Blockchain

Trần Văn Quý

Bibliography

Andreas M. Antonopoulos.(2017). Mastering Bitcoin.

Imran Bashir .(2018). Mastering Blockchain.

https://www.udemy.com

Main goal of the course

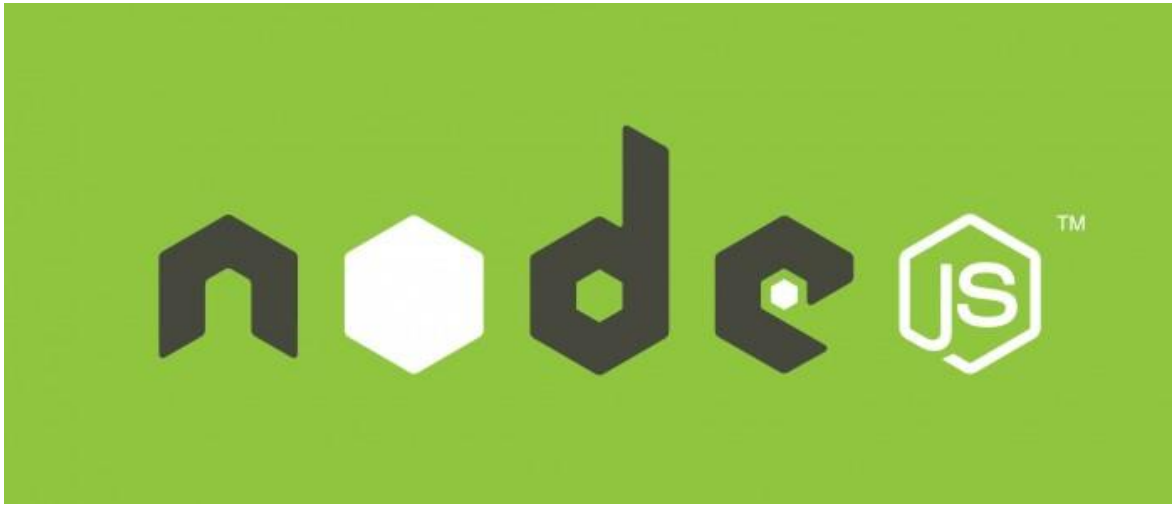
- Build a blockchain and cryptocurrency to understand these technologies.
- Learn by building

Course roadmap

- Code the course blockchain.
- Build an API around the blockchain.
- Create a dynamic peer-to-peer server for multiple contributors.
- Implement a proof-of-work system.
- Create an transaction system for a cryptocurrency.

Programming language

Uses NodeJS or any programming languages for the project.

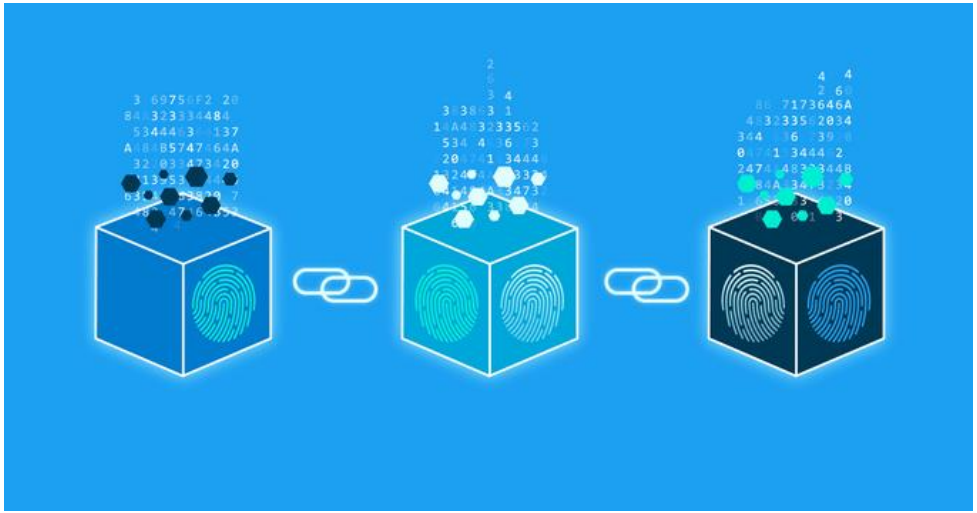


Motivation

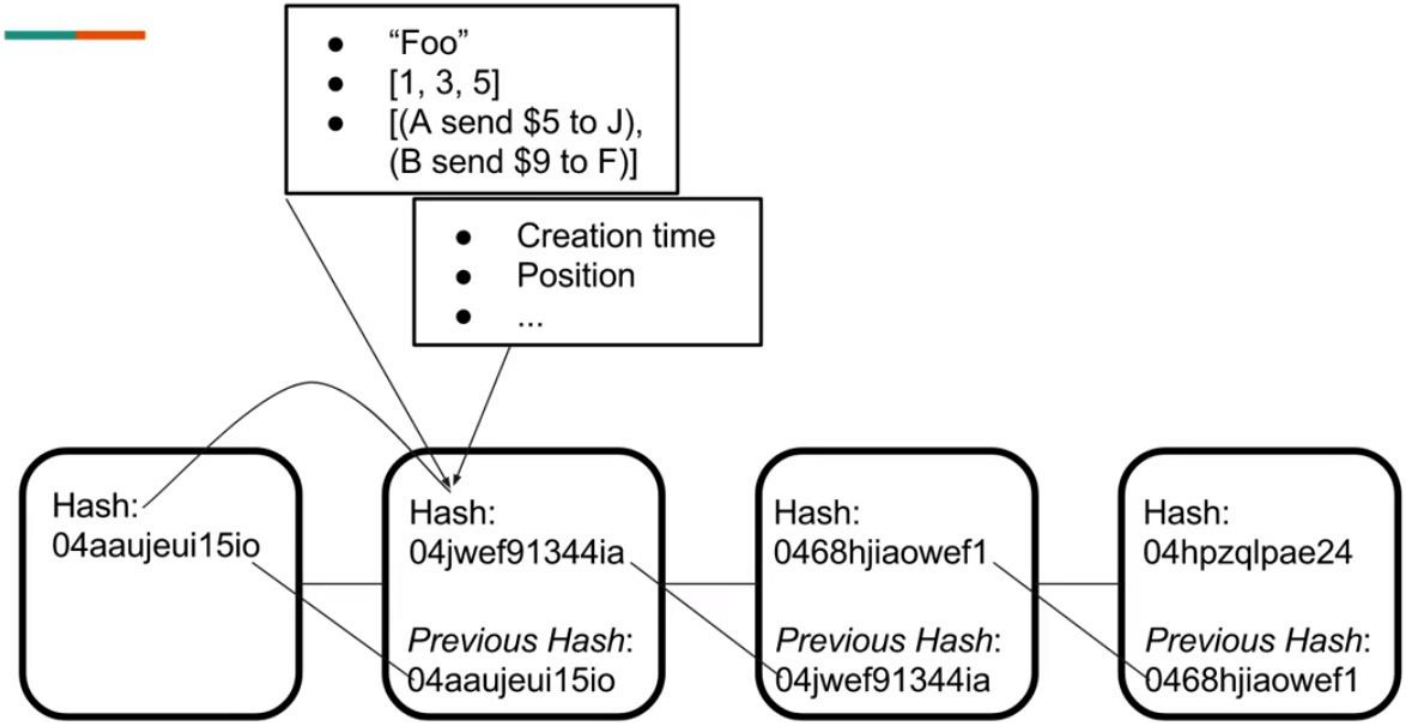
- The exponentially-growing need for blockchain engineers.
- Build blockchain yourself, and demonstrate understanding.
- It’s fascinating and fun.

What is the Blockchain?

The blockchain is a distributed and decentralized ledger that stores data such as transactions between individuals and that ledger is publicly shared across all the nodes of its network.

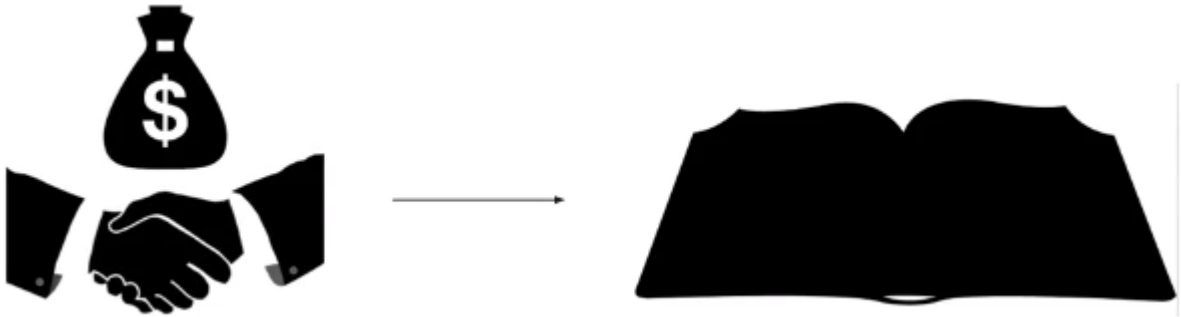


What is the Blockchain?

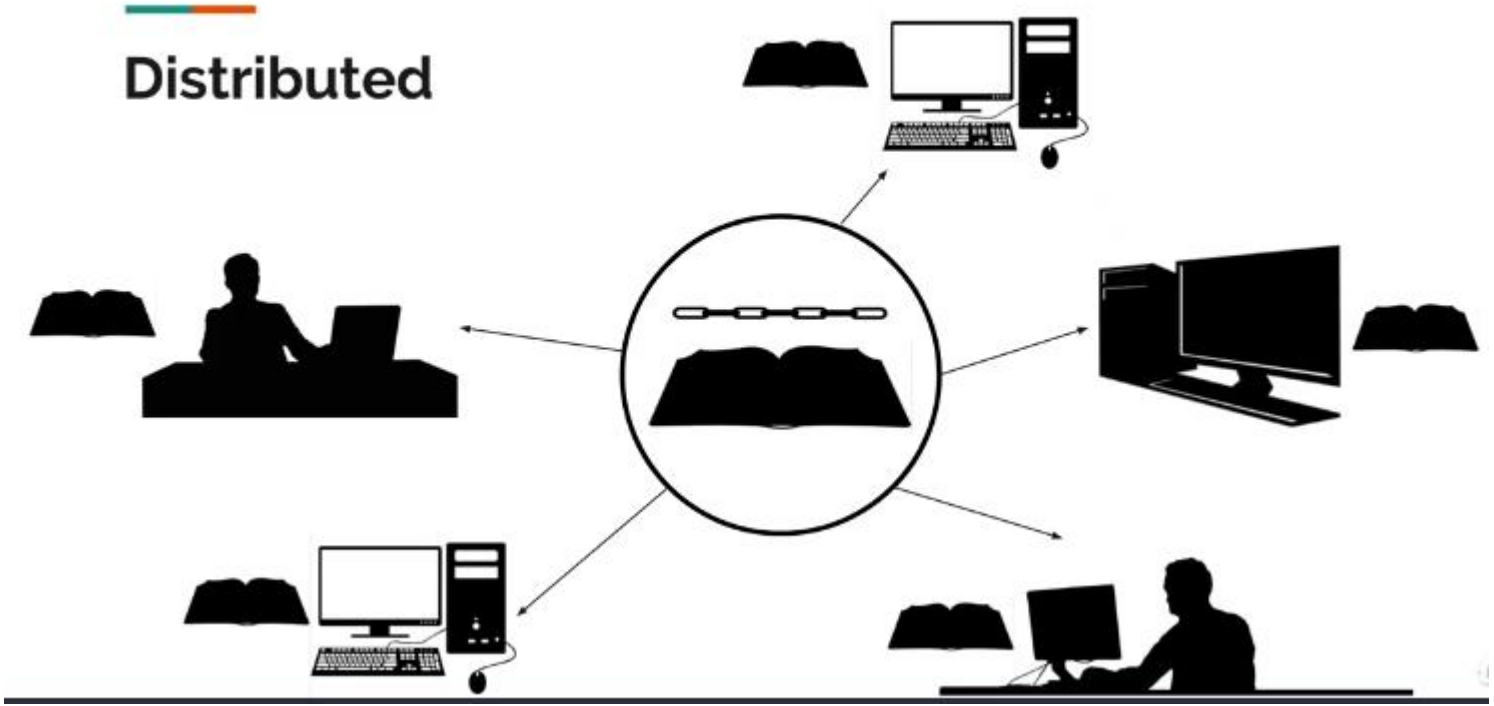


Ledger

A ledger is a record keeping book that records all the transactions of an organization.



Distributed



Centralized vs Decentralized

Centralized	Decentralized
● Only entity records the data ● The central entity has a lot of power ● Full authority to fine or reward ● Complete trust with the entity	● Everyone records the data ● Everyone has equal power ● Fair and transparent system ● Trustless

Why use the Blockchain?

- Decentralization leads to a trustless system.
- No middle men and no fees.
- High secure and no central point of failure.
- Dependable data.

The Blockchain in Practice

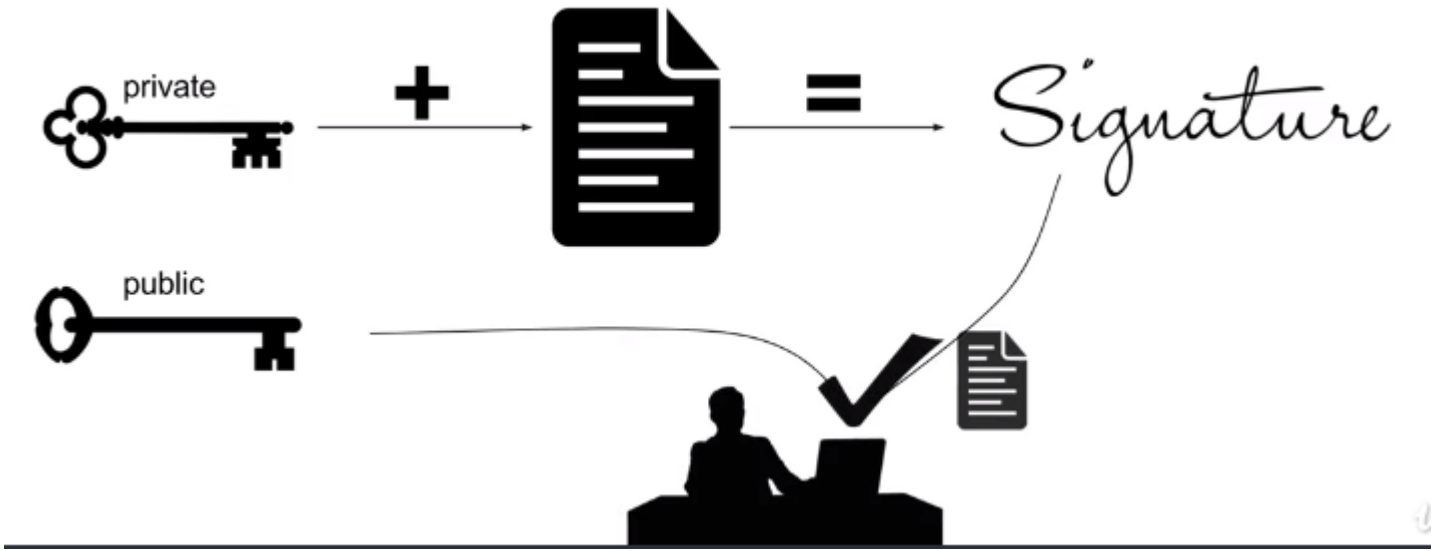
Cryptocurrency

- A cryptocurrency is a digital medium of exchange.
- It has three main features: a secure blockchain, wallets, and mining.

Leverages the blockchain

- How is this secure?
- Uses cryptography to generate digital signatures.

Digital Signatures



Digital Signatures

```
const EC = new ec('secp256k1');

const generatePrivateKey = (): string => {
    const keyPair = EC.genKeyPair();
    const privateKey = keyPair.getPrivate();
    return privateKey.toString(16);
};

const getPublicFromWallet = (): string => {
    const privateKey = getPrivateFromWallet();
    const key = EC.keyFromPrivate(privateKey, 'hex');
    return key.getPublic().encode('hex');
};
```

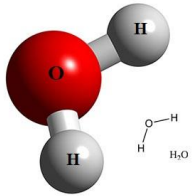
Digital Signatures

```
const signTxIn = (transaction: Transaction, txInIndex: number,
    privateKey: string, aUnspentTxOuts: UnspentTxOut[]): string => {
    const key = ec.keyFromPrivate(privateKey, 'hex');
    const signature: string = toHexString(key.sign(dataToSign).toDER());

    return signature;
};

const validateTxIn = (txIn: TxIn, transaction: Transaction, aUnspentTxOuts: Unsp
    const key = ec.keyFromPublic(address, 'hex');
    const validSignature: boolean = key.verify(transaction.id, txIn.signature);
    if (!validSignature)
        return false;
    return true;
};
```

Digital Signatures

- ~10^{77}

- ~10^{47}



- ~10^{18}

Wallets

- Objects that store the private and public key of an individual.
- The public key is the address of the wallet.
- Help sign transactions.

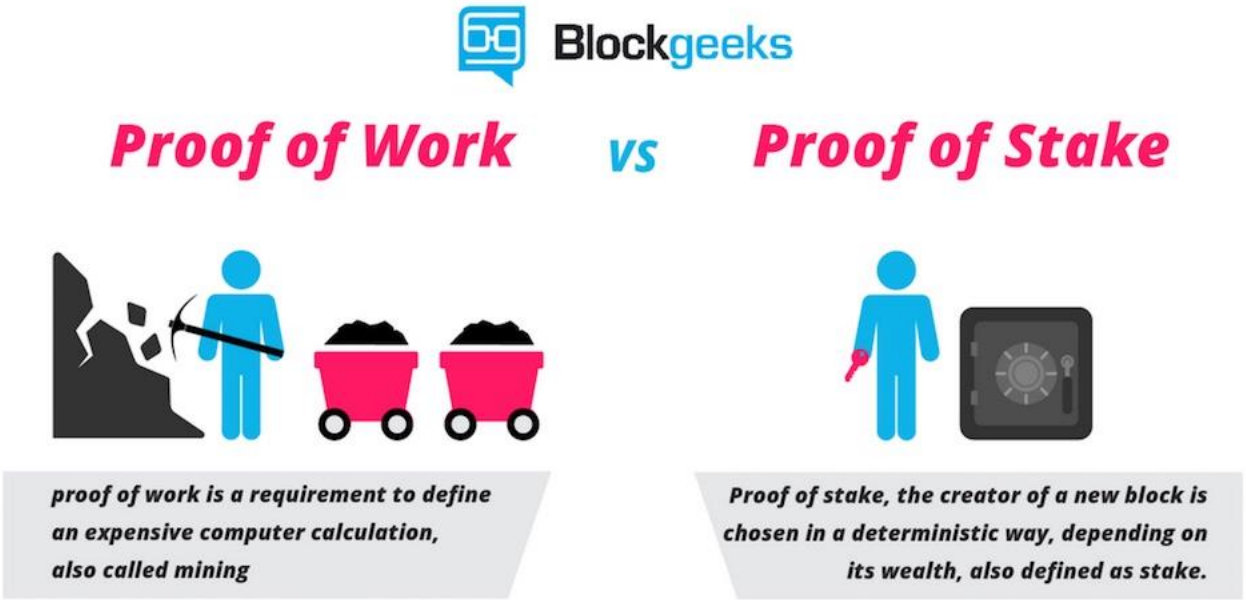
Mining

- Transactions are temporarily “unconfirmed”.
- Include blocks of transactions by solving a “proof of work”
  - Difficult to solve, and computationally expensive.
  - One solved, the miner can add the block and other miners will verify.
  - Miners are rewarded for adding a block to the chain.
  - The difficulty can adjust to control the rate of new blocks coming in.

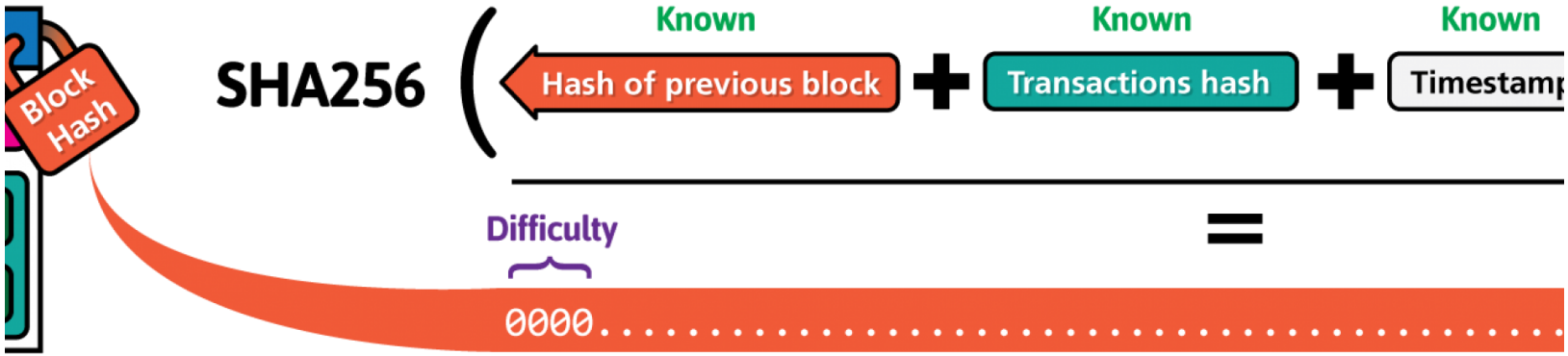
Consensus Algorithms

- Proof of Work (PoW)
- Proof of Stake (PoS)
- Delegated Proof of Stake (DPoS)
- Proof of Elapsed Time (PoET)
- Proof of Deposit (PoD)
- Proof of Importance (PoI)
- Proof of Activity (PoA)
- Proof of Capacity (PoC)
- Proof of Storage (PoS)

Consensus Algorithms



Proof-of-Work (PoW)



Proof-of-Work (PoW)

```
const findBlock = (index: number, previousHash: string, timestamp:
    let nonce = 0;
    while (true) {
        const hash: string = calculateHash(index, previousHash
            , timestamp, data, difficulty, nonce);

        if (hashMatchesDifficulty(hash, difficulty)) {
            return new Block(index, hash, previousHash
                , timestamp, data, difficulty, nonce);
        }
        nonce++;
    }
};
```

BITCOIN

- The first decentralized cryptocurrency in 2009.
- Great growth, and widespread adoption.

ethereum

ripple

IOTA

litecoin

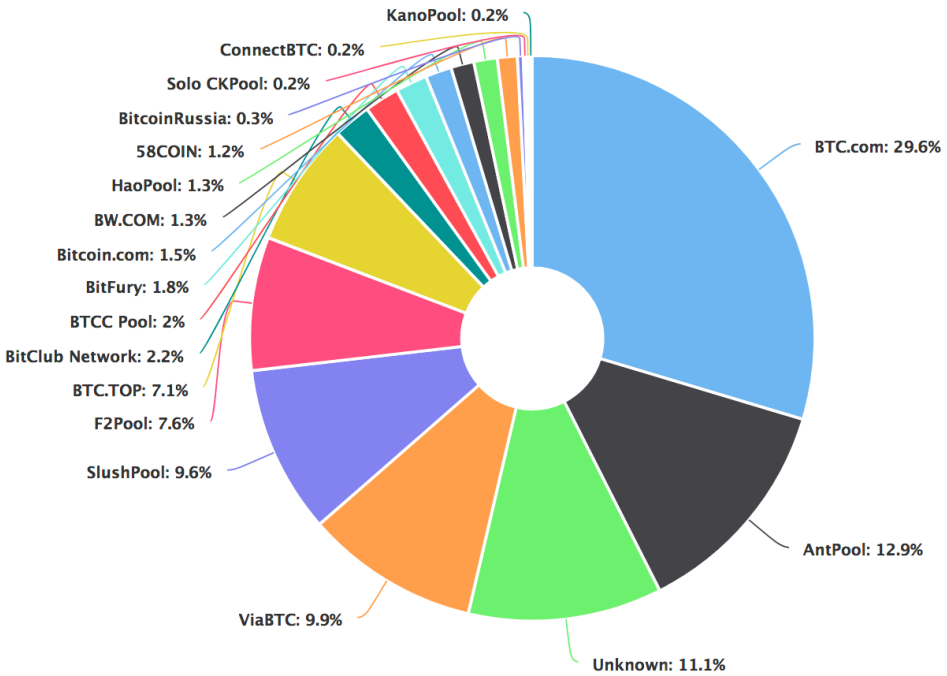
BITCOIN


Bitcoin Price in USD
from Jan. 2017 to Feb. 2018
$20k
$15k
$10k
$5k
0
J
F
M
A
M
J
J
A
S
O
N
D
J
F

Mining


1800W Switching Power Supply
- 90% High Efficiency
-Designed for Mining Machine
180V-260V

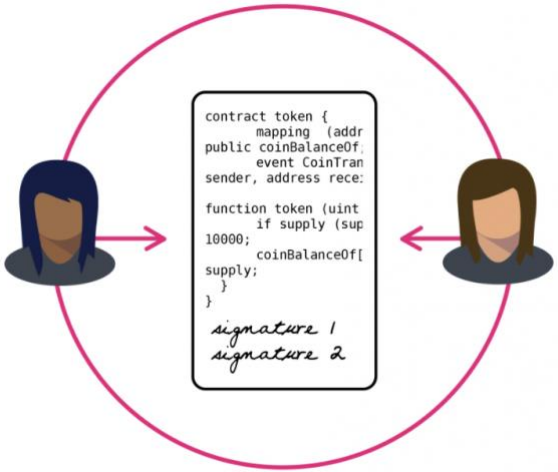
Mining

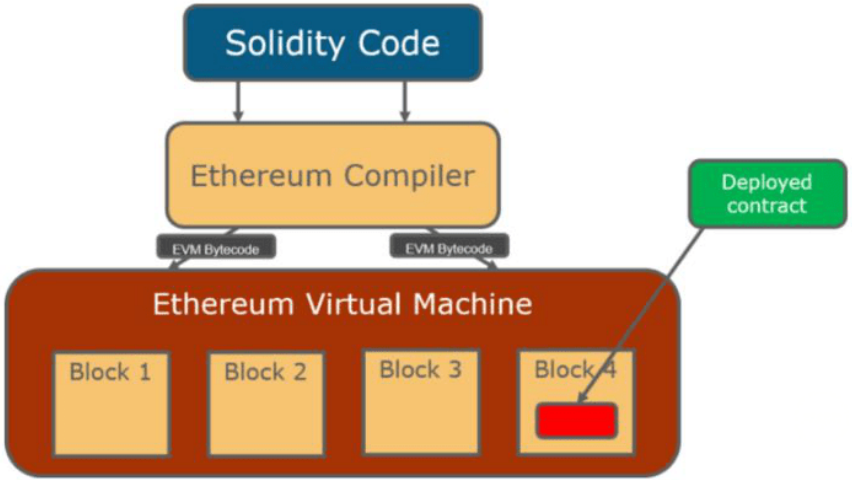

KanoPool: 0.2%
ConnectBTC: 0.2%
Solo CKPool: 0.2%
BitcoinRussia: 0.3%
58COIN: 1.2%
HaoPool: 1.3%
BW.COM: 1.3%
Bitcoin.com: 1.5%
BitFury: 1.8%
BTCC Pool: 2%
BitClub Network: 2.2%
BTC.TOP: 7.1%
F2Pool: 7.6%
SlushPool: 9.6%
ViaBTC: 9.9%
Unknown: 11.1%
AntPool: 12.9%
BTC.com: 29.6%

ETHEREUM

- Ethereum is a **decentralized platform that runs smart contracts**: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference

ETHEREUM


contract token {
mapping (addr
public coinBalanceOf;
event CoinTran
sender, address rece:
function token (uint
if supply (sup
10000;
coinBalanceOf[
supply;
}
}
signature 1
signature 2


Solidity Code
Ethereum Compiler
Deployed contract
EVM Bytecode
EVM Bytecode
Ethereum Virtual Machine
Block 1
Block 2
Block 3
Block 4

Block Structure

Block

- Timestamp in milliseconds.
- previvousHash – the hash of block before it.
- hash – based on its own data.
- The data to store

Block

```
class Block {
    constructor(index, previousHash, timestamp, data, hash) {
        this.index = index;
        this.previousHash = previousHash.toString();
        this.timestamp = timestamp;
        this.data = data;
        this.hash = hash.toString();
    }
}
```

Data Hash

```
var CryptoJS = require("crypto-js");
var calculateHash = (index, previousHash, timestamp, data) =>
{
    return CryptoJS.SHA256(index + previousHash
        + timestamp + data).toString();
};
```

Generating a block

```
var generateNextBlock = (blockData) => {
    var previousBlock = getLatestBlock();
    var nextIndex = previousBlock.index + 1;
    var nextTimestamp = new Date().getTime() / 1000;
    var nextHash = calculateHash(nextIndex, previousBlock.hash
        , nextTimestamp, blockData);
    return new Block(nextIndex, previousBlock.hash
        , nextTimestamp, blockData, nextHash);
};
```

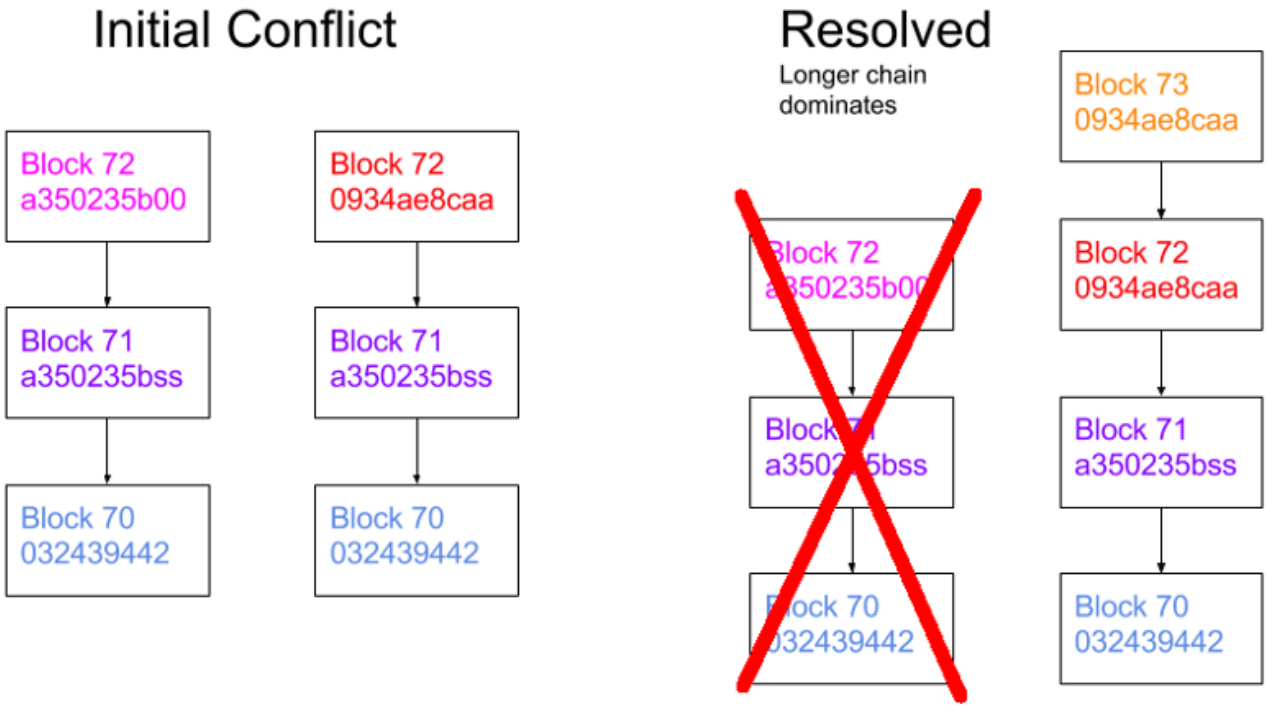
Genesis block

```
var getGenesisBlock = () => {
    return new Block(0, "0", 1465154705, "my genesis block!!"
   , "816534932c2b7154836da6afc367695e6337db8a921823784c14378abed4f7d7");
};

var blockchain = [getGenesisBlock()];
```

Validating the integrity of blocks

```
var isValidNewBlock = (newBlock, previousBlock) => {
    if (previousBlock.index + 1 !== newBlock.index) {
        console.log('invalid index');
        return false;
    } else if (previousBlock.hash !== newBlock.previousHash) {
        console.log('invalid previoushash');
        return false;
    } else if (calculateHashForBlock(newBlock) !== newBlock.hash) {
        console.log('invalid hash');
        return false;
    }
    return true;
};
```

Choosing the longest chain



Replace Chain

```
var replaceChain = (newBlocks) => {
    if (isValidChain(newBlocks)
        && newBlocks.length > blockchain.length)
    {
        console.log('Received blockchain is valid');
        blockchain = newBlocks;
        broadcast(responseLatestMsg());
    }
    else {
        console.log('Received blockchain invalid');
    }
};
```

Valid Chain

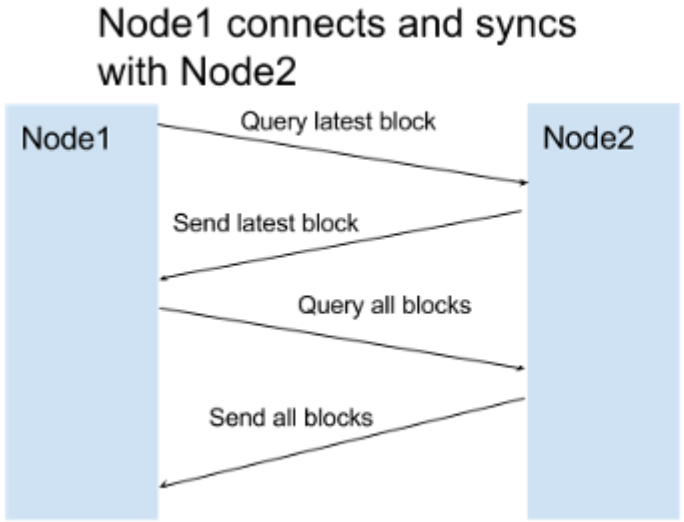
```
var isValidChain = (blockchainToValidate) => {
    if (JSON.stringify(blockchainToValidate[0]) !==
        JSON.stringify(getGenesisBlock())){
        return false;
    }
    var tempBlocks = [blockchainToValidate[0]];
    for (var i = 1; i < blockchainToValidate.length; i++) {
        if (isValidNewBlock(blockchainToValidate[i]
            , tempBlocks[i - 1])){
            tempBlocks.push(blockchainToValidate[i]);
        }
        else {
            return false;
        }
    }
    return true;
};
```

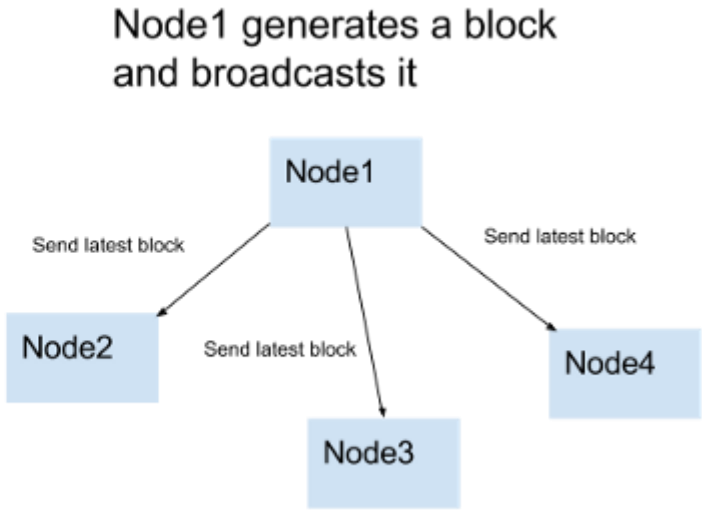
Communicating with other nodes

An essential part of a node is to share and sync the blockchain with other nodes. The following rules are used to keep the network in sync.

- When a node generates a new block, it broadcasts it to the network
- When a node connects to a new peer it querys for the latest block
- When a node encounters a block that has an index larger than the current known block, it either adds the block the its current chain or querys for the full blockchain.

Communicating with other nodes





Q & A