

# Distributed Bitcoin Miner

Team 32

April 29, 2021

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Bitcoin</b>	<b>2</b>
<b>3</b>	<b>Our Approach</b>	<b>3</b>
3.1	Flow Diagram . . . . .	3
3.2	Transactions . . . . .	4
3.3	Timestamp Server . . . . .	4
3.4	Proof of work . . . . .	4
3.5	Difficulty level . . . . .	5
<b>4</b>	<b>Features</b>	<b>5</b>
4.1	Network . . . . .	5
4.2	User Interface . . . . .	6
4.2.1	Fields Description: . . . . .	6
4.2.2	Miner Details: . . . . .	6
4.3	Merkle Tree . . . . .	7
4.4	Block Structure: . . . . .	8
4.5	Blockchain: . . . . .	8
4.6	Resolve Conflicts: . . . . .	9
4.7	Reward: . . . . .	9
<b>5</b>	<b>Scope Of Improvement:</b>	<b>9</b>
<b>6</b>	<b>Video Link</b>	<b>9</b>
<b>7</b>	<b>References:</b>	<b>9</b>
<b>8</b>	<b>Team Members</b>	<b>9</b>

## 1 Introduction

Bitcoin was created by Satoshi Nakamoto, who published the invention and later it was implemented as open source code. A merely end- to-end version of electronic cash would allow online payments to be sent straight from one person to another without going through

an economic body. Bitcoin is a network practice that enables folks to transfer assets rights on account units called "bitcoins", created in limited quantities. When a person sends a few bitcoins to another individual, this information is broadcast to the peer-to-peer Bitcoin network.

Central to Bitcoin's operation is a global, public log, called the blockchain, that records all transactions between Bitcoin clients. The security of the blockchain is established by a chain of cryptographic puzzles, solved by a loosely-organized network of participants called miners. Each miner that successfully solves a crypto puzzle is allowed to record a set of transactions, and to collect a reward in Bitcoins. The more mining power (resources) a miner applies, the better are its chances to solve the puzzle first. This reward structure provides an incentive for miners to contribute their resources to the system, and is essential to the currency's decentralized nature.

The Bitcoin protocol requires a majority of the miners to be honest; that is, follow the Bitcoin protocol as prescribed. By construction, if a set of colluding miners comes to command a majority of the mining power in the network, the currency stops being decentralized and becomes controlled by the colluding group. Such a group can, for example, prohibit certain transactions, or all of them. It is, therefore, critical that the protocol be designed such that miners have no incentive to form such large colluding groups.

## **2 Bitcoin**

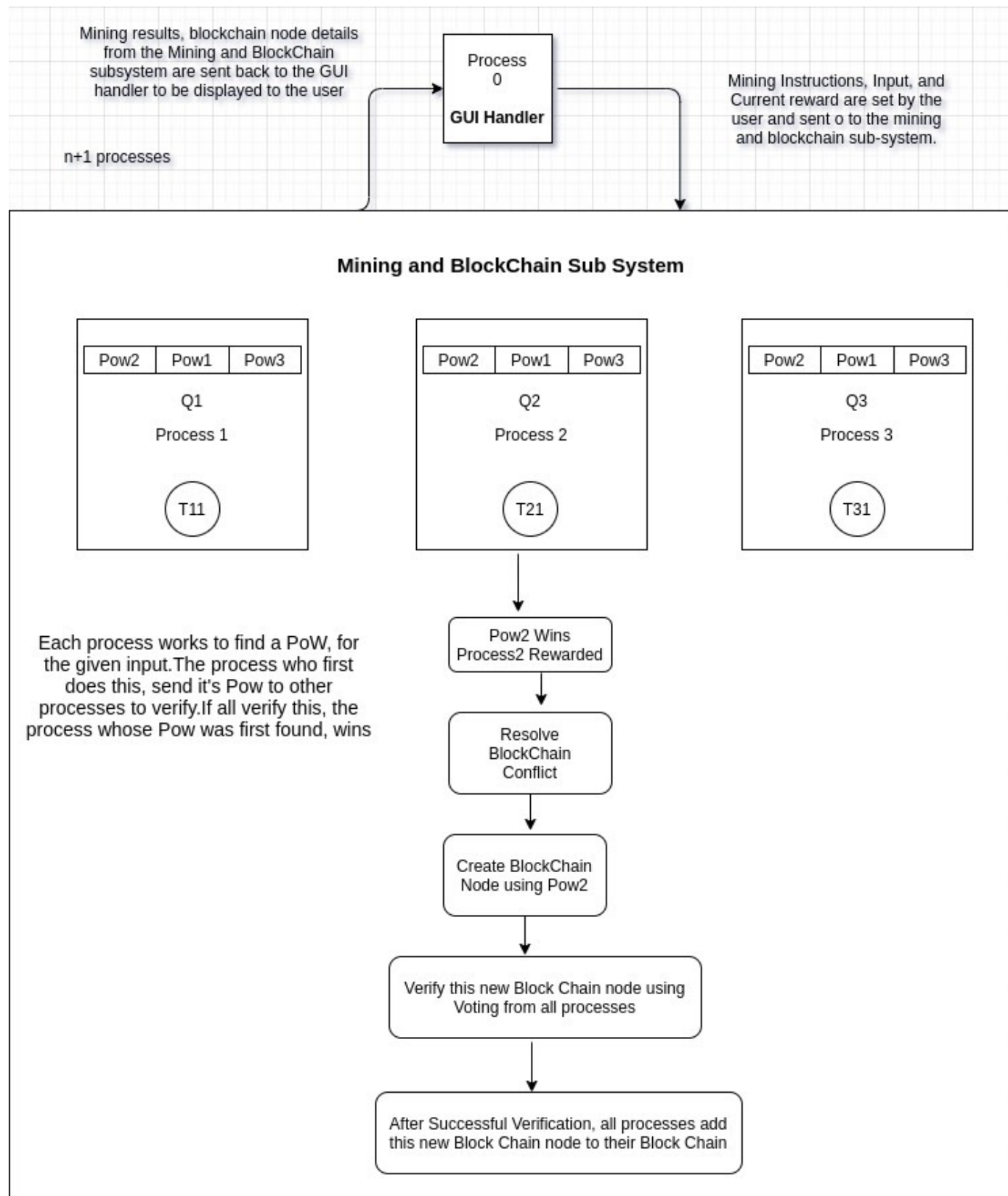
An electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers.

In this paper, a solution is proposed to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions.

The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

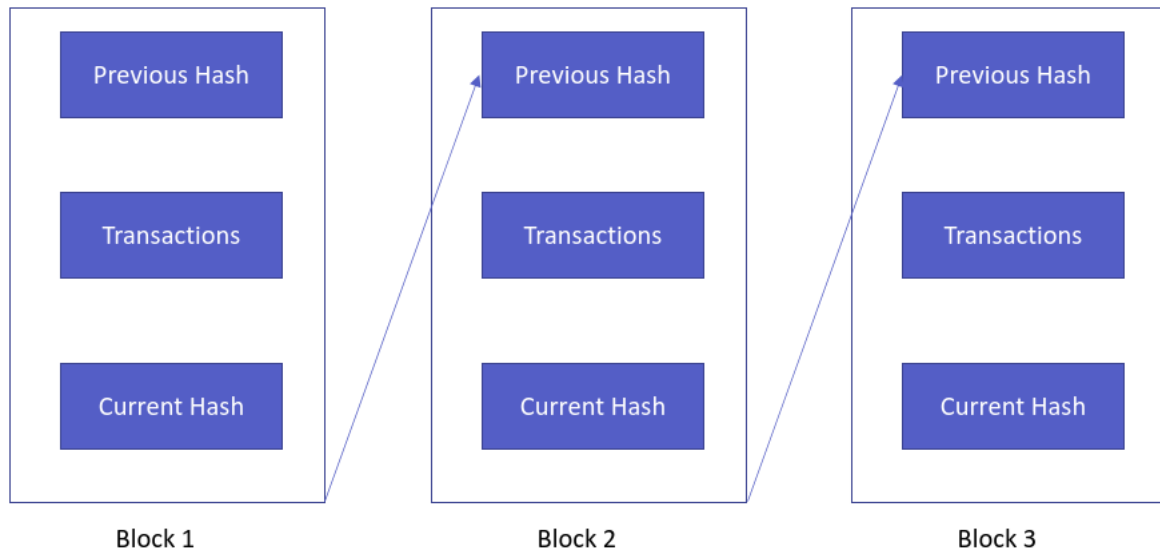
## 3 Our Approach

### 3.1 Flow Diagram



### 3.2 Transactions

In our project, Transaction is an input string provided for mining. This input string is provided at the first time. Based on this input Proof of work is solved. This input string is added as a transaction in the block.

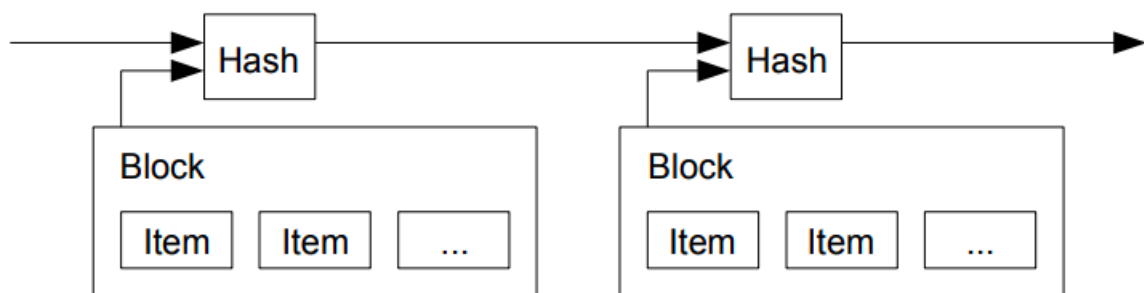


### 3.3 Timestamp Server

A timestamp server works by taking a hash of a block of items to be time stamped and widely publishing the hash, such as in a newspaper or Usenet post.

The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash.

Each timestamp includes the previous timestamp block's hash in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

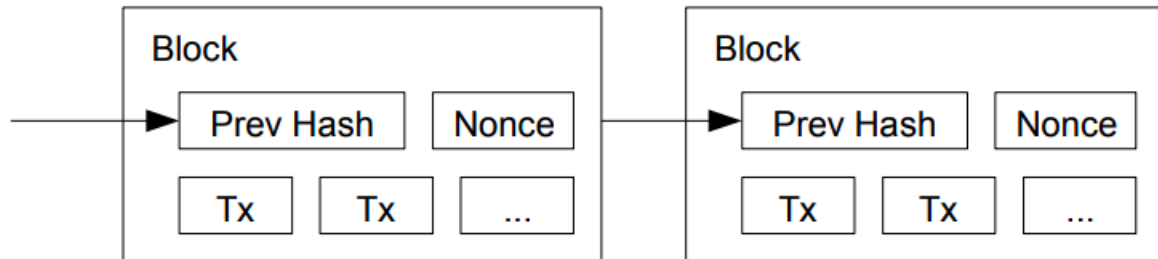


### 3.4 Proof of work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits.

On a peer-to-peer basis, we built a distributed timestamp server. The proof-of-work entails looking for a value that leaves a path of zero bits when hashed, such as with SHA-256. The

average amount of work required is proportional to the number of zero bits required, which can be checked with a single hash. We enforce the proof-of-work in our timestamp network by incrementing a nonce in the block until a value is reached that gives the block's hash the necessary zero bits. The block cannot be modified without redoing the work until the CPU effort has been spent to fulfil the proof-of-work.



### 3.5 Difficulty level

The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.

To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes.

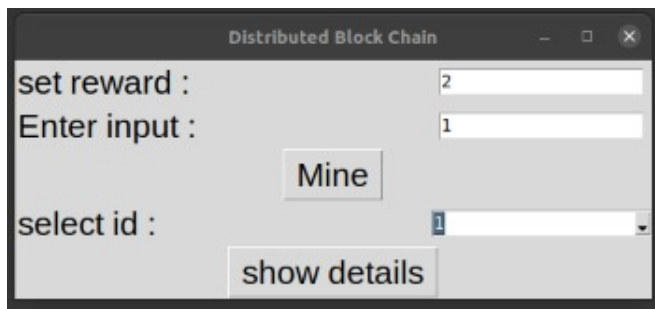
## 4 Features

### 4.1 Network

The steps to run the network are as follows:

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a difficult proof-of-work for its block.
4. When a node finds a proof-of-work, it broadcasts the proof of work to all nodes.
5. After successfully verifying proof of work, Every node resolves the conflict of chain among them.
6. Resolve conflicts ensures that every node has updated chain in the network.
7. After resolve conflict, the winner node sends the block to all the nodes.
8. Nodes accept the block only if the block is successfully verified.
9. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

## 4.2 User Interface



The screenshot shows a window titled "Distributed Block Chain". It has three input fields: "set reward :" with the value "2", "Enter input :" with the value "1", and "select id :" with the value "1". There are two buttons: "Mine" and "show details".



The screenshot shows a window titled "Miner Details". It contains a table with the following data:

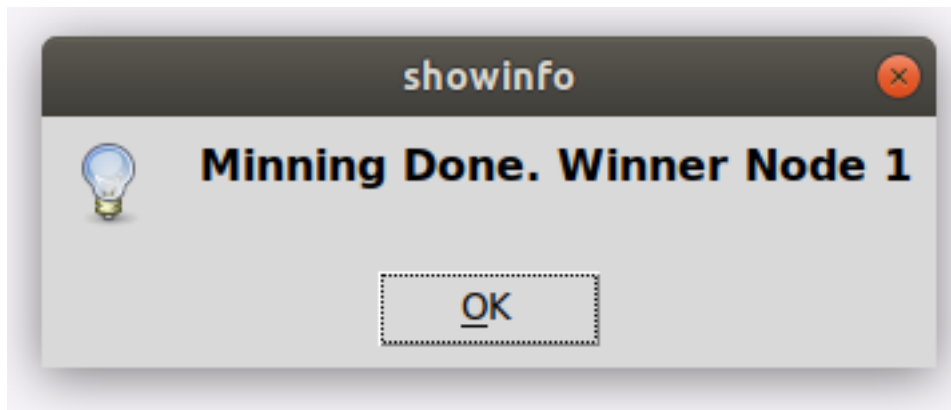
ID	Total Blocks	Timestamp	Merkle Tree	Last PoW	Hash of Block	Reward
1	3	2021-04-24 09:30:10.930690	2215e8ac4e2b871c2a48189e	581320	0846ba2e771648dddebb9ae6	1

### 4.2.1 Fields Description:

1. Set rewards: This field sets the incentive for the winning node.
2. Input String: This is where input string (transaction) is provided for mining.
3. Mine: This button start the mining process.
4. Select ID : Select the node ID of which you want to see the last block's details.
5. Show Details: This button shows the details of the last block mined of the selected node.

### 4.2.2 Miner Details:

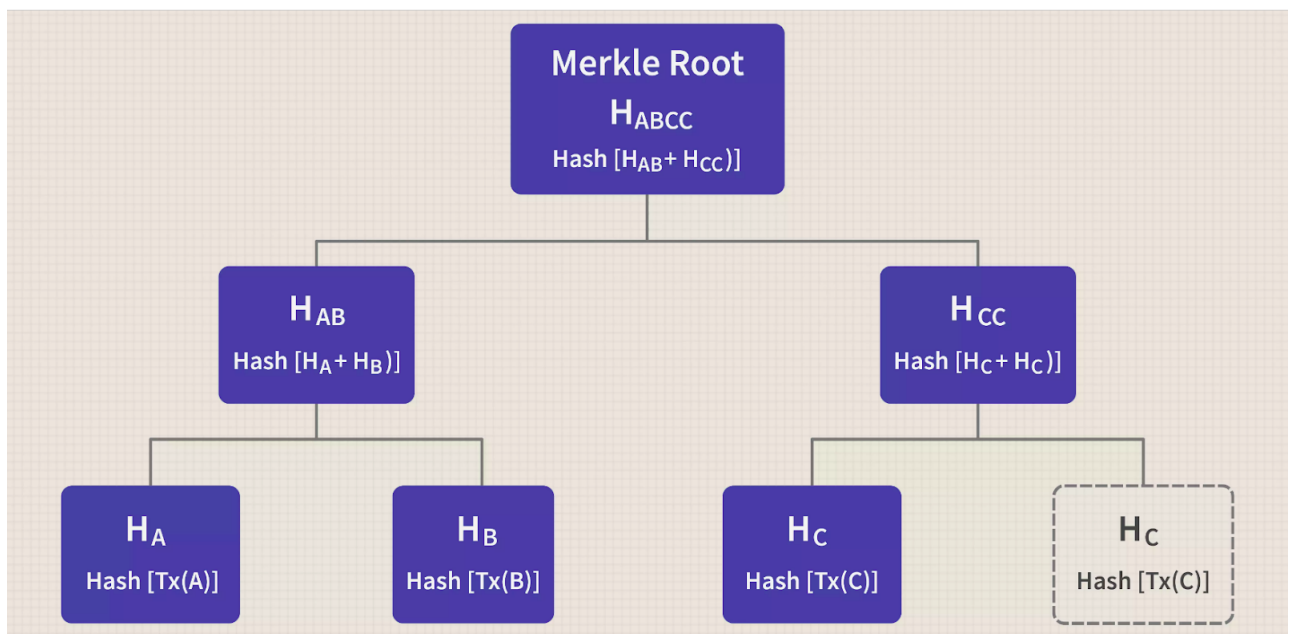
1. ID: ID of the node of which you want to see the details.
2. Total Blocks: Length of the blockchain.
3. TimeStamp: The time when block is created.
4. Merkle Tree: The root of the merkle tree.
5. Last PoW: Last successfully solved proof of work.
6. Hash of Block: Hash of the current block.
7. Reward: Reward if any for the node.



When mining is done. User gets to see the message and the winning node.

### 4.3 Merkle Tree

Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree, with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes does not need to be stored.



## 4.4 Block Structure:

typically blockchain block contains following fields:-

```
{
  'header':
  {
    'block_version':len(self.chain) + 1,
    'previous_hash':previous_hash,
    'timestamp': timestamp,
    'proof':data['proof'],
    'merkle_tree':root.merkle_root
  },
  'payload':
  {
    'currentHash':current_hash,
    'nodeID':data['nodeID'],
    'transaction':data['data']
  }
}
```

1. "block\_version" : Version of block used
2. "previous\_has" : Hash of previous block,
3. "proof" : Proof of work generated by a node
4. "merkle\_tree" : Merkle tree root
5. "current\_hash" : Hash value of current block
6. "node\_id" : Rank of the Process
7. "transaction" : The input provided for mining.

## 4.5 Blockchain:

1. A blockchain database is a special kind of database.
2. It differs from a traditional database in the way data is stored: blockchains store data in blocks, which are then chained together.
3. As new information is received, it is entered into a new block. If the block has been loaded with data, it is chained onto the previous block, resulting in a chronological sequence of data.
4. A blockchain can store a variety of data, but the most popular use so far has been as a transaction ledger.
5. In the case of Bitcoin, blockchain is used in a decentralised manner, meaning that no one individual or organisation has authority, but rather all users jointly do.
6. Decentralized blockchains are static, meaning that the data reached cannot be changed. This ensures the transactions in Bitcoin are forever registered and accessible to everyone.



## 4.6 Resolve Conflicts:

Due to node failure or communication breakdown or channel failure, it might be possible that any node can't add a block to its chain. In that case the length of chain on different nodes can be different. So when before adding a node to the chain, every node broadcasts its chain length to every node and gathers chain length from every other node. Then the node with maximum chain length sends its chain to the nodes which have chain length less than it. The nodes who have chain length less than maximum chain length in the network receive the chain from the maximum-chain-length node and verify the chain. After successful verification, the old chain gets replaced with the received chain. And the block gets added at the end of the new chain.

## 4.7 Reward:

When one nodes solves the proof of work at first and it gets verified by all the other nodes. The winning nodes gets a reward for its work. This reward can be set from the UI before the start of the mining.

## 5 Scope Of Improvement:

Points such as

1. Wallet creation facility and ability to send and receive money/coins.
2. Scaling the architecture to suit a highly distributed environment where a large number of GPUs can be used for faster parallel computation.
3. Making use of digital signature to add more security features.

## 6 Video Link

[https://iitaphyd-my.sharepoint.com/:f:/g/personal/aayush\\_upadhyaya\\_students\\_iit\\_ac\\_in/EqVlNzDdOQROtRk6YfJ\\_uqMBslTvilQhenc\\_yLiuPmCrWg?e=Zcrolc](https://iitaphyd-my.sharepoint.com/:f:/g/personal/aayush_upadhyaya_students_iit_ac_in/EqVlNzDdOQROtRk6YfJ_uqMBslTvilQhenc_yLiuPmCrWg?e=Zcrolc)

## 7 References:

<https://bitcoin.org/bitcoin.pdf>

## 8 Team Members

1. Indranil Pradhan(2019202008)
2. Aayush Upadhyaya(2019202010)
3. Anurag Pateriya(2019201057)
4. Himanshu Mandhan(2019201094)
5. Jeevesh Kataria(2019201058)
6. Vishal Malhotra(2019201018)