutputindex	ds value	ds time	ds ufla g	ds ucounter	ds recipient	utxo sender	utxo transhash	utxo ioutputindex	utxo value

The *transaction* table specifies the transaction properties like time of transaction occurrence, transferring amount in value, size of transaction, sender of transaction, recipient of transaction, set of inputs, set of outputs and transaction hash. The transaction hash was determined based on applying SHA256 of the encoded transaction data which includes timestamp, value, size, sender, recipient and inputs. This transaction hash is stored in *transhash* table along with the values used for calculating transaction hash which includes sender of the transaction, receiver of the transaction, transferred value, transaction time, the encoded transaction value and serialise value which stores the transaction hash of a particular transaction. The block table holds the block height or index of the transaction, set of all transactions, their counter value, previous transactionhash reference value, timestamp of current transaction, block hash value which is determined similar to transactionhash, then nonce value which is used by miners and merkle value which is used to identify the particular block.

The *blockchain* holds the set of all blocks that exist in the current blockchain starting from gensisblock (block 0) to current block that is to be added. This holds common pools like *utxopool*, *pool* and *stxopool* for every transactor. Each transactor holds separate blockchain. The variations of blockchain in every transactor are solved by miners who are involved in verifying and validating the block of transactions before it is added to the blockchain. The *inputpool* holds the set of all input transactions before it is validated and added to the block. The *outputpool* holds the set of all output transactions generated based on input transactions. The *utxopool* holds the list of all unspent transactions. The *stxopool* holds the set of all spent transactions. The *pool* holds the general list of all transactions for every transactor that is both spent and unspent transactions.

The proposed Dual payout based on Lost Agreement Amount (DPL2A) identifies the set of all double spended or multiple spended transactions which uses same transaction hash and same amount for more than one set of recipients by the same sender. This architecture identifies this double spended transactions based on comparing *inputpool* with individual transactors *utxopool* and *stxopool* values. The resultant double spended transactions are stored in *doublespend* table. The proposed architecture now identifies the original transactions out of multiple spended transactions with different recipients. For example, consider the case

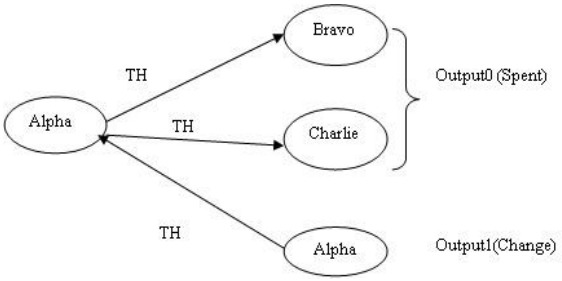

Figure 5: Double spending problem

In the Figure 5 the difficulty of double spending is portrayed, where Alpha is the sender Bravo, Charlie and Alpha were recipients. Output0 indicates the original recipient of transaction and Output1 indicates the change value sent to the sender. All these recipients share the same transaction hash value (TH). The only difference is there is a change in outputindex and the value sent to the recipients. Assume if Alpha is holding initially 100 bitcoins then Bravo and Charlie will receive 30 with output index as 0 and Alpha will receive balance amount as 70 with output index as 1. Consider the below transactions.

transaction1:
sendername: <__main__.Blockchain instance at 0xb63099cc>
transactionhash: b4418f0b8d4ec73ec3aa2921ab72faccaad0d509f14c0f55019fda5599e6e250
outputindex: 0
value: 30.0
time1554706418.77
usedflag: 1
usedcounter: 6
recipientname: <__main__.Blockchain instance at 0xb6309a0c>
transaction2:
sendername: <__main__.Blockchain instance at 0xb63099cc>
transactionhash: b4418f0b8d4ec73ec3aa2921ab72faccaad0d509f14c0f55019fda5599e6e250
outputindex: 0
value: 30.0
time1554706418.77
usedflag: 1
usedcounter: 6
recipientname: None
transaction3:
sendername: <__main__.Blockchain instance at 0xb63099cc>
transactionhash: b4418f0b8d4ec73ec3aa2921ab72faccaad0d509f14c0f55019fda5599e6e250
outputindex: 0
value: 30.0
time1554706418.77
usedflag: 1
usedcounter: 6

recipientname: <__main__.Blockchain instance at 0xb630972c>

Here the transaction1, transaction2 and transaction3 have identical sendername (0xb63099cc), identical transactionhash (b4418...6e250), identical outputindex (0), identical time (1554706418.77) and identical value (30.0) except the variation in recipient names (0xb6309a0c, None, 0xb630972c). These types of transactions are called it as double spended transactions or multiple spended transactions. In this case in order to identify the original transaction out of these recipients we are using a technique based on the values of transhash, unconfirmed inputpool and utxopool tables. Here we are assuming whenever the sender Alpha sent amount to recipient, they initially set the recipient field as None. Whenever the recipient uses this input for other transaction it retains the same value as recipient. In case if hacker inserts his name for recipient field, he will not receive the amount because those who are having none as recipient only can able to send as well as receive. Based on that principle we are identifying the original transaction recipient and sender based on comparing the values from *utxopool*, *unconfirmed inputpool* and *transhash* table values.

The following algorithm soldoublespend () identifies the original transaction among the list of double spended transactions. Based on transhash value the double spended transactions are compared with different pools like unconfirmed inputpool, transhash table and utxopool tables. If transhash is "None" it was compared with unconfirmed inputpool. If transhash is not "None" and output index is 0 then *doublespend* and *transhash* table is compared by every row to find the matching transaction hash reference if it founded then we store the set of rows to *dspart1sol* table. If transhash is not "None" and output index is 1 then compare *doublespend* and *utxopool* table for finding the matching rows which is then stored into *dspart2sol* table. Then the checktranshash () algorithm to find out the original transaction from the double spended transactions. Finally, the original transaction is identified and stores the result into *confirmedinput* pool which would be used for adding the transaction to the block.

Algorithm soldoublespend()

```
// identifying the original transaction among the double spended list of transactions
{
    //compare the table values of doublespend and transhash for checking whether
    // transactionhash !=None and outputindex=0 then store the matching elements
    // in dspart1sol table
    for every row in doublespend
    {
        for every row in transhash
        {
            if match exists
        read row
                    if row = None
                        break
                    insert row into dspart1sol table
        }
    }
    //compare the table values of doublespend and utxopool for checking whether
    // transactionhash !=None and outputindex=1 then store the matching elements
    // in dspart2sol table
    for every row in doublespend
    {
        for every row in utxopool
        {
            if match exists
             read row
                    if row = None
                        break
                     insert row into dspart2sol table
        }
    }

    // call the checktranshash() if transactionhash value is None
    checktranshash()
}
```

Algorithm checktranshash()

```
// identifying the transactions whose transactionhash = None which was spended multiple times
{
    // retrieve all rows in the inputpool table whose transactionhash and recipientname is None
     for every row in inputpool
     {
        if match exists
           read row
           if row = None
                break
           write each row in confirmedinput table
        }
    }

    // checking the columns inside dspart1sol table whose matching results are stored
    // confirmed inputpool table
     for every row in dspart1sol
     {
        if match exists
            read row
            if row = None
                break
            write each row in confirmedinput table
        }
    }
    // checking the columns inside dspart2sol table whose matching results are stored
    // confirmed inputpool table

     for every row in dspart2sol
     {
        if match exists
           read row
            if row = None
```

```
break
            write each row in confirmedinput table
        }
    }

}
```

A. RESULTS AND DISCUSSION

The proposed work was implemented on Intel i3 Core processor with 2GB RAM at Ubuntu 32-bit operating system and Python is used as frontend and Postgresql database as backend server. The work was successfully carried out based on 54 and 108 sets of transaction data and the following results which were depicted in Table 2.

Table 2: Detecting the original transaction from double spended pool

PROCESS TYPE	TRANSACTION VALUES		
Input pool	54	108	156
Doublespend pool	24	46	59
Confirmedinputpool	15	28	37

From the table 2, it is concluded that for the set of 54 input transactions 24 double spended data found of this original 15 transaction was found and it was added to the Confirmed inputpool. Equivalently the test for the set of 108 input transactions of that 46 double spended data was found and from that the proposed system identified the original 28 transactions which were added to the Confirmed input pool. From the set of 156 transactions, 59 double spended data was identified of these 37 original authentic transactions was identified using proposed architecture.

V.CONCLUSION AND FUTURE SCOPE

This paper discussed about the double spending problem and its effect of double spending in various factors like affecting governance and accountability issues. It also describes the methodology handled by different authors for avoiding double spend occurrence. This paper proposed the new architecture for preventing the double spend by identifying **"*Authentic Contract Recognition based on Trans_UTXO_Input pool* (ACRT)"** architecture for identifying the original bitcoin transaction from the set of faulty transactions. The present work used the architecture of existing bitcoin with few changes in order to mitigate the double spend attack. This method will identify the primary transaction correctly and it gets added to the blockchain. With the implementation of this method one can avoid the double spend attack occurrence in the anticipated blocks. This attack will eliminate most of the double spending attacks including race attack, vector attack, and brute force attack. The current work focuses on including the genesis (original) transaction into the block before it gets confirmed by the miners. The future work focuses on preventing the double spending attack in the existing block of blockchain.

REFERENCES

1. Nakamoto Satoshi, "Bitcoin: A Peer-to-Peer Electronic Cash System", Journal of Cryptology, Springer, 2008:28.
2. Vijayalakshmi, J., and A. Murugan. "Crypto Coin Overview of Basic Transaction." *International Journal of Applied Research on Information Technology and Computing* 8, no. 2 (2017): 113-120.
3. Ouaddah, Aafaf, Anas Abou Elkalam, and Abdellah Ait Ouahman. "Towards a novel privacy-preserving access control model based on blockchain technology in IoT." In *Europe and MENA Cooperation Advances in Information and Communication Technologies*, pp. 523-533. Springer, Cham, 2017.
4. Bonneau, Joseph, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies." In *2015 IEEE Symposium on Security and Privacy*, pp. 104-121. IEEE, 2015.
5. Karame, Ghassan, Elli Androulaki, and Srdjan Capkun. "Two Bitcoins at the Price of One? Double-Spending Attacks on Fast Payments in Bitcoin." *IACR Cryptology ePrint Archive*2012, no. 248 (2012).
6. Conti, Mauro, E. Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. "A survey on security and privacy issues of bitcoin." *IEEE Communications Surveys & Tutorials* 20, no. 4 (2018): 3416-3452.
7. Lin, Iuon-Chang, and Tzu-Chun Liao. "A Survey of Blockchain Security Issues and Challenges." *IJ Network Security* 19, no. 5 (2017): 653-659.
8. Li, Xiaoqi, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. "A survey on the security of blockchain systems." *Future Generation Computer Systems* (2017).
9. Chohan, Usman W. "The Double Spending Problem and Cryptocurrencies." *Available at SSRN 3090174* (2017).
10. Kadam, Mandar, Praharsh Jha, and Shravan Jaiswal. "Double spending prevention in bitcoins network." *International Journal of Computer Engineering and Applications* 9, no. VIII (2015).
11. Bastiaan, Martijn. "Preventing the 51%-attack: a stochastic analysis of two-phase proof of work in bitcoin."In *Available at http://referaat.cs.utwente.nl/conference/22/paper/7473/preventingthe-51-attack-astochastic-analysis-of-two-phase-proof-of-work-in-bitcoin. pdf*. 2015.
12. Murugan, "Discovering the bitcoin Double Spend Using Lost Agreement Amount" *International Journal of Recent Technology and Engineering3, no*. VIII(2019)

AUTHORS PROFILE

Dr. A. Murugan is an Associate Professor and Head of the Department of PG and Research in Computer Science. His research interests focus on Molecular Computation, Graph Theory, Data Structure, Analysis of Algorithms, Theoretical Computer Science and Cryptocurrency Security. He completed Ph.D in Computer Science, Master of Science(M.Sc) in computer science. He published 3 books in computer science subjects. His publication focuses on DNA Computing, Cloud Computing, Mobile Communication and Security aspects of all networking. He published more than 90 papers in both International and National Journals and springer proceedings. He has produced 8 Ph. D research scholars. He is an Editorial Board Member, Technical Advisory and Reviewer in various conferences and journals. He acted as a Board of Studies member in various reputed colleges of Chennai. He plays a variety role in both colleges and university.

J. Vijayalakshmi is a full-time research scholar of PG & Research Department of Computer Science in Dr. Ambedkar Govt. Arts College. She obtained her Master of Philosophy(M.Phil) degree in Computer Science at Bharathidasan college, Tiruchirappalli in the year 2006 and obtained her Master of Computer Application(M.C.A) under University of Madras in the year 2004 and she also completed her Bachelor of Science(B.Sc) in Computer science under University of Madras in the year 2001. She had working experience of 9 years in private colleges. She has published her papers in international journal and also in springer proceedings. Her interest is focus on providing security in virtual currency management.