# coinBD: An Enhanced Version of Proof of Work With Less Computational Power

**SUBMISSION DATE: 17.12.18**

Mohammad Badrul Hossain-18341006
Sadman Sakib Akash-18341007
Md. Asraful Haque-13101245
Department of Computer Science and Engineering

**Supervisor:**

**Dr. Mahbub Alam Majumdar**
Professor
Department of Computer Science and Engineering

## Declaration

It is hereby declared that this thesis /project report or any part of it has not been submitted elsewhere for the award of any Degree or Diploma.

**Authors:**

Author Name 1:

MOHAMMAD BADRUL HOSSAIN

Student ID: 18341006

Author Name 2:

Sadman Sakib Akash

Student ID: 18341007

Author Name 3:
MD. ASRAFUL HAQUE
Student ID: 13101245

**Supervisor:**

**Supervisor's Name:**
Dr. Mahbub Alam Majumdar
Professor,
Department of Computer Science and Engineering,
BRAC University.

## Acknowledgement

Thanks to our supervisor Dr. Mahbub Alam Majumdar for giving us this opportunity to research on this topic and assisting us through the process. We would like to express our special gratitude and appreciation to him for giving us such attention and time. We are also thankful to Tarik Adnan Moon and Mr. Nadim who are blockchain experts for their valuable suggestions.

## Abstract

Cryptocurrencies are digital currency system where transactions happen in a decentralized peer to peer network without and security provided by cryptography. Blockchain is the backbone technology behind cryptocurrencies which plays the role of distributed ledger. Bitcoin is the firstly invented cryptocurrency of the world and till present day it is the most popular in this sector. Bitcoin is maintained by very organized blockchain network which has secured validation systems for transactions, stable and simultaneous consensus and mining process. But our concern was at its proof of work solving procedure which is used at mining phase. Although proof of work is highly secured but it needs too much computational power and network latency to solve it. It also needs time to mine as if difficulty rises proof of work solving takes longer time. To solve the problem, we implemented a cryptocurrency named coinBD where we used our own approach proof of rigor (por) instead of proof of work which directly works with transaction amount. Proof of rigor does not take much time and high computation as proof of work for mining. Our expectation is to establish coinBD's proof of rigor as a solution to proof of works those complexities.

## Keywords

Bitcoin, Blockchain, Block, Chain, Cryptograhy, Cryptographic, Cryptocurrency, Consensus, Algorithm, Hash Value, Hash Function, SHA256, SHA512, ECDSA, Proof of Work, Proof of Rigor, Merkle Root Tree, Merkle Tree, Merkle Root.

# TABLE OF CONTENTS

# LIST OF FIGURES

# Chapter 1

# Introduction

In this unique moment of time we are experiencing many new technologies among them cryptocurrency is on the lead. Among many online cryptocurrencies bitcoin is the most stable and well heard by everybody. Cryptocurrency is solely based on cryptographic puzzle and with the help of cryptography the need of a third party for instance bank, office is replaceable. Proof of work was introduced to secure the system which need a lot of computational power. Miners are the part of this system who do this proof work and by do so they solve a cryptographic puzzle and achieve the right to add the block to blockchain. To solve a proof of work, a miner needs high configured hardware and a minimum amount of time. Computing this much cryptographic puzzles is very time and energy consuming. In this function, miner has to generate sha value which should meet the target by generating a nonce. Daian1, Eyal1, Juels2, and Sirer1 in their piecework has developed a new proof of work which is called piecework which depends on a prime number [19]. From the permacoin [18], they developed a two-step small proof of work. We have developed a prototype of bitcoin and are trying to minimize the time and energy consumption. For this reason, we developed our own small cryptographic puzzle which is Proof of rigor. In Proof of rigor, the miner will be guessing the transaction amount by iterating a loop match with a specific value. The needed iteration will work as a proof of work. In simple term, our main goal is to minimize the computational cost. To do so we give the miner not so hard work like generating a nonce value or not just mitigate the difficulty so that they can get to the target easily. We developed an easy way so that miner can mine a block very fast and can do so in much less computation. For this reason, the risk of happening fork will be reduced dramatically. As our system is fast so we will not be giving a constant reward for every block that is mined. We are going to provide the mining reward depending on iterative value which will not only determine the mining reward but also the user’s fees. So in this paper we are going to introduce a new simple proof of work.

# Chapter 2

# Literature Review

Cryptocurrency is a new thing to accept for our day to day life. It is a totally new thing and a new technology. One of the important and main property of cryptocurrency is Proof of Work. To find out what actually is proof of work we first started reading the basic of proof of work which is Hashcash. Back (2002) [3] in his research paper said about a CPU cost-function MINT(). If the MINT() of a cryptocurrency system can be controlled we can reduce the execution time. MINT() is used for creating tokens to participate in a protocol with server. A cost-function should be efficiently verifiable, but parameterisably expensive to compute. The server will check the value of the token using an evaluation function VALUE(), and only proceed with the protocol if the token has the required value. If we take Bitcoin as a model of Cryptocurrency, then it can be said that Bitcoin's proof of work is just fine on its own. There is a slide chance of improvement but we can skip it for now. If we don't improve the efficiency of proof of work then we need to look at the bigger picture, upgrading the hashing algorithm. Secure Hash Algorithm is the main zest that converts the public key of owner and previous transaction history into a hash value through cryptographic function. Wherever hash value or identity protection was needed most of them were using

SHA-1(SHA-256). But now upgraded and more secure hash function has come forward. SHA 384 and SHA 512 is in every way more efficient than SHA 256. After covering several paper, we came to know that most of them are on hardware level. Emam and Emami (2007) [6] in their research paper they implemented SHA 512 with FPGA on a hardware level. They said that they integrate two SHA512 module and they successfully can take double amount input can produce double output at the same time of an individual SHA512. We are thinking of using it for our thesis purposes which can certainly increase the efficiency of execution and throughput In other studies we found that McEvoy (2006) [10] and others in their research paper found that SHA-256 is a function with up to 264 bits of input which are broken into 512-bit blocks and with a 256-bit output. It is conjectured to be collision-, preimage-, and second-preimage resistant at (resp.) 128-, 256-, and 256-bit security levels, but it has the unfortunate property that given SHA-256(m) but not m, it is relatively easy to compute SHA-256(m|p|m') for an arbitrary suffix m', where p depends only on the length of m. so we are also thinking of shortening the length of m. There is a huge chance of improvement for Blockchain. The blockchain data structure is an ordered back-linked list of blocks of transactions. The blockchain can be stored as a flat file, or in a simple database. The bitcoin core client stores the blockchain metadata using Google's LevelDB database. Blocks are linked "back", each referring to the previous block in the chain. If we going back through blocks with previous block then the first block is called the Genesis block. By studying research papers like we came across a great idea from

Kosba, Miller, Shi, Wen, Papamanthou's (2016) [8] Hawk. They managed to analyse bitcoin's blockchain and in his paper he used generic blockchain. They said that they adopt a generic blockchain model where the blockchain can run arbitrary Turing-complete programs. In comparison, previous and concurrent works [4], [1], [16], [9] retrofit the artifacts of Bitcoin's limited and hard-to-use scripting language. Miller and LaViola (2014) [11] in their paper stated about blockchain that the block hash is not actually included inside the block's data structure, neither when the block is transmitted on the network, nor when it is stored on a node's persistence storage as part of the blockchain. Instead, the block's hash is computed by each node as the block is received from the network.

## 2.1 Blockchain

Blockchain is a data structure constructed by ordered back linked-list containing blocks or list of transactions [2]. Each blocks in blockchain is connected with its previous block where the previous block is considered as parent block of newly generated block. A hash value generated by SHA256 which is a cryptographic hash algorithm is used as unique identifier of each block. Reference of parent block is saved by blocks by containing the hash value of previous block. Therefore, blocks can iterate backward to their parent blocks following the sequence of previous hash values. By traversing in this way a chain gets created which backtracks to the first created block which is known as genesis block. However, blockchain is a secured peer to peer network system which is decentralized and broadcasted to each peer or node on the network. Cryptrographic hash functions used in generating block identity has transformed the whole blockchain into a highly secured system. As every block contains a field of previous block hash therefore if somehow the hash value of parent block gets changed then the identity hash of next or child block will also be get changed. The theme is any change in parent node hash value will affect the identifier of next block. In depth if any modification occurs in parent block its identifier hash value gets changed. As a result, previous block hash field of child block automatically gets replaced by newly generated parent hash value. Lastly as a part of the block verification process hash value of child block also gets changed and a new hash value gets assigned as its identifier. The process continues through each connected child blocks until the last block occurs. To complete this whole process, it is necessary to have huge hardware resource and computational power as it needs a vast range of recalculation operation and as blockchain grows dynamically faster it is nearly impossible to modify or hamper its blocks which fulfils the security system of blockchain stable enough and keeps the whole blockchain immutable.

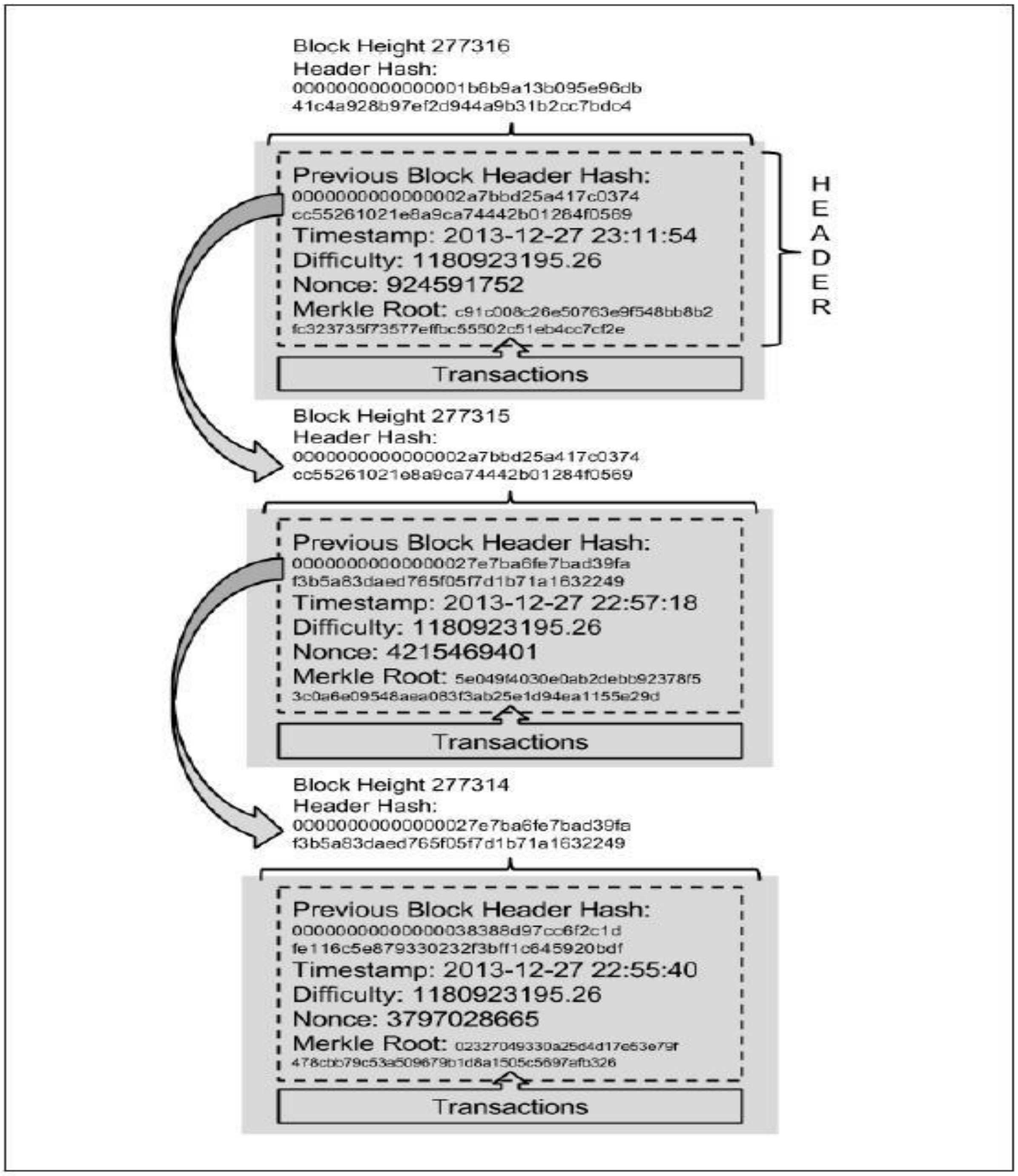

Block Height 277316
Header Hash:
00000000000000001b6b9a13b095e96db
41c4a928b97ef2d944a9b31b2cc7bdc4
Previous Block Header Hash:
00000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Timestamp: 2013-12-27 23:11:54
Difficulty: 1180923195.26
Nonce: 924591752
Merkle Root: c91c008c26e50763e9f548bb8b2
fc323735f73577effbc55502c51eb4cc7cf2e
Transactions
HEADER
Block Height 277315
Header Hash:
00000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Previous Block Header Hash:
000000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249
Timestamp: 2013-12-27 22:57:18
Difficulty: 1180923195.26
Nonce: 4215469401
Merkle Root: 5e049f4030e0ab2debb92378f5
3c0a6e09548aea083f3ab25e1d94ea1155e29d
Transactions
Block Height 277314
Header Hash:
000000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249
Previous Block Header Hash:
000000000000000038388d97cc6f2c1d
fe116c5e879330232f3bff1c645920bdf
Timestamp: 2013-12-27 22:55:40
Difficulty: 1180923195.26
Nonce: 3797028665
Merkle Root: 02327049330a25d4d17e53e79f
478cbb79c53a509679b1d8a1505c5697afb326
Transactions

**Fig. 2.1: A Fraction of Blockchain**

## 2.2 Peer-to-Peer Network (P2P)

Peer-to-Peer network at first introduced by Napster. Napster is an application which uses Peer-to-Peer network for sharing files most commonly compressed audio files format (mp3). However, Peer-to-Peer network not only used for file sharing but also used to establish a multimedia communication networks based on the Peer-to-Peer concepts. Basically "peers" means the computers systems are connected through the internet. Files can be shared directly throughout the networks without the help any server system. It is completely opposite to client-server model. Peer-to-Peer network works simultaneously as a file server as well as client. The basic and only requirement of using Peer-to-Peer network is to connect the computers via internet and P2P software's. Some common P2P software's are Kazaa, Limewire, BearShare, Morpheus, Acquisition etc [13]. Once someone is connected to P2P network, he can search for files and folders from other computers. In the meantime, other computers can also check files and folders from his computers which are shared only. For distinguishing Peer-to-Peer network definition, it can be splitted into two sub-definitions, "Hybrid" Peer-to-Peer network and "Pure" Peer-to-Peer network. The key difference between "Hybrid" Peer-to-Peer network and "Pure" Peer-to-Peer network is the "Hybrid" Peer-to-Peer network always adds a central entity. Our coinBD use Peer-to-Peer networks to connect all the nodes and passing the information between them. This also provide the security of coinBD. If someone wants to cheat or attempts to create an illegal transaction, he or she will be caught by the others as they are interconnected with the Peer-to-Peer network.

## 2.3 Bitcoin Network

Bitcoin is the first cryptocurrency invented by Satoshi Nakamoto at 2008. It is a decentralized digital currency having no control from central authorities. It uses cryptographic functions for overall operations. Entire Bitcoin runs by a blockchain in a peer to peer network where the blockchain is a distributed ledger. Each block of blockchain contains the information of transaction between sender user and receiver user which added to blockchain after mining. Whole blockchain with correspondent updates is being broadcasted to every node of the network after every single unit of mining. Blockchain technology is also developed by Satoshi firstly in order to build Bitcoin. As we know each block of blockchain contains previous hash value and its own hash value in Bitcoin network it also contains transaction information between two participants. Here participant could be anyone, coin owners or miners. Users have to maintain a wallet where they get unique public key and private key generated by cryptographic functions for authentications. When a participant sends coins to another one then a block gets initialized with its default parameters which was previously described. Then the block gets some new fields to import transaction information. It stores public keys of both participants and transaction amount and then it goes through a validation process which is also controlled by cryptographic operation. If the block is valid for transaction, then it gets passed to mining pool otherwise it gets dumped. When a miner gets a block for mining then the block also has to go through another validation process before gets mined which is called proof of work [3]. Proof of work basically generated by cryptrographic hash algorithm but in Bitcoin network miner has to solve the proof of work to mine a block according to given difficulty by the network itself which varies. To resolve it miner needs high computation power and network latency. Usually it takes 10 minutes to solve a proof of work in Bitcoin network but it differs from time to time according to their difficulty setup. After solving the proof of work mining process is successful then the new block gets mined in blockchain and gets broadcasted to every peer on the network and miner gets amount of Bitcoins as reward which depends on difficulty level he solved passed while solving proof of work. The whole system is running under the protection of cryptography. As it is a peer to peer decentralized network so no one has the super ability to bring any changes in the system except having the approval from majority of participants. About blockchain earlier we have discussed about robustness of its security. Finally, it takes a lot of computational power to solve proof of work then to manipulating it will cost an attacker nearly unimaginable computation. Therefore, Bitcoin can be considered as highly secured cryptocurrency.

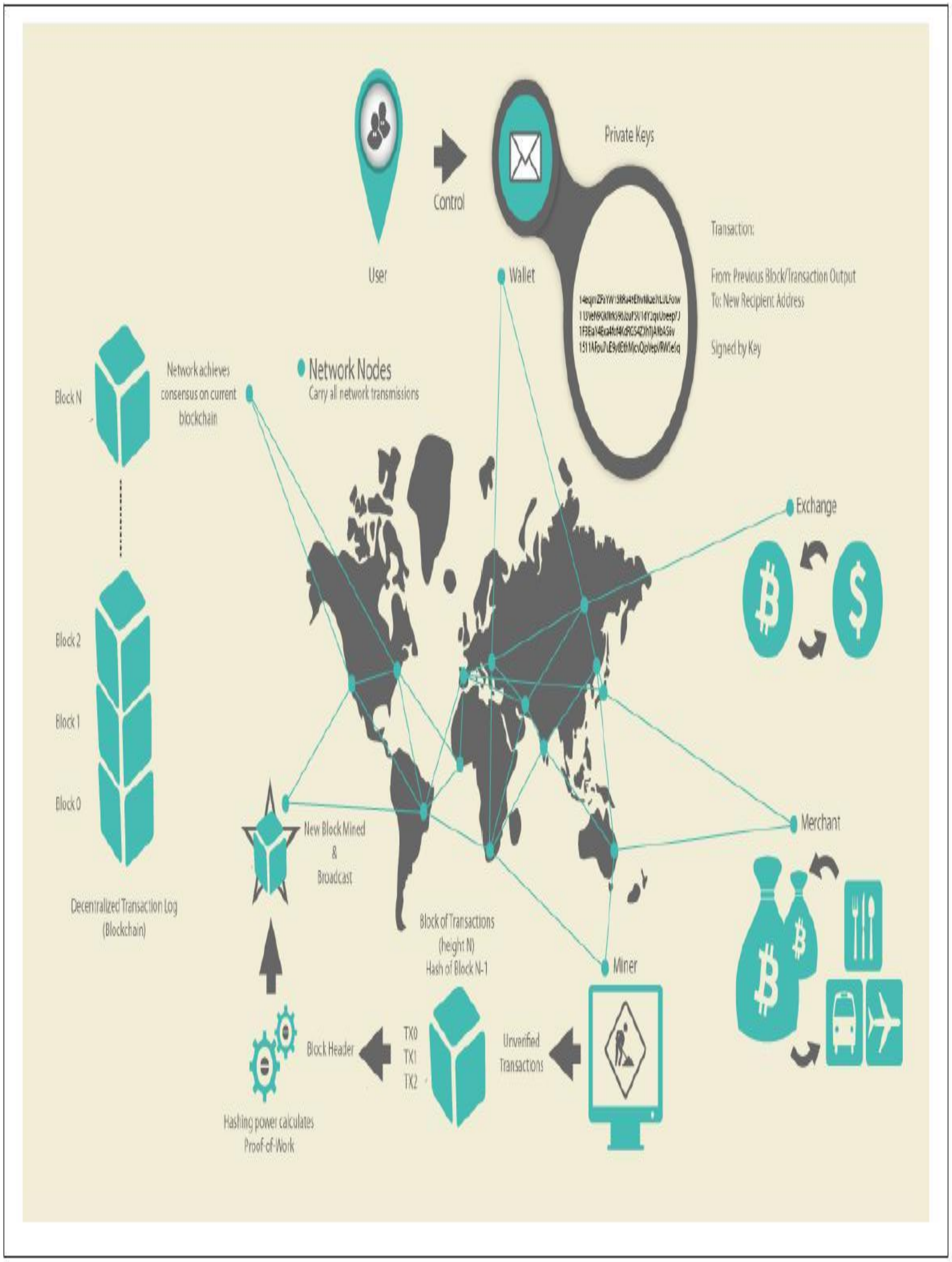

User
Control
Wallet
Private Keys
Transaction:
From: Previous Block/Transaction Output
To: New Recipient Address
Signed by Key
Network Nodes
Carry all network transmissions
Network achieves consensus on current blockchain
Block N
Block 2
Block 1
Block 0
Decentralized Transaction Log (Blockchain)
Exchange
Merchant
Miner
New Block Mined & Broadcast
Block of Transactions (height N) Hash of Block N-1
Unverified Transactions
TX0
TX1
TX2
Block Header
Hashing power calculates Proof-of-Work

**Fig. 2.2: Bitcoin Network**

## 2.4 Consensus Algorithm

After getting a block in mining pool in Bitcoin network again validate then mine the block to generate new unit of it. This total sequential and spontaneous operation is known as consensus algorithm '[14]. When a transaction occurs if sender have sufficient balance then newly generated blocks with previously defined parameters goes through a validation process where it gets signature hashed by SHA256 hash algorithm which gets created by checking sender's private key, we can consider it as a password is used as validator of transaction block. Then the block is being sent to mining pool. From here consensus starts. When miner gets a block for mining he has to solve another validation process named proof of work [15]. Proof of work is a hexadecimal representation of hash value of SHA256 hash algorithm which generates 256 bits length hash value. Therefore, proof of work consists the validator of 64 bits in hexadecimal. Miner needs to solve or match the proof of work according to current network provided difficulty. Difficulty is the benchmark which decides the reward of miner for mining blocks. For Bitcoin it works like that if difficulty set 00 that means miner needs to generate proof of work which has hexadecimal representation combined in such way where it's left most two bits are zero. In this difficulty miners get 50 bitcoins as reward. If difficulty is 0000 then proof of work's left most 4 bits should be zero and reward will be 25 coins. In such ratio difficulty and reward is being decided by Bitcoin by monitoring participation and competition of miners. After successfully creating proof of work new block gets mined in the blockchain and miner gets his reward. Blockchain then broadcasts its updated version to every user in the network. Although consensus of Bitcoin is highly secured and trustworthy but it needs huge amount of electricity, computational power and network bandwidth.

## 2.5 Solving forks

As a result of being a decentralized system every nodes of the peer has the same copy of blockchain. While mining a new block miner has to solve proof of work which depends on some vital factors as computational power, network latency, geographical distance etc.

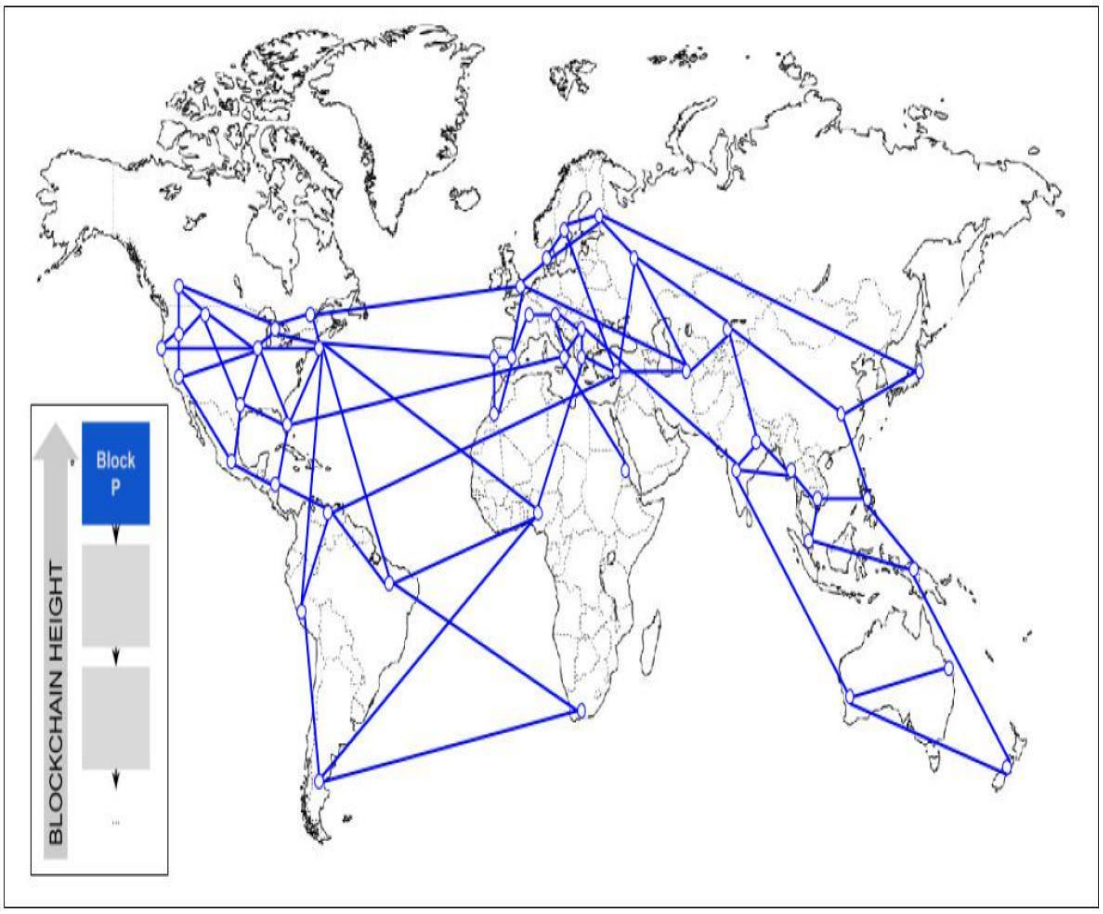


**Fig. 2.3: Blockchain before Fork**

In Fig. 2.3, we see the whole blockchain network around the world. Nodes are interconnected between each other. Every node contains the blockchain where last block is block P. When a block is sent to mining pool miner needs some time to solve proof of work. As it is a decentralized system another miner from another zone may also solve the proof of work for the same block within difference of nano seconds of time as a result of having high configured machine or fast bandwidth or situated nearest zone from mining pool. As a result, same block is created with two different identifiers. As because there were no invalid proof or activities occurred therefore blockchain does not reject any blocks. It just generates branches in the chain connected with last block if for figure 1 it is block P according to newly mined blocks and add them as last block according to their mining zones. It creates the scenario that two different blocks having different identities but they are containing the same transaction information. This condition is called blockchain fork [2].

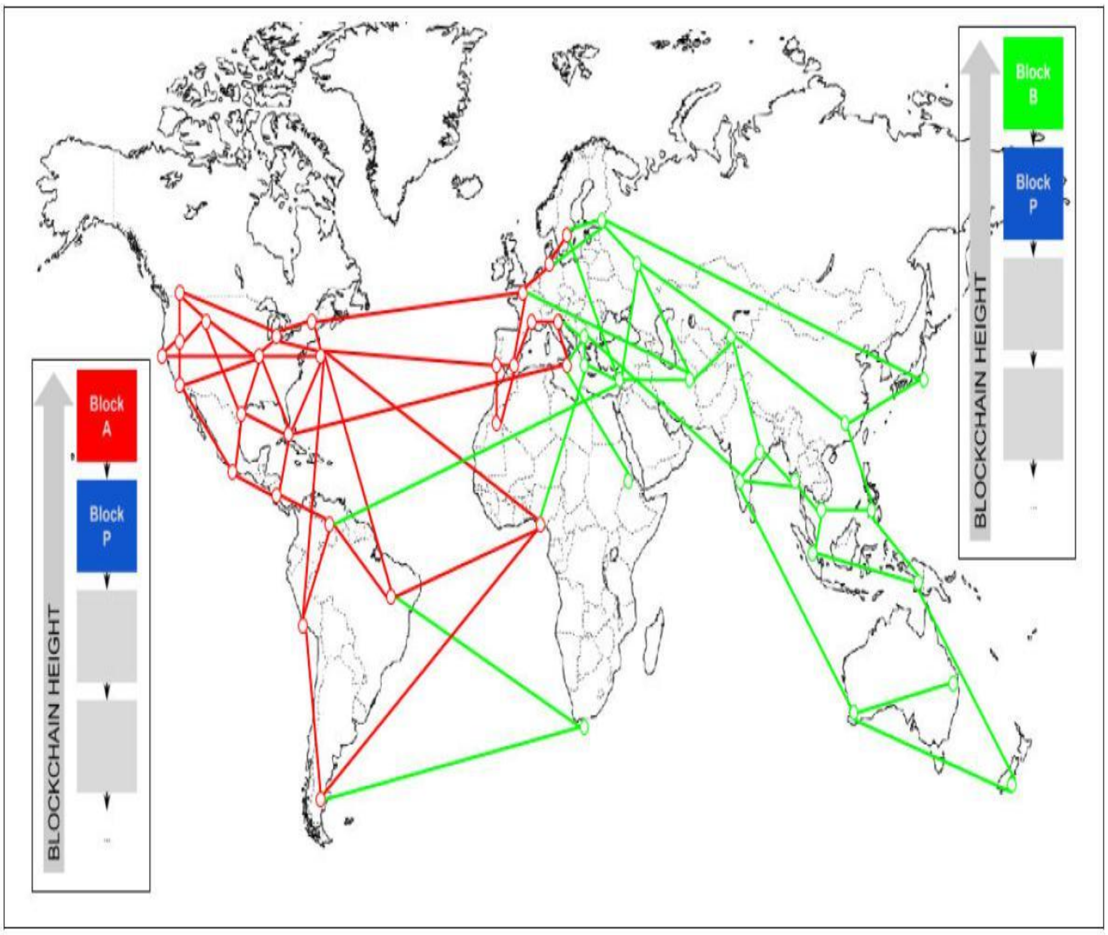


**Fig. 2.4: Blockchain after Fork Event**

In Fig.2.4, if we analyze for the fork network divided into red and green part and at the red part last block is identified as A and green part it is B where both of them are containing same data. From now blocks which are mined from nearest pools from A they will be connected with block A same will happen for B. The process will again run simultaneously in this way. Blockchain will decide here a winner chain which will be accepted or rejected in blockchain by monitoring the largest local chain or higher number of proof of work generator. To handle this blockchain does not monitors only certain number of block generation in local chains from their correspondent pools. Consensus resolves the problem by participation of a random pool which is neither near from A nor from B. It waits for generating and mining blocks from another zoned pool to watch the behavior which local chain it connects.

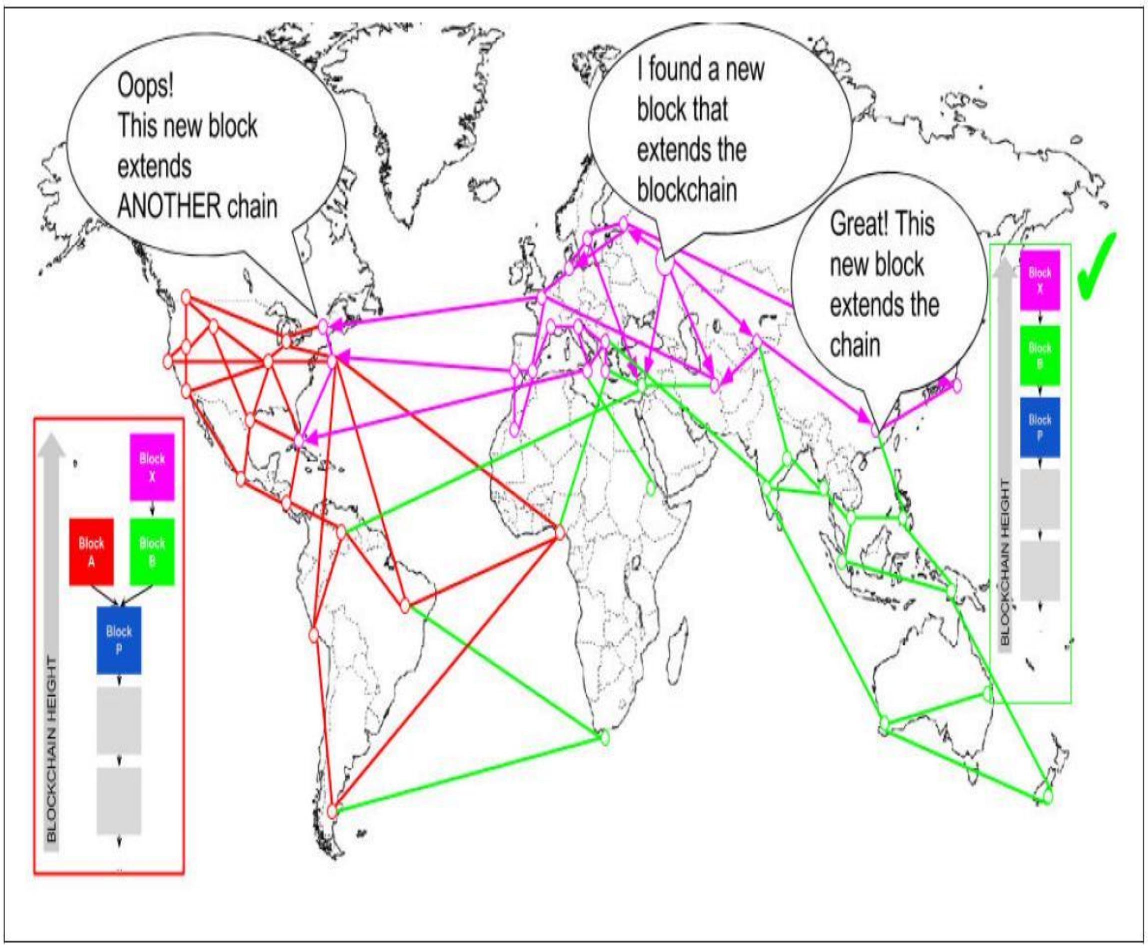

Oops! This new block extends ANOTHER chain
I found a new block that extends the blockchain
Great! This new block extends the chain
BLOCKCHAIN HEIGHT
Block X
Block A
Block B
Block P
BLOCKCHAIN HEIGHT
Block X
Block B
Block P

**Fig. 2.5: Selecting New Longest Chain**

If we look at Fig.2.5, new block X generated from another common pool it gets mined under block B. Therefore, all next blocks from that common pool and previous pool will be connected in local chain generated from B which will make it the longer chain. Therefore, consensus will choose the local chain generated from B as global blockchain. It will then remove the children of block A from their local chain and dump them in the mining pool in mining queue and permanently remove block A from the system. Children of block A will be mined again in regular process and blockchain will be a one branched simultaneous chain again. This is how consensus resolves the fork issue of blockchain.

## 2.6 Merkle Root Trees (MRT)

Merkle Root Tree is a data structure which basically is a binary tree which contains cryptrographic hash values [21]. It is efficient for verification of vast amount of data. In Bitcoin network every blocks of its blockchain uses Merkle Root Tree to keep the summary of transactions. Merkle Root Trees generate signatures using cryptographic hash algorithm which provides extended security in verifying transactions in a designated block. Bitcoin uses SHA 256 hash algorithm to generate digital fingerprint in its tree and it is used twice which makes the process known as double SHA 256 [2]. Another key efficient point for the tree is it is of very low complexity. N amount of data gives the worst case of 2*log(N) which very smooth calculative for data structures. MRT is constructed by a pair of hashed nodes which traverses recursively until the last node is found which is known as merkle root [2].

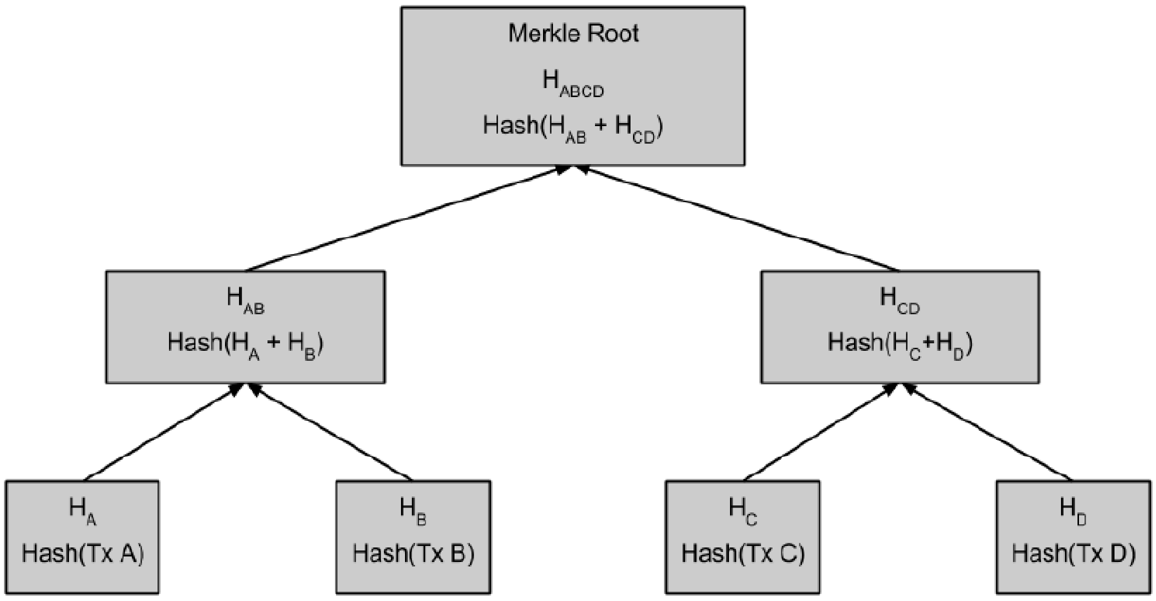


**Fig. 2.6: Calculating Nodes**

In Fig 2.6, four transactions of four blocks are indicated as A, B, C and D. These transactions first get hashed by cryptographic hash algorithm in Bitcoin which is SHA 256. The hashed transactions are here $H_A$, $H_B$, $H_C$, and $H_D$. For Bitcoin it works as if we consider for transaction A then,

$$H_A = \text{SHA 256(SHA 256(transaction A))}$$

This process is known as double hash. These hashed transactions stored in leaf nodes of Merkle Root Tree in pairs. Then two related child nodes sum up their hash values together and construct their parent node with the combined hash. In this case two 32 bits hashed child nodes combines their hash and construct their 64 bits hashed parent. The process works as if we consider this for $H_A$ and $H_B$,

$$H_{AB} = \text{SHA 256(SHA 256(}H_A + H_B\text{)}$$

This procedure continues until the only one node found which is the merkle root.

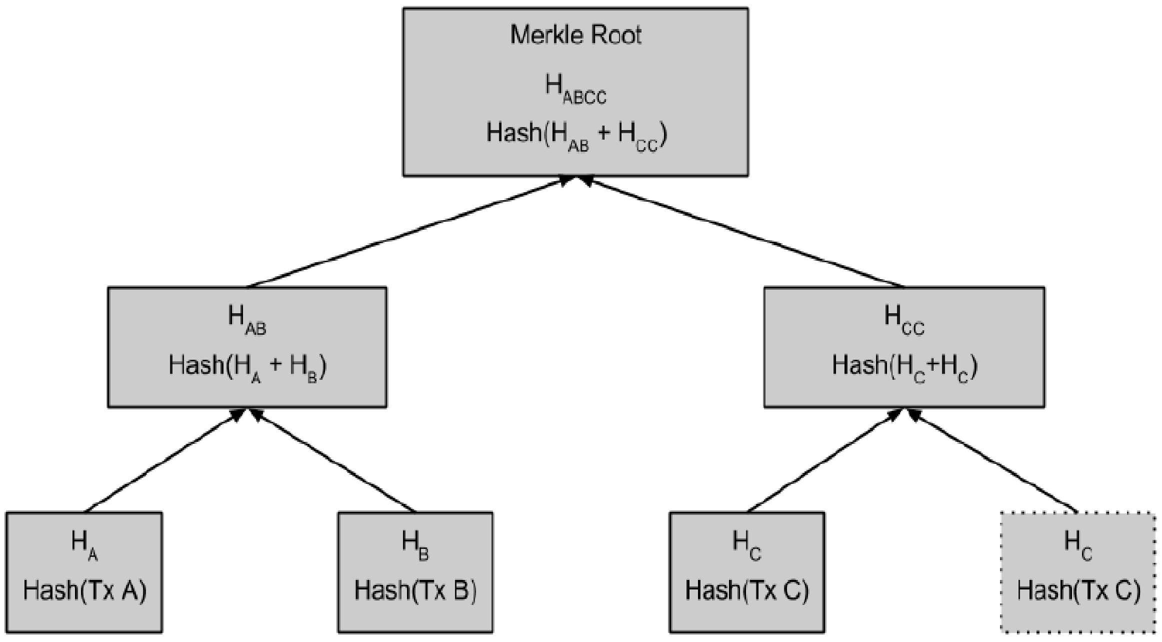


**Fig. 2.7: Duplicating for Odd Numbered Transaction**

In Fig. 2.7, we spectate scenario for odd numbers of transaction information. If transaction number is odd, then the last transaction hash gets duplicated and it gets added into leaf nodes which makes even number of leaf nodes as it is a binary tree it is compulsory to maintain even numbers of leaf nodes and the process handles the odd numbered transactions issue perfectly.

# Chapter 3

# Implementation

Bitcoin's wallet is so far the best wallet available still now so for this reason with some edition we used the same wallet in our project.

## 3.1 Wallet

To start our service a user need to install our wallet in their system. A unique user will receive a unique 512 bits of random seeds. From this personal unique random seeds, it will always generate a fixed 48 mnemonic words. Mnemonic seed is a series of 2048 words of dictionary which is represented in a matrix from. With the help of mnemonic words, we can always generate a same unique 512 bits of hexavalued private key by hashing it a specified limited numbers of time. With these upper few steps of wallet mechanism, a user will have his own private key. Private Key is fixed for one wallet of one user but a wallet will always keep track of the transactions with the help of public key. We used Elliptic Curve Digital Signature Algorithm (ECDSA) to generate a 512 bit of hexvalue from a 512 bits of private key.

### 3.1.1 ECDSA

ECDSA algorithm is based on Elliptic Curve Cryptography. ECDSA is the most secured public key generator algorithm nowadays and it's 163 bit guarantees as secure as 1024 bit Rivest-Shamir-Adleman (RSA) public key algorithm which has been also used in bitcoin's initial epoch not only this but also ECC is more appropriate for use on secure devises such as smart cards and wireless devices with constrained computational power consumption and memory resources [7].

The mathematical operations of ECC is defined over the elliptic curve $y^2 = x^3 + ax + b$ , in this equation for different values of a and b it gives a different elliptic curve. Bitcoin uses 0 and 7 value in terms of a and b and as well we also used it. After putting 0 and 7 in a and b the equation will be $y^2 = x^3 + 7$ , then it will be look like:

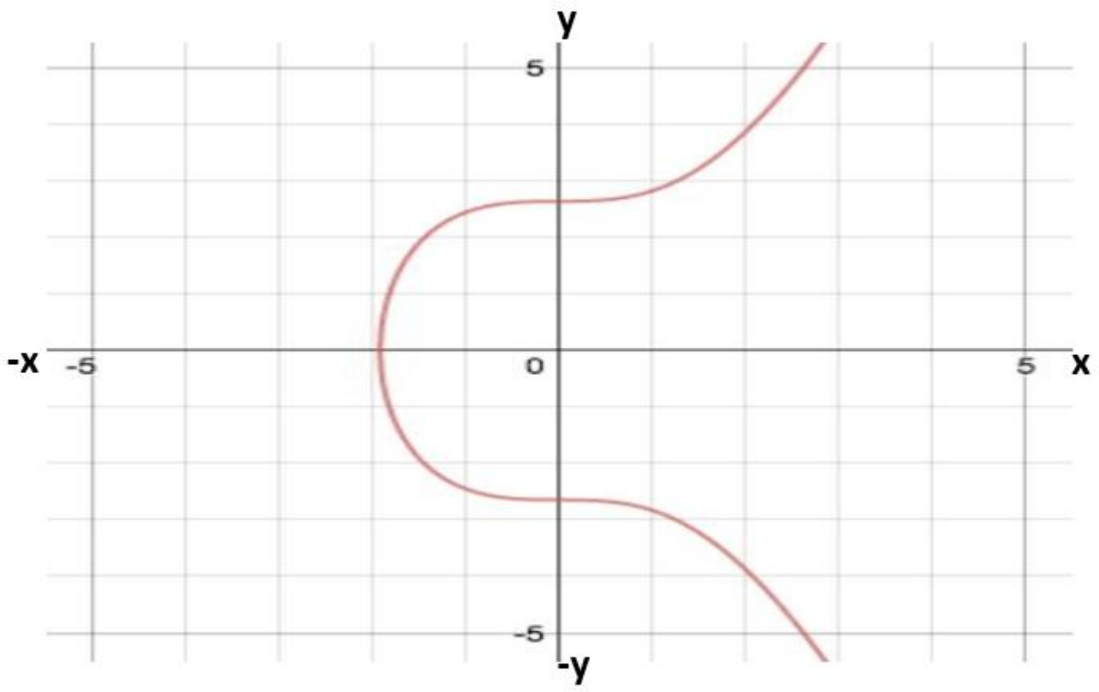


**Fig. 3.1: ECC Curve**

Elliptic curve has many useful properties to use. For instance, a non-vertical line intersecting. Two non-tangent points on the curve will always intersect a third point on the curve. Another property is that a non-vertical line tangent to the curve will always intersect precisely one other point on the curve. These two features can be used for point addition and point doubling. There is another feature, point subtraction which will not be needed for our project.

**Point Addition:**

Point addition is the addition of two points J and K on an elliptic curve to obtain another point L on the same elliptic curve.

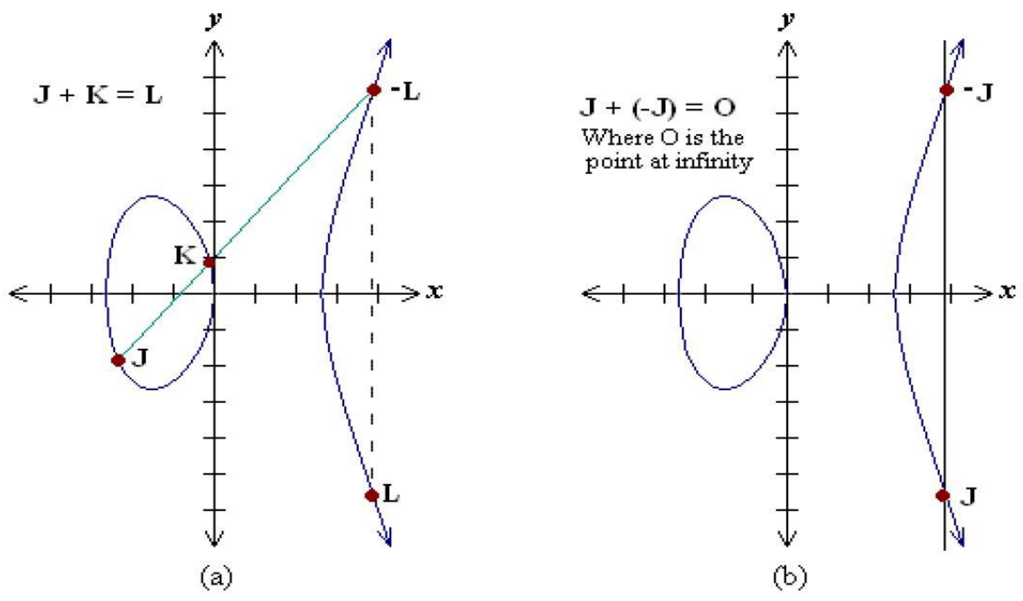


**Fig. 3.2: Point Addition**

On figure (a) J and K are two points which are not equal then by drawing a line through J and K will always intersect the elliptic curve at exactly one more point –L. The reflection of the point –L with respect to x-axis gives the point L, which is the result of these two point addition. So the result will be, $L = J + K$.

Consider two distinct points J and K such that J = $(X_J, Y_J)$ and K = $(X_K, Y_K)$

Let, $L = J + K$ , where $L = (X_L, Y_L)$, then

$$S = (Y_J - Y_K)/(X_J - X_K)$$

S is the slope of the line through J and K.

$$X_L = S^2 - X_J - X_K$$

$$Y_L = -Y_J + S(X_J - X_L)$$

If $K = -J$ i.e. $= (X_J, X_J)$ , then $J + K = O$. Where O is the point at infinity.

If $= J$, then $J + K = 2J$ then point doubling equations are used.

Also $J + K = K + J$

**Point Doubling:**

Point doubling is the addition of a point J on the elliptic curve to itself to obtain another Point L on the same elliptic curve. J is a point on an elliptic curve as shown in figure (a). If J has no y coordinate then if we draw a tangent line at J, it will intersect the elliptic curve at elliptic curve at only on another point -L with respect to x-axis gives the point L and that is the result of L=2j.

Consider a point $J$ such that $J = (X_J, Y_J)$, where $Y_J \neq 0$

Let, $L = 2J$ where, $L = (X_L, Y_L)$, Then

$$S = (3{X_J}^2 + a)/2Y_J$$

S is the tangent at point J and a is one of the parameters chosen with the elliptic curve

$$X_L = S^2 - 2X_J$$

$$Y_L = -Y_J + S(X_J - X_L)$$

If, $Y_J = 0$, then $2J = O$, where $O$ is the point at infinity.

The domain parameters for Elliptic curve over $Fp$ are $p, a, b, g, n \; and \; h.$ p is the prime number defined for finite field $p$. a and b are the parameters defining the curve $y2\ mod\ p = \ x^3 \ + \ ax \ + \ b\ mod\ p$. G is the generator point$(X_G, Y_G)$, a point on the elliptic curve chosen for cryptographic operations. n is the order of the elliptic curve. The scalar for point multiplication is chosen as a number between 0 and n – 1. h is the cofactor where $h = \#E\left(F_p\right)/n$. $\#E(Fp)$ is the number of points on an elliptic curve.

As the Elliptic curve can continue for infinitely, bitcoin restrict it with finite field. We use the same rules if p is the prime number defined for finite field $Fp$. a and b are the perameters defining then $y2\ mod\ p = \ (x^3 + 7)\ mod\ p$ will be the curve.

Prime modulo,

p = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE FFFFFC2F

$X_G$ =55066263022277343669578718895168534326250603453777594175500187360389116729240

$Y_G$ =32670510020758816978083085130507043184471273380659243275938904335757337482424

The **order n** of the base point, which is not independently selected but is a function of the other parameters, can be thought of graphically as the number of times the point can be added to itself until its slope is infinite, or a vertical line.

n = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

In brief, this particular realization of values goes by the name of secp256k1 and is part of a family of elliptic curve solutions over finite fields proposed for use in cryptography [20].

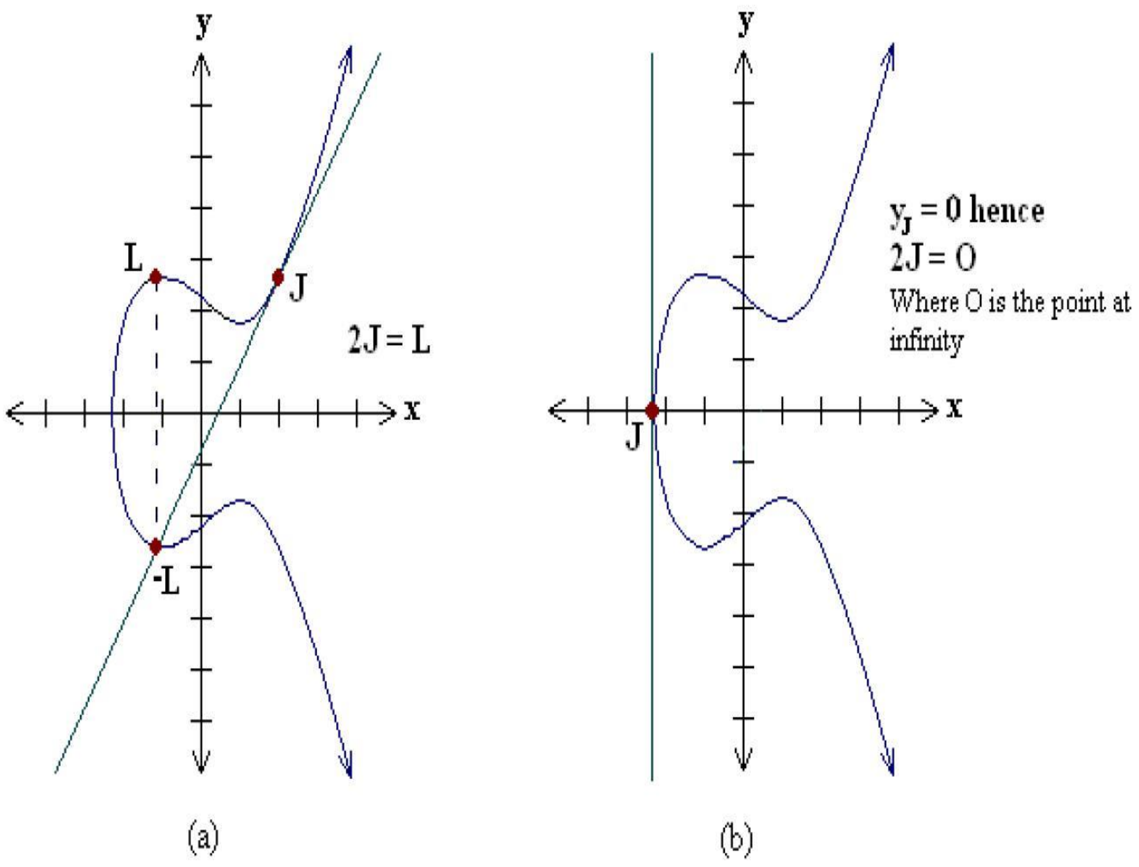


**Fig. 3.3: Point Doubling**

**Psudocode of Signing a Transaction:**

```
For signing a Transaction m by sender A, using A's private key $dA$
1. Calculate e = HASH (m), where HASH is a cryptographic hash function, such as SHA-512
2. Select a random integer k from [1,n − 1]
3. Calculate $r = x1 \; mod \; n$, where $(x_1, y_1) = k * G$. If r = 0, go to step 2
4. Calculate s = k − 1(e + dAr)(mod n) $s = (k - 1)(e + dAr) \; mod \; n$. If, $s = 0$, go to step 2
5. The signature is the pair (r, s)
```

**Fig. 3.4: Psudocode of Signing Transaction**

## 3.2 coinBD's Consensus algorithm

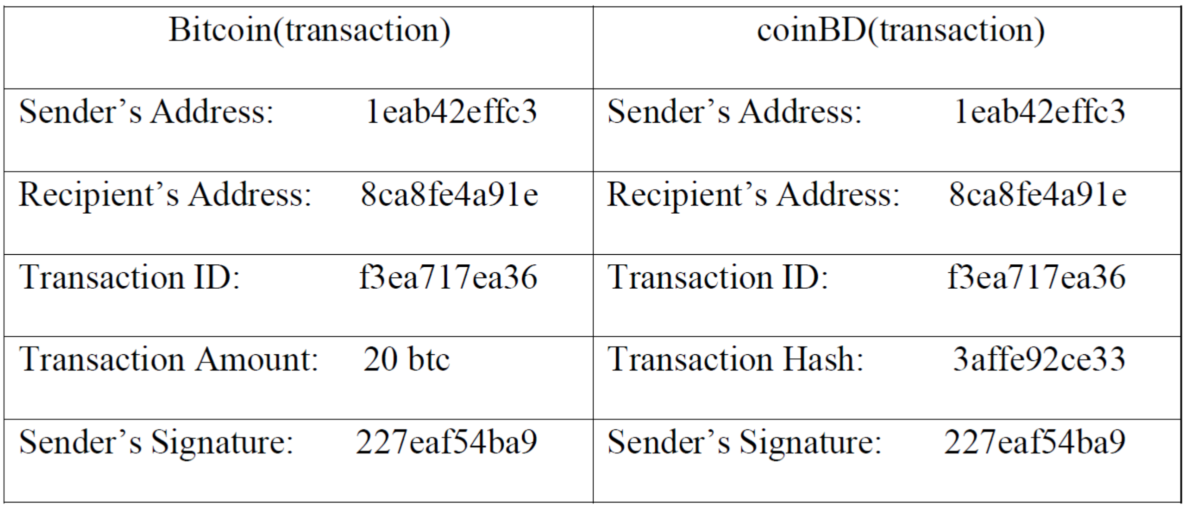
### Mining pool:

After reaching the transaction to the mining pool coinBD does a change in the transaction. Still now both Bitcon and coinBD will have same transaction attribute as below:

| Bitcoin(transaction) | | coinBD(transaction) | |
|---|---|---|---|
| Sender's Address: | 1eab42effc3 | Sender's Address: | 1eab42effc3 |
| Recipient's Address: | 8ca8fe4a91e | Recipient's Address: | 8ca8fe4a91e |
| Transaction ID: | f3ea717ea36 | Transaction ID: | f3ea717ea36 |
| Transaction Amount: | 20 btc | Transaction Amount: | 20 CBD |
| Sender's Signature: | 227eaf54ba9 | Sender's Signature: | 227eaf54ba9 |

**Fig. 3.5: Transaction before Broadcasting**

But just before picked up by a miner the pool will make a change in transaction amount attribute. In the place of transaction amount the pool will sent the double sha512 hash value of the whole transaction and keep the transaction amount to itself for later use. Finally it will be ready to pick up by any miner on the coinBD mining process. The change is shown below.



| Bitcoin(transaction) | coinBD(transaction) |
|---|---|
| Sender's Address: 1eab42effc3 | Sender's Address: 1eab42effc3 |
| Recipient's Address: 8ca8fe4a91e | Recipient's Address: 8ca8fe4a91e |
| Transaction ID: f3ea717ea36 | Transaction ID: f3ea717ea36 |
| Transaction Amount: 20 btc | Transaction Hash: 3affe92ce33 |
| Sender's Signature: 227eaf54ba9 | Sender's Signature: 227eaf54ba9 |

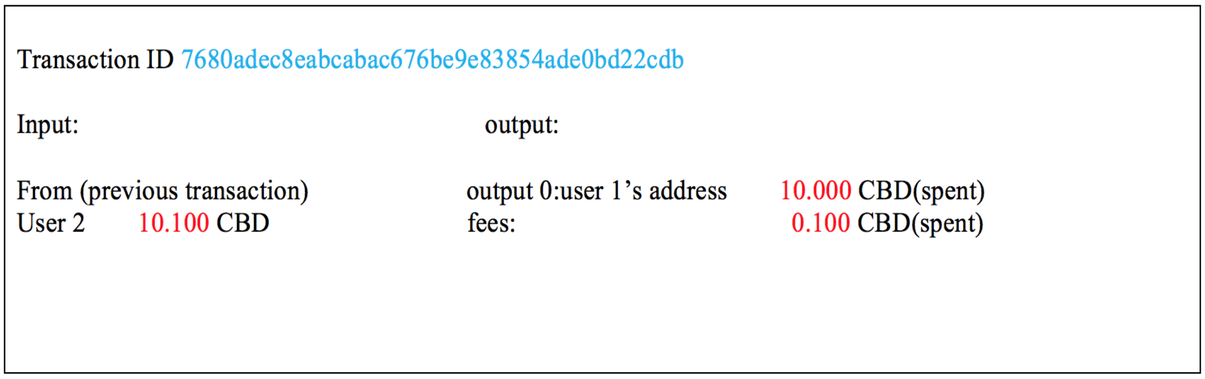
**Fig. 3.6: Transaction after Broadcasting**

**coinBD Transaction:**

Technically, transactions are basically moving from input to output. Inputs are where the coin value are coming from in other word sender’s public key which contains coins. And the outputs are where the coin value are spending or receiver’s public key where the money will be sent [12]. Suppose, user 1 is a new user who achieved his first coinBD in amount 10 BDC from his friend user 2 for exchange for physical money.

**User 2’s Transactions:**

Now user 1 wants to buy something from user 3 which will cost 5 BDC. User 1 is now making a transaction with user 3.

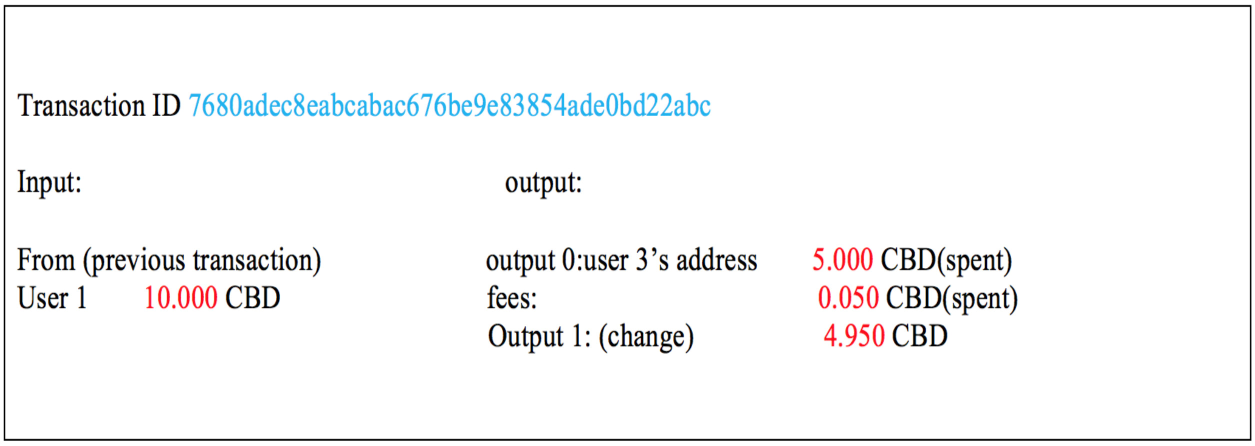


Transaction ID 7680adec8eabcabac676be9e83854ade0bd22cdb

| Input: | | output: | |
|---|---|---|---|
| From (previous transaction) | | output 0:user 1’s address | 10.000 CBD(spent) |
| User 2 | 10.100 CBD | fees: | 0.100 CBD(spent) |

**Fig. 3.7: User 2 Transaction**

**User 1’s Transaction:**

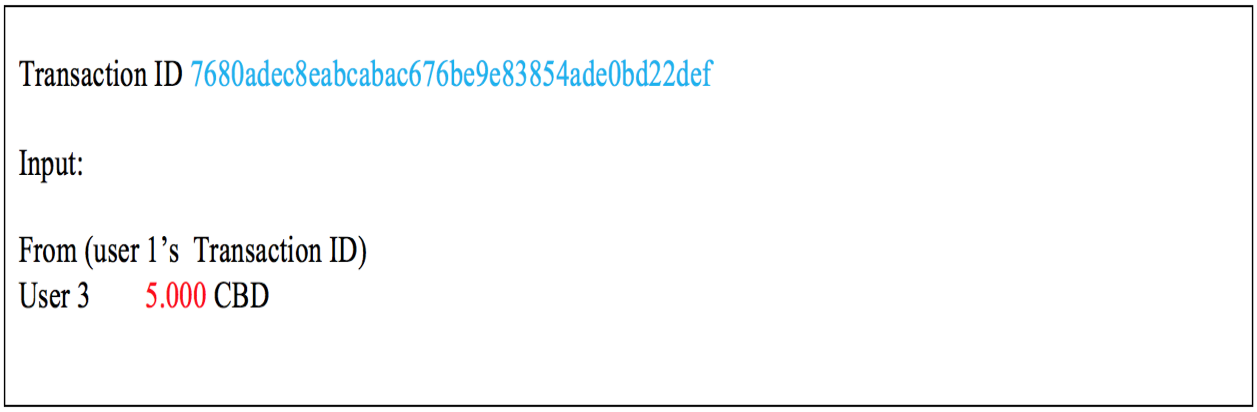


Transaction ID 7680adec8eabcabac676be9e83854ade0bd22abc

| | | | |
|---|---|---|---|
| Input: | | output: | |
| From (previous transaction) | | output 0:user 3’s address | 5.000 CBD(spent) |
| User 1 | 10.000 CBD | fees: | 0.050 CBD(spent) |
| | | Output 1: (change) | 4.950 CBD |

**Fig. 3.8: User 1 Transaction**

**User 3’s Transaction:**



Transaction ID 7680adec8eabcabac676be9e83854ade0bd22def

Input:

From (user 1’s Transaction ID)
User 3 5.000 CBD

**Fig. 3.9: User 3 Transaction**

If we take storage for both input and output balance, then by simply doing input – output we can get the balance of a user. Let’s say, user 1 is a new client of our services. He buys some BDC from user 2. User 2 is making this transaction for this reason the spent amount of BDC is adding to user 2’s output storage and the unspent amount in the input storage as well as the specific transaction amount is added to user 1’s input storage (Fig 2.8). But when user 2 is making a transaction the spent amount is going to be added to user 2’s output storage and the unspent amount in the input storage as well as the specific transaction amount in the user 3’s input storage which user 3 can spend later. This transaction balance system is just partially different from Bitcoin’s current transaction system. Then the transaction will be send to the p2p network for reaching to mining pool.

```
Psudocode of Transaction:
If Sender(x) has public key and sender’s signature
        Sending transactions for x = Select all the transaction where x is sending money
        If (Output == Balance (Sending transactions for x))
                Continue
        Else break
            Receiving transaction for x = Select all the transaction where x is receiving
                                                                                money
        If (Input == Balance (Receiving transactions for x))
                Continue
        Else break
Funds = Input – Output
Return Funds
```

**Fig. 3.10: Psudocode of Transaction**

## Consensus algorithm:

As referred earlier Bitcoin's consensus algorithm takes four parameters which are wallet, blockchain, network and difficulty [12]. For coinBD's consensus algorithm there is a change in perimeters. In place of difficulty the system will provide a modified transaction. As bitcoin's uses some work against rewarding miners and this work cost high computational power and electronics and by setting difficulty (varies from time to time). The Bitcoin network customizes to produce each block in about ten minutes length by providing difficulty level [5]. Though with the prospering technology the same computation can be solved easily but by increasing difficulty it can be controlled. But coinBD does not require any difficulty or any time length on the other hand in exchange of a hard computation of nonce miner will return an iteration value (we will discuss it in later part). It can be called partially real time exchange of cryptocurrency.

## Mining:

When a miner pulls a transaction from the mining pool in terms of bitcoin miner does about ten minutes of computational work and get a nonce value and again throws it to the mining pool. But in coinBD's case miners are not going to generate any nonce but they had to do some work in exchange as there is no third party to secure the transaction. What the miners will actually do is they have to guess the transaction amount this work is called Proof of Rigor.

**Use Case Diagram:**

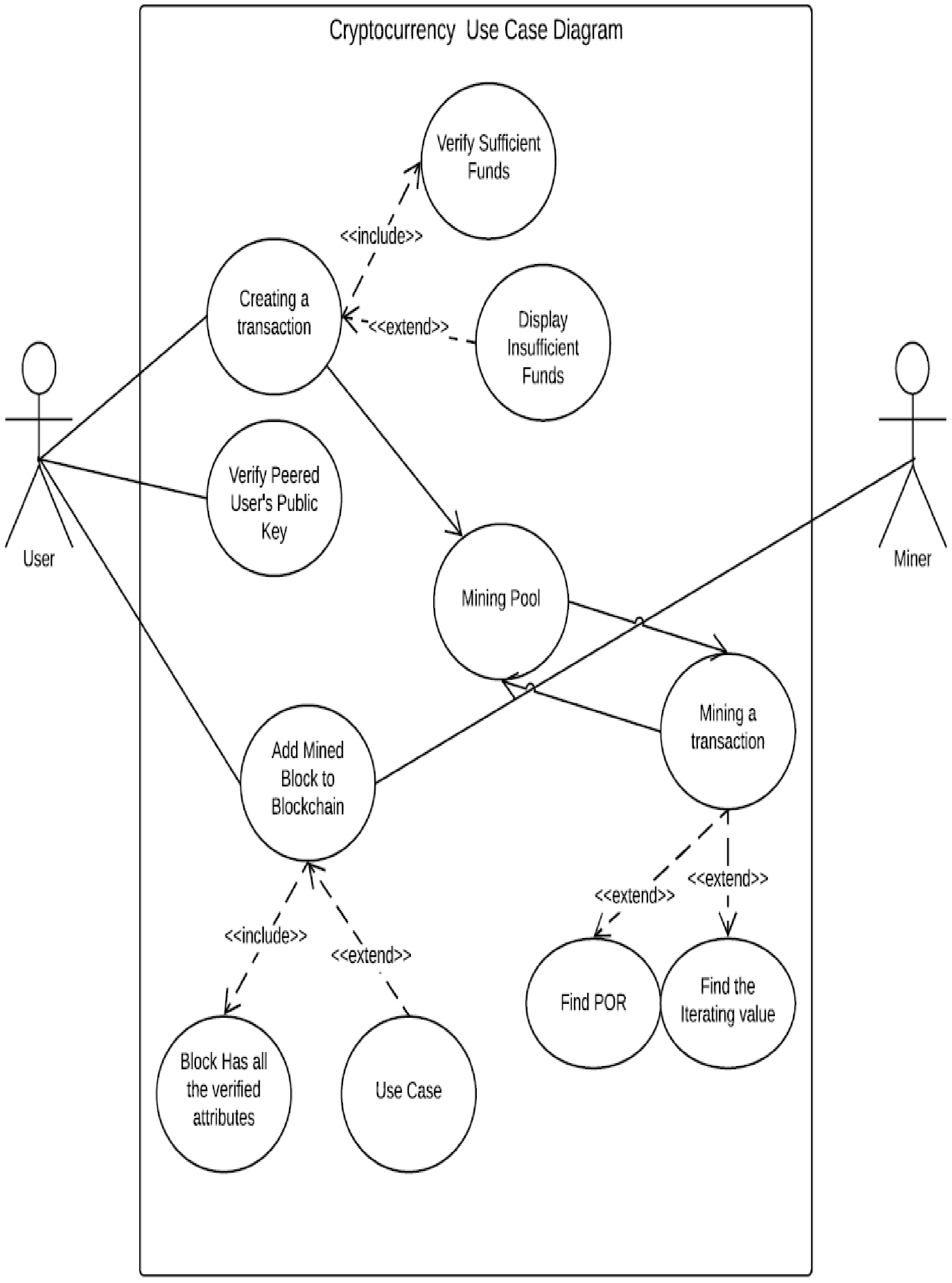


**Fig. 3.11: coinBD’s Use Case Diagram**

**Proof of Rigor:**

Proof of Rigor is a simple and thorough process for mining coin. It is close to real time and will take only millisecond or few seconds depends on the value of transaction amount as the miner has to guess the transaction amount of that specific transaction. Mining pool will keep secret the transaction amount and sends the double sha512 hash value of the whole transaction. Mining pool creates the SHA value of sender's address, receiver's address, Transaction's Id, sender's signature and transaction amount.

SHA512 (SHA512 (Sender's Address + Receiver's Address + Transaction's Id + Sender's Signature + Transaction Amount)) = 512 bits of Transaction hex value.

Let's say, user 4 has send user 5 a transaction amount 500 CBD. The system knows how much is the transaction amount and it also knows that how much Iteration Value it will need. Then miners will take the transaction and start to compare the value with default iteration value of Zero. When they will find the true Iteration Value they will create a block with hash value of the last added block in the blockchain as previous block and send it to mining pool. As we mentioned earlier that mining pool knows that what will be the iteration value so it will check it and add it to the blockchain. An honest miner will get the reward but if some dishonest miner tries to create the same block without the accurate iteration value they will penalized for that.

**Psudocode of mining block:**

```
Func Mine_Block():
        Last Block = Blockchain[Last Index]
        For Iteration Value: 0→ infinite
                If(SHA_512(SHA_512(Sender's Address + Recipient's
                    Address +Transaction's ID + Sender's Signature +
                  iteration Value)) ==Transaction Hexvalue)
                        Create Block
                        Add SHA_512(Last Block) to the Block
                        Return the block to mining pool
```

**Fig. 3.12: Psudocode of Mining Block**

**Psudocode of resolving Fork:**

```
Longchain = self.chain
For node in self.peerNodes
        If node.chain.length > self.chain.length
                Validate(blockchain)
                Longchain = node.chian.length
                 Copy the blockchain to self.secondary.blockchain
```

**Fig. 3.13: Psudocode of Resolving Fork**

**Mining Reward:**

The new proof of rigour is fast and simple that the mining time will depend on how big the transaction amount is. For this reason, coinBD cannot always provide the same reward as bitcoin does. We are going to provide the reward according to the logarithm of iteration value.

$$Mining\ Reward\ =\ log\ (iteration\ value)$$

Psudocode of Mining Reward:

```
If ( Pool.Iterationvalue == Miner.Iterationvalue)
If the block has all the criterias
      $Mining\ Reward\ =\ log\ (Iterationvalue)$
      Acess will be given to add to blockchain
      Mining reward will be provided
```

**Fig. 3.14: Psudocode of Mining Reward**

But with the flow of time blockchain will be getting longer at that time same logarithmic reward will not suffice for miner's cost.

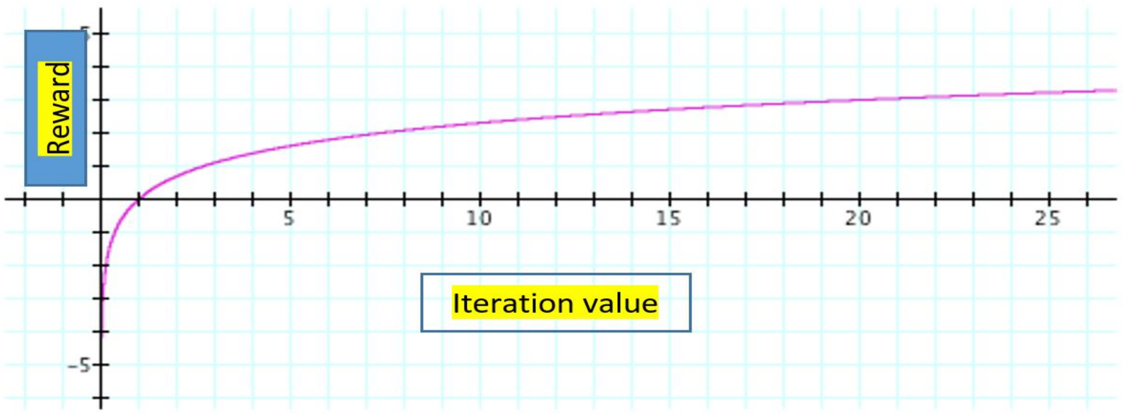


**Fig. 3.15: POR Reward Curve**

At that time the system will provide the mining reward according to the e base logarithm of iteration value.

$$Mining\ Reward\ =\ log_e\ (iterationvalue)$$

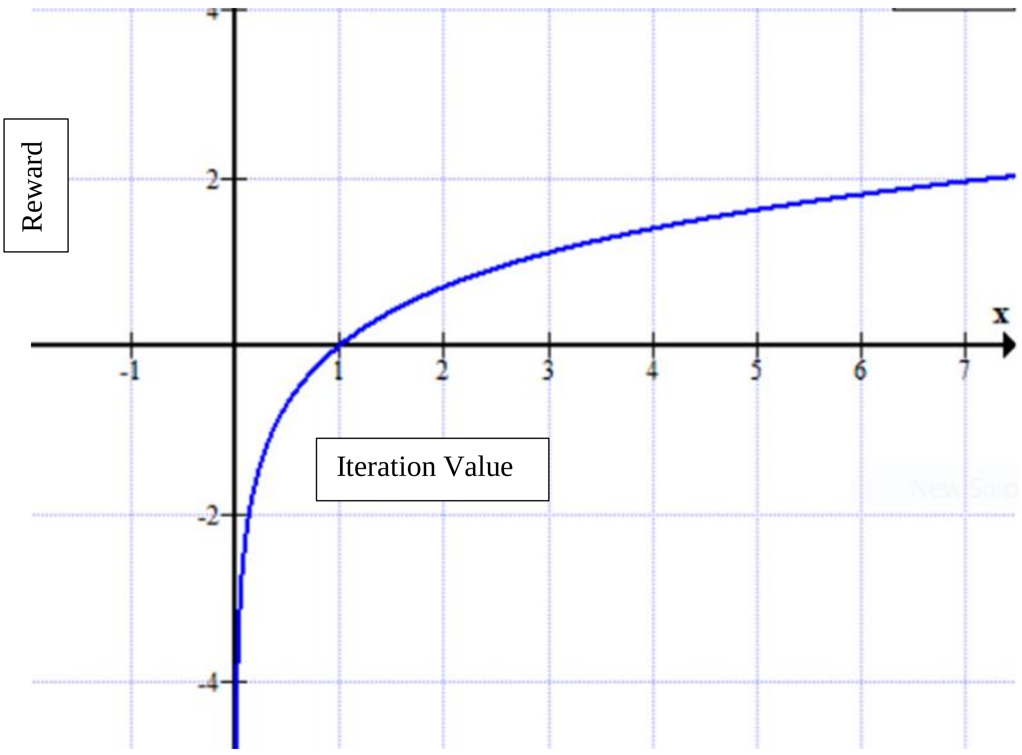


**Fig. 3.16: Future POR Reward Curve**

## Drawbacks:

As our Proof of Rigour is fast and through. There will be a chance for people to send money without any need to get mining reward. Because by doing that if two miner user send money to each other. When miner 1 send money to miner 2, then as our system is close to real time miner 1 will try to mine his own block and take the reward for himself and same goes to miner 2. To prevent this fraud to happen, coinBD will limit the user’s money transaction amount per day as a result dishonest people cannot arbitrarily make transaction as wish.

# Chapter 4

# Result

Our main purpose of this project was to reduce the computational energy used by bitcoin as a proof of work.

For this we have come up with new type of proof of work which is proof of rigour. It takes very few time to compute the required goal, the amount of work to determine the transaction amount.

The mining reward system of Bitcoin is not so feasible. Because for each block created the mining reward is same whether the transaction amount is big or small or how many transaction is attached to a block. On the other way around, from the proof of rigour, the system will have the iteration value which will be the key factor of mining reward. The miners will not be deprived or nor be over flowed with reward. It is a very feasible and has no central authorization which is also one of the key feature of cryptocurrency.

The main concept is time. I need some currency now at that time if bitcoin take some time about 10 to 15 minutes which is not acceptable. On the contrary coinBD take only several seconds of time to complete a transaction.

Bitcoin has a great flaw which is fork. Because of many miners trying to do the same proof of work on a same transaction as it stayed in the mining pool for at least 10 to 15 minutes. As a result 2 or 3 miners on different end of world will add a block with different hash as a result next blocks may be in right order that specific block will create a fork. On other point of view, coinBD transactions are not lurking around in the mining pool for very this reason happening a fork in coinBD's chance is very low.

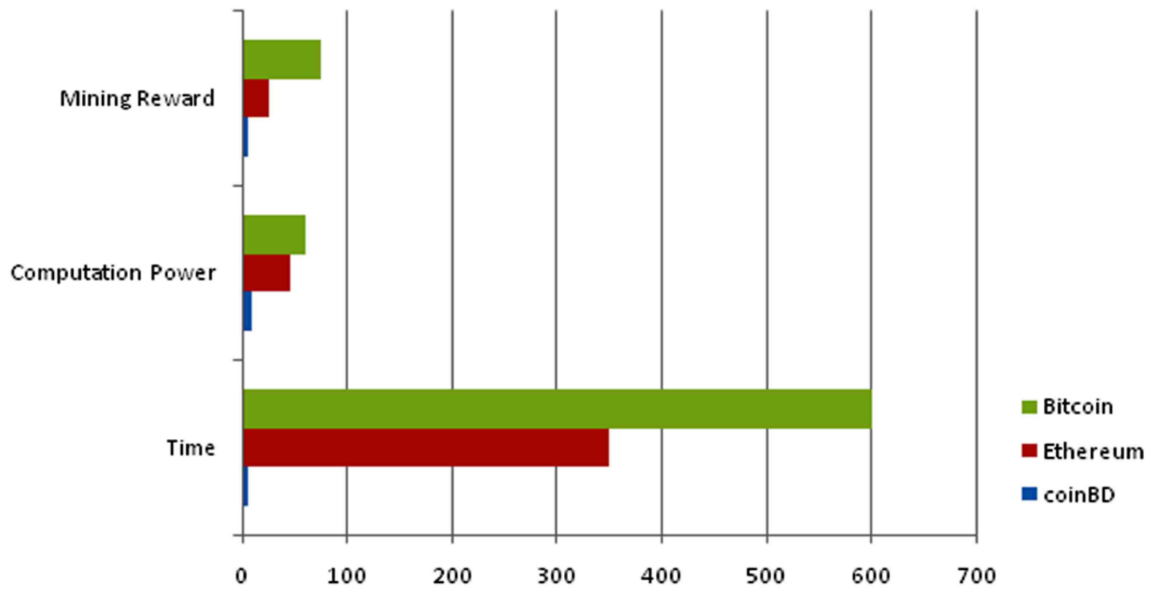


**Fig. 4.1: Comparison among Bitcoin, Ehereum, coinBD**

# Chapter 5

# Future Work and Conclusion

## 5.1 Conclusion

We have implemented our own cryptocurrency, following the mechanisms of bitcoin that has the secured transaction system as bitcoin but needs reduced computational energy for mining. Throughout the paper, our approach was to introduce our own mining validation system proof of rigor which is more simplified and less complex than bitcoin's proof of work that can determine the iteration value by guessing the transaction amount instead of finding the nonce in bitcoin. Moreover, we also use our own reward function that offers a miner an amount of reward based on the transaction amount by using a logarithm function. Our overall analysis has shown that our system would be faster and more feasible than bitcoin or other cryptocurrencies.

## 5.2 Future work

At present, our coinBD uses Merkle Trees as a security purpose similar to the Bitcoin. In future we are planning to use a newer version of trees named M.A.S.T which means Markelized Abstract Syntax Trees [Merkelized Abstract Syntax Trees]. M.A.S.T is a cryptographic tool that would be enable the addition in complex data sets in to the data associated with transactions. The two algorithms, Merkle Trees and Abstract Syntax trees are combined in M.A.S.T to provide a better security. Markle trees provide the way of recording the data, removing the need for all data to be downloaded before confirming that the data belongs in the set using the Markle Root. On the other hand Abstract Syntax trees makes the division and labeling of data in a set. These two algorithms provide a way of handling complex data in block chain. BIP114, BIP116 and BIP117are the three BIP's that have been proposed to develop M.A.S.T by bitcoin core developer Mark Friedenbach. Implementation of M.A.S.T would increase privacy, faster transaction speed and adding more complex data sets into our coinBD.

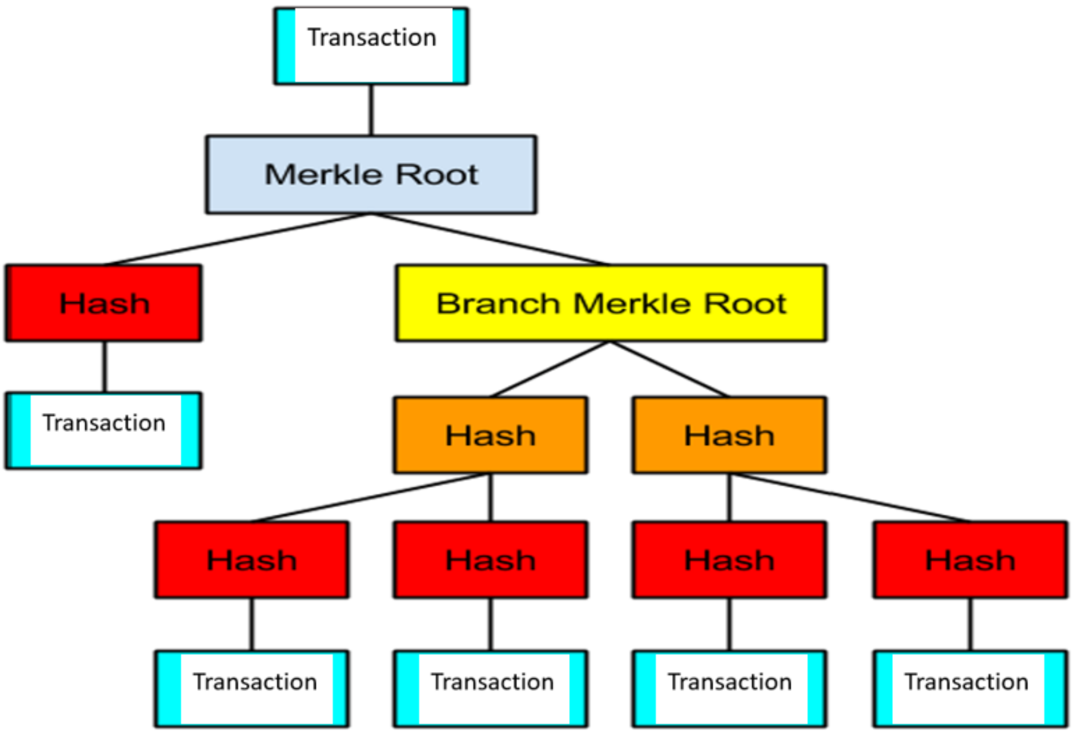


**Fig 5.1: Markelized Abstract Syntax Tree**

Currently, coinBD is giving reward to the miners based on the transactions amount by using a logarithm function that gives a miner a fixed amount of coinBD. In future, coinBD would use LN function for giving a better reward to the miners.

Recently, coinBD is using a logarithm function to set the difficulty in order to efficient mining same as the reward giving function of miner. This difficulty is enough for the initial stage. As our coinBD would bigger day by day, in future we would use prime number to set the adjustment of difficulty used in Primecoin [17].

## Reference

[1] Andrychowicz, M., Dziembowski, S., Malinowski, D., and Mazurek, L. (2014). Secure multiparty computations on bitcoin. In Security and Privacy (SP), 2014 IEEE Symposium on, pages 443–458. IEEE.

[2] Antonopoulos, A. M. (2014). Mastering Bitcoin: unlocking digital cryptocurrencies. " O'Reilly Media, Inc.".

[3] Back, A. et al. (2002). Hashcash-a denial of service counter-measure.

[4] Bentov, I. and Kumaresan, R. (2014). How to use bitcoin to design fair protocols. In International Cryptology Conference, pages 421–439. Springer.

[5] Buterin, V. et al. (2014). A next-generation smart contract and decentralized application platform. white paper.

[6] Emam, S. A. and Emami, S. S. (2007). Design and implementation of a fast, combined sha-512 on fpga. Int. J. Comput. Sci. Network Secur, 7:165–168.

[7] Han, J.-H., Kim, Y.-J., Jun, S.-I., Chung, K.-I., and Seo, C.-H. (2002). Implementation of ecc/ecdsa cryptography algorithms based on java card. In Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on, pages 272–276. IEEE.

[8] Kosba, A., Miller, A., Shi, E., Wen, Z., and Papamanthou, C. (2016). Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In 2016 IEEE symposium on security and privacy (SP), pages 839–858. IEEE.

[9] Kumaresan, R. and Bentov, I. (2014). How to use bitcoin to incentivize correct com-putations. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pages 30–41. ACM.

[10] McEvoy, R. P., Crowe, F. M., Murphy, C. C., and Marnane, W. P. (2006). Optimisation of the sha-2 family of hash functions on fpgas. In Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on, pages 6–pp. IEEE.

[11] Miller, A. and LaViola Jr, J. J. (2014). Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. Available on line: http://nakamotoinstitute.org/research/anonymous-byzantine-consensus.

[12] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[13] Schollmeier, R. (2001). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In Peer-to-Peer Computing, 2001. Proceedings. First International Conference on, pages 101–102. IEEE.

[14] Vujiciˇc,´ D., Jagodic,´ D., and Randi¯c,´ S. (2018). Blockchain technology, bitcoin, and ethereum: A brief overview. In INFOTEH-JAHORINA (INFOTEH), 2018 17th International Symposium, pages 1–6. IEEE.

[15] Xue, T., Yuan, Y., Ahmed, Z., Moniz, K., Cao, G., and Wang, C. (2018). Proof of contribution: A modification of proof of work to increase mining efficiency. In 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), pages 636–644. IEEE.

[16] Yang, B. and Garcia-Molina, H. (2003). Ppay: micropayments for peer-to-peer systems. In Proceedings of the 10th ACM conference on Computer and communications security, pages 300–310. ACM.

[17] King, S. (2013). Primecoin: Cryptocurrency with prime number proof-of-work. July 7th.

[18] Miller, A., Juels, A., Shi, E., Parno, B., & Katz, J. (2014, May). Permacoin: Repurposing bitcoin work for data preservation. In 2014 IEEE Symposium on Security and Privacy (SP) (pp. 475-490). IEEE.

[19] Daian, P., Eyal, I., Juels, A., & Sirer, E. G. (2017, April). (Short Paper) PieceWork: Generalized Outsourcing Control for Proofs of Work. In International Conference on Financial Cryptography and Data Security (pp. 182-190). Springer, Cham.

[20] Qu, M. (1999). SEC 2: Recommended elliptic curve domain parameters. Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6.

[21] Merkle, R. C. (1980, April). Protocols for public key cryptosystems. In Security and Privacy, 1980 IEEE Symposium on (pp. 122-122). IEEE.