

# 100 Days of Code

When tweeting please tag the following account: @OAN\_OpenApps and use hashtags: #openapps and #100daysofcode.

## Day 1 & 2

### Module 1: Blockchain Fundamentals

#### 1. Blockchain concept and context :

##### i) How did it all start?

The first blockchain started as a research project. Satoshi Nakamoto wrote the paper Bitcoin: 'A Peer to Peer Electronic Cash System in 2009' in which he described a way of transferring value from one person to another without the need for physical currency or bank. Bitcoin itself is a very simple blockchain. At its core, it's just a list of accounts and the value held within them.

##### ii) How did it improve?

In 2014 people realized that blockchains could be used to run small programs and a way to transfer money safely.

The Bitcoin protocol isn't particularly well equipped for programs, so in 2015 Vitalik Buterin co-created Ethereum which allowed developers to create *smart contracts* on a blockchain. Ethereum vastly improved the usability of blockchains and attracted the attention of companies like Microsoft, Bank of America, and Google.

##### iii) So what exactly is blockchain?

Basically a blockchain is a "cryptographically secure transactional singleton machine with shared-state".

Let's break into three pieces and take cryptocurrency as example :

i) **Singleton machine** :- Single global truth that everyone believes in.

(Ex: Global Truth- A valid transaction verified by mechanisms accepted by everyone)

ii) “With shared-state” means that the state stored on this machine is shared and open to everyone.

(Ex: A valid transaction obtained from singleton machine is stored in all the members of blockchain)

### iii) Cryptographically Secure :

It means that the creation of digital currency is secured by complex mathematical algorithms that are obscenely hard to break.

(Ex: A valid transaction obtained from singleton machine and is stored in all the members of blockchain has a feature of immutability i.e the users who have the data stored with them cannot change them at any cost )

Think of a firewall of sorts. They make it nearly impossible to cheat the system.

(e.g. create fake transactions, erase transactions, etc.)

In a nutshell, A information that is universally accepted/Verified (Singleton Machine) is broadcasted to all the users and every user stores this information with them(Shared state) , and this stored info cannot be manipulated at any cost (Cryptographically Secure)

### iv) Blockchain vs Centralized server:

Blockchain networks look like regular networks, with a group of computers all connected. However, they differ in the way that programs and applications run. On a regular network when a user wants to find the results of an application, they send the request to a particular server who runs a program and returns the answer to the user. An issue with this process is that there isn't any way to verify that the response the user got from the server is correct, or that the program the server ran was the program the user requested.

On a blockchain network when a request is sent from the user to the network, all the computers run the program and agree on the answer before sending it back to the

user. This way the user can verify that the answer is the correct answer and that the correct program was run in the first place. Calling a program is referred to as a *transaction*. All transactions are saved in a database and shared with all the other computers on the blockchain network. Because of this, users can prove that they performed a certain action.

A layman's introduction to blockchain by Jaspreet Bindra, Advisor at Blockchain India & SVP at Mahindra Group

[https://www.youtube.com/watch?v=8fbhl1qVj0c&ab\\_channel=TEDxTalks](https://www.youtube.com/watch?v=8fbhl1qVj0c&ab_channel=TEDxTalks)

## 2.How does a distributed ledger work?

[https://www.youtube.com/watch?v=SSo\\_ElwHSd4&ab\\_channel=SimplyExplained](https://www.youtube.com/watch?v=SSo_ElwHSd4&ab_channel=SimplyExplained)

**Def:** A **distributed ledger** is a database that is consensually shared and synchronized across multiple sites, institutions, or geographies, accessible by multiple people.

Let's split the definition to understand better using a fictional cryptocurrency as

### **Example:**

**Ledger** - A ledger can be considered as a digital register/Database which can store any content. Assume a new transaction detail "Trump paid 10 paise to Kim jung Un" is stored in the digital database or 'ledger' present in north korea.

Assume countries like India, China, Northkorea, and the United States also have ledgers with them. To achieve 'distributed' status all the above countries copies/ syncs the same transaction - 'Trump paid 10 paise to Kim jung Un' into their ledgers.

**Synchronized ledger** - Whenever any new transaction is made, that transaction details replicated to all the ledgers of countries India, China, United states and NorthKorea

(Note: new transaction - 'Kim paid 5 paisa to Trump'..etc , replication achieves synchronization with others)

## Why synchronization?

If all the ledgers belonging to countries are synchronized and if someone manipulates the data in their ledger, We can easily find out that black sheep by comparing with other synchronized ledgers.

**Consensually shared-** A new transaction generated is called valid transaction only if it meets certain rules, and these rules are universally accepted. Every one who is generating the transaction need to follow certain rules like sender must be verified and receiver must be verified (Such that north korea cannot generate whatever it wants and say it as a transaction).

Hence the transaction which abided by the universal rule and is shared/Replicated within all the ledgers constitutes a distributed ledger.

But wait! How can we ensure that the person who is writing the transaction is following the universally acceptable rule?

You might say that as it is universally acceptable , so we can ask other countries whether the particular transaction is valid or not. If a country didn't follow the universal rule the validity of other countries can help reject the transaction.

You must have heard about DDOS attack i.e distributed denial of service , generally this occurs when millions of fake requests generated by malicious bots hit the website and the website loses its control and crashes In this case the genuine requests cannot survive as the fake requests have destroyed the System Now if we don't want that to repeat in our use case , let us punish malicious dude !

In this case each country needs to put some of their valuable men at stake to prove their

Seriousness, i.e if they made a faulty transaction , UN can jail those valuable men they kept at stake !

The below link has awesome explanation of whole blockchain:-

<https://hackernoon.com/wtf-is-the-blockchain-1da89ba19348>

## Can we do the same thing in bitcoin?

Lets enter Mining!

The purpose of mining is probably a little confusing at first. I will keep the Bitcoin blockchain as an example throughout this explanation. Mining is NOT about creating new bitcoins. Mining is the mechanism that allows the blockchain to be a decentralized and secured ledger. It secures the bitcoin system and enables a system without a central authority. Do not confuse the rewards given to miners (new bitcoin) with the process itself.

## Mining in Bitcoin

Miners validate new transactions and record them on the global ledger (blockchain). On average, a block (the structure containing transactions) is *mined* every 10 minutes. Miners compete to solve a difficult mathematical problem based on a cryptographic hash algorithm. The solution found is called the *Proof-Of-Work*. This proof proves that a miner did spend a lot of time and resources to solve the problem. When a block is 'solved', the transactions contained are considered *confirmed*, and the bitcoin concerned in the transactions can be spent. So, if you receive some bitcoin on your wallet, it will take approximately 10 minutes for your transaction to be confirmed.

Miners receive a reward when they solve the complex mathematical problem. There are two types of rewards: new bitcoins or transaction fees. The amount of bitcoins created decreases every 4 years (every 210,000 blocks to be precise). Today, a newly created block creates 12.5 bitcoins. This number will keep going down until no more bitcoin will be issued. This will happen around 2140, when around 21 millions bitcoins will have been created. After this date, no more bitcoin will be issued.

Miners can also receive rewards in the form of transaction fees. The winning miner can 'keep the change' on the block's transactions. As the amount of bitcoin created with each block diminishes, the transactions fees received by the miner will

increase. After 2140, the winning miner will only receive transaction fees as his

Amount of bitcoin created with each mined block ( dates are indications )

Before November 2012	Nov 2012 - Jul 2016	Jul 2016 - 2020	2020 - 2024	2024 - 2028	2028 - 2032	2031 - 2036	2036 - 2040	2040 - 2044	2044 - 2048
50	25	12,5	6,25	3,125	1,5625	0,78125	0,390625	0,1953125	0,09765625

reward.

The amount of bitcoin issued with each block is divided by 2 every 210 000 blocks.  
So we can calculate the maximum amount of bitcoin with some code.

```
// First 210 000 blocks reward

const start_reward = 50

// The reward is modified every 210000 blocks

const reward_interval = 210000

const max_btc = () => {

// 50 BTC = 5000000000 Satoshis

// Satoshis are the smallest denomination in bitcoin

let current_reward = 50 * 10 ** 8

let total_btc = 0

while( current_reward > 0 ) {

total_btc += reward_interval * current_reward
```

```
current_reward = current_reward / 2

return total_btc

console.log(`The maximum amount of BTC created will be
max_btc() } Satoshis, or ${max_btc() / 10**8} BTC`)

// The maximum amount of BTC created will be 21000000000000000
Satoshis, or 21000000 BTC
```

So, yeah, 21 millions will be probably the maximum amount of bitcoin issued in the end!

### How does it work, anyway?

The question is, how can everyone in the network agree on an universal 'truth' about bitcoin ownership, without having to trust anyone in said network? The blockchain is not created or managed by any central authority. Every node has access to the public ledger of transactions that acts as an authoritative record. Somehow, every node in the network, acting on those insecure informations, come up to the same conclusions and are able to replicate the same ledger. Let's try to explore how this works.

It will probably help if we take a real life example. Meet block [#502426](#). We'll follow the lifecycle of this block from its construction to its final validation. Let's say the winning miner was called Joe.

### The previous block

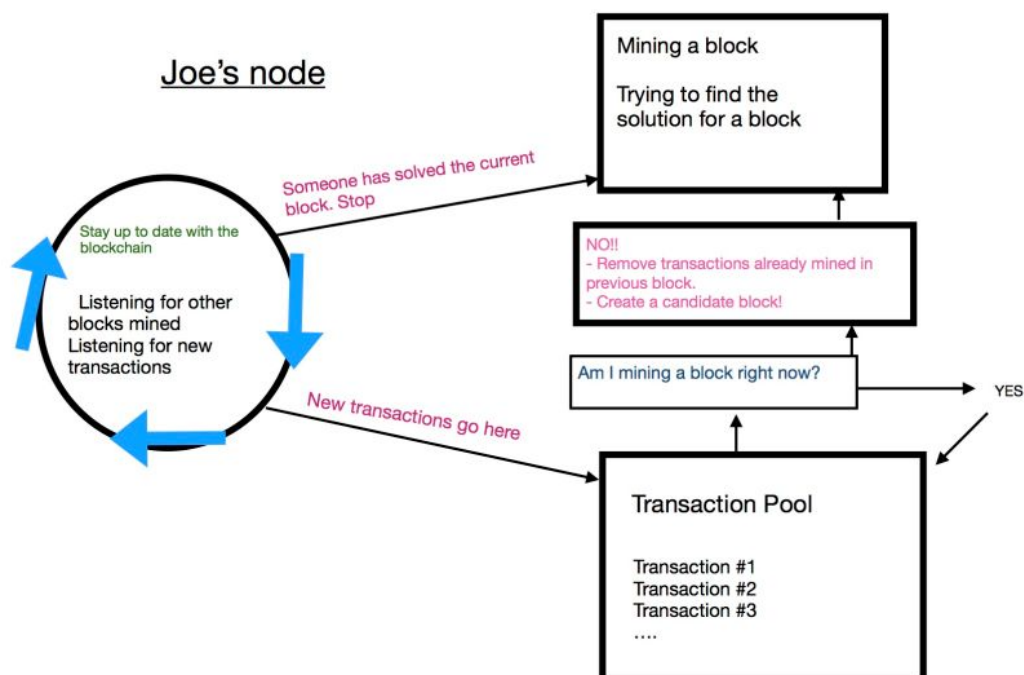
In the world of Bitcoin, it takes approximately 10 minutes to validate a new block. Our miner Joe was competing to validate the block 502425, the previous one. Unfortunately, someone else solved the problem before him. But, the end of one block's competition means the beginning of a new one. As soon as the block

502425 was mined, Joe updated his local copy of the blockchain and started to create a *candidate block*, the block 502426. While Joe's computer ( or *node* ) was searching for the *Proof of Work* for the previous block, it was also listening for new transactions. Those new transactions are added to the *memory pool* or *transaction pool*. This is where transactions wait until they can be included in a new block and validated.

### Creating the candidate block

When Joe's node is notified that the current block has a valid Proof of Work, it starts constructing a candidate block by gathering the transactions in the transaction pool. It removes the transactions already present in the previous block, if there are any. The block is called a candidate block because it doesn't have a valid Proof of Work yet.

So, we can see that block #502426 has 3189 transactions inside it. This was the number of transactions present in Joe's transaction pool when he created his candidate block.





## Coinbase transaction

The first thing Joe's node does is creating the *coinbase transaction*. Very simply, this is his reward for mining the block. This transaction says => *Pay Joe's wallet address xxx BTC to reward him for finding a valid Proof of Work*. This transaction is different from the other ones because, as I explained earlier, the bitcoins in the reward are created from nothing. They do not come from someone's wallet. Joe's node also calculates the transaction fees in the block.


*Joe's reward = Reward for mining block + transactions fees*

In this case, we can see that the block reward is 12.5 BTC (*Block Reward in left column*) and the transaction fees are equal to 4.86507997 BTC (*Transaction fees in left column*).

$12.5 + 4.86507997 = 17.36507997 \text{ BTC}$

You can see the details of this transaction in the list below.

## Transactions

5a6085862133d3233ca2202eb31ab63366d2a9cf40dddfdb0cf1c45021bf7373		2018-01-03 21:12:39
No Inputs (Newly Generated Coins)	 18cBEMRxxHqz... (ViaBTC Bitcoin Mining Pool) Unable to decode output address	17.36507997 BTC 0 BTC
		17.36507997 BTC

You can see *No Inputs (Newly Generated Coins)*. Like I said, coinbase transactions do not come from anyone's wallet, so they can't have any inputs. You only have the winning miner's wallet address here.

## Constructing the block header

In a previous article, I described what was contained in [a block](#). Joe's node has the responsibility to create a proper block header for the block he is mining. In the [article](#), I mostly focused on the merkle tree, a summary of transactions and mentioned that there are three different sets of data in a block header: the previous

block hash, the merkle tree root and data for the mining competition. We'll dig deeper in the last one.

## Data fields

**This metadata set contains:**

- **The Version:** This is a version number to track the software and/or protocol upgrades.
- **Timestamp:** Seconds from Unix Epoch. When the block was created.
- **Target:** Proof of Work algorithm target for this block
- **Nonce:** Counter used for the Proof of Work algorithm

When the block #502426 was mined, the version number was 2. It becomes *0x20000000* when converted in little-endian format in 4 bytes. (*Version in the left column*)

Next, we get the 4-byte timestamp. This is the number of seconds elapsed since January 1, 1970. We see the timestamp for this block is *2018-01-03 21:12:39* (*Timestamp in left column*). If we convert this in seconds, we get *1515013959* seconds.

The target field defines the required Proof of Work to make this a valid block. To put it simply, the target is generated by the network and defines what makes a block's hash valid or not. If your hash is higher than the target, it is not valid. This is what is used to calculate the *difficulty*. In our block, the difficulty is *1,931,136,454,487.72*. Take a look at the block's hash:

**00000000000000000000000020c60222099aaebc6e7795784f74628ec640b223d3d339**

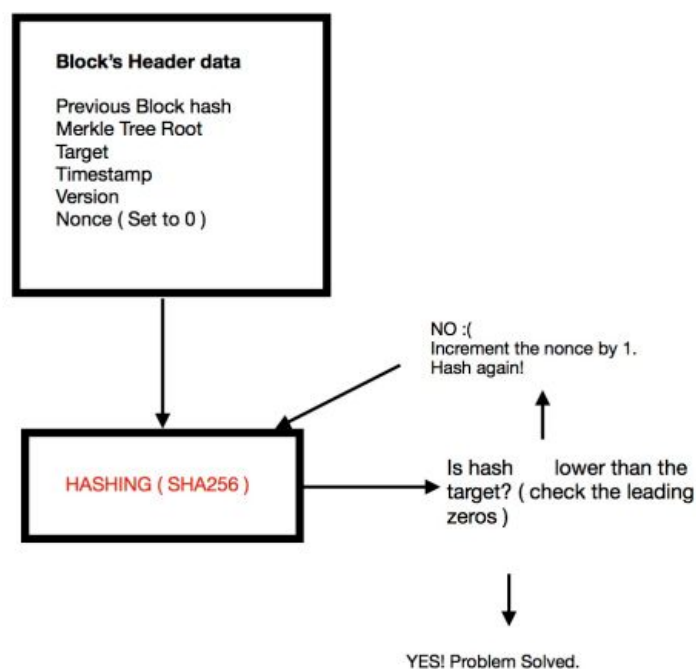
There are 18 leading zeros. This is our difficulty. Every hash with less than 18 leading zeros is invalid (every hash with 17 leading zeros and less would be lower than the required target).

The last field is the nonce. It is initialized to zero.

Everything is now ready for Joe's node to mine the block. The goal is to find a value for the nonce that will result in hash lower than the target. So, the mining node will try billions or trillions of nonce values before it gets a valid hash.

## Mining

In a picture, mining is:



As you can see, mining is like a lottery. There is no way to predict which nonce will solve the problem.

In a [previous article](#), I implemented a simple blockchain that demonstrates the concept of a nonce and how it changes the hash created.

In the case of Bitcoin, the hash function used is called SHA256. A hash algorithm always produces the same arbitrary length data given the same inputs. It is impossible to compute the same hash with two different inputs (collision). It is also impossible to predict the output of any given data in advance.

SHA256 always produces an output 256 bits long. Mining is finding the *nonce*, the only input that changes every time we run the hash function. It is very easy to prove that the nonce found indeed produces a valid hash. All the information is available, everyone can run the hash function and confirm if the hash is valid or not. Because it is also impossible to predict what the nonce will be, it also acts as a proof that the miner worked to get a valid hash (Hence => *Proof-of-Work*).

### Adjusting the target

In Bitcoin, a block is mined every 10 minutes or so. The difficulty is calculated so that it never deviates too much from this limit. If the difficulty stays the same in the long term, while computer power increases, it will take less and less time to mine a block. To make sure this doesn't happen, the Proof of Work target is a dynamic parameter. In the Bitcoin world, the target is adjusted every 2016 blocks. Then, we check the amount of time it took to mine those 2016 blocks. It should have taken 20160 minutes ( $2016 * 10 \text{ minutes}$ ). The difficulty increases or decreases depending on the time it took to mine those blocks.

Success!!

Joe's node is getting to work and starts hashing the block's header. After approximately 10 minutes, it discovers a valid hash. We can see the nonce used was 2469953656 (*Nonce in the left column*).

Joe's node immediately transmits the block to all its peers. They need to validate the new block before propagating it to its peers. This is the part where a dishonest miner can be found out. If the data is invalid, the miner would have wasted his time and computing power. Valid data includes:

- Block header hash is less than the target
- Block size is within acceptable limits
- Block timestamp is less than two hours in the future.
- The first transaction is a coinbase transaction (and only the first)
- The coinbase transaction has a valid reward.
- All transactions within the blocks are valid (also have a checklist on their own)

Every node validates independently new blocks following the exact same rules. This assures that miners cannot cheat. This is a key component of the decentralized consensus.

If the block is valid, the other miners will update their own copy of the blockchain with the new block 502246. Joe's block hash is now used by all the miners to mine the block 502247.

### 3.Types of blockchain: Public and private blockchains

There are basically two types of blockchain network related to access control.

1. Public Blockchain
2. Private(and consortium) Blockchain

#### Public Blockchain

Public Blockchain is accessible to everyone – anyone who wants to read, write can join the blockchain and can perform respective operations. The information once validated in the network cannot be changed and no single entity can have control over the network. Bitcoin is one of the first public blockchain networks to prove the value can be moved anywhere around the world without banks or other third parties.

Take the example of [Bitcoins](#), which are traded on such peer-to-peer networks. This cryptocurrency can be quite practical when the idea is to dissimulate criminal activities, since it's virtual, meaning that it can travel across borders in a seemingly anonymous way. "Seemingly" because we cannot discard what the benefits of a Blockchain, public or not, are: Every single transaction is recorded

across all nodes of the network, which provides a complete transaction history accessible by anybody. Now, it's not a routing number and a bank account number that are recorded but rather an (unusable because just made-up) I'D like this one: 1BsghtSLRHtKNgkdBEengR76b53CHAtpyT. The trick is then to convert Bitcoins into fiat currencies since it must be done via a centralized exchange platform like Coinbase/Binance. And transactions from Coinbase to a bank account can be flagged.

### Private Blockchain

These blockchains work in the same way as public blockchain but with restricted access. The restriction is applied to the users who are authorized to join the network and operate. They may have one or more entities that control the network.

Private blockchains are amazing for using at a privately-held company or organization that wants to use it for internal use-cases. By doing so, you can use the blockchain effectively and allow only selected participants to access the blockchain network. The organization can also set different parameters to the network, including accessibility, authorization, and so on!

**Example: Hyperledger Fabric** — hosted under Linux Foundation — is a private, permissioned and open source blockchain solution. Private means that blockchain networks are not publicly accessible and only invited parties can join the network. Permissioned means each party is clearly identified and every transaction is authenticated, authorized, validated and tracked. They are crafted to take advantage of blockchains without sacrificing the authority aspect of a centralized system.

### Consortium Blockchain:

A consortium blockchain is a blockchain where the consensus process is controlled by a pre-selected set of nodes; for example, one might imagine a consortium of 15 financial institutions, each of which operates a node and of which 10 must sign every block in order for the block to be valid. The right to read the blockchain may be public, or restricted to the participants, and there are also hybrid routes such as the root hashes of the blocks being public together with an API that allows members of the public to make a limited number of queries and get back

cryptographic proofs of some parts of the blockchain state. These blockchains may be considered "[partially decentralized](#)"

A consortium blockchain (also known as Federated blockchains) is a creative approach to solving the needs of organizations where there is a need for both public and private blockchain features. In a consortium blockchain, some aspects of the organizations are made public, while others remain private.

The consensus procedures in a consortium blockchain are controlled by the preset nodes. More so, even though it's not open to mass people, it still holds decentralized nature. How? Well, a consortium blockchain is managed by more than one organization. So, there is no one single force of centralized outcome here.

To ensure proper functionality, the consortium has a validator node that can do two functions, validate transactions, and also initiate or receive transactions. In comparison, the member node can receive or initiate transactions.

In short, it offers all the features of a private blockchain, including transparency, privacy, and efficiency, without one party having a consolidating power.

**Example:** Let's take an Oil & Gas consortium made of twenty companies for example. Each company operates one node on the network. In a consortium Blockchain, a percentage of the consortium's participants must sign every block in the chain before the block can be validated. So, it could be ten or fifteen companies required to validate a transaction, before the same transaction is validated throughout the network for all participants. A federated Blockchain is less decentralized than a public Blockchain and less centralized than a private Blockchain in regard to write permissions.--Harshil

Public Blockchain	Private Blockchain	Consortium or Federated Blockchain
Anyone can run BTC/LTC full node	Anyone can't run a full node	Selected members of the consortium can run a full node
Anyone can make transactions	Anyone can't run a full node	Selected members of the consortium can make transactions
Anyone can review/audit the blockchain	Anyone can't review/audit the blockchain	Selected members of the consortium can review/audit the blockchain



# Day 3

## 4.State of blockchain and industry wise implementations

### 1. Payment processing and money transfers

In April 2018, Banco Santander launched the world's first blockchain-based money transfer service known as "Santander One Pay FX," the service uses Ripple's xCurrent to enable customers to make same-day or next-day international money transfers. By automating the entire process on the blockchain, Santander has reduced the number of intermediaries typically required in these transactions, making the process more efficient. Blockchain technology can be used to decrease the cost of these transfers by reducing the need for banks to manually settle transactions.

### 2. Monitor supply chains

Shipping giant DHL is at the forefront of blockchain-backed logistics, using it to keep a digital ledger of shipments and maintain integrity of transactions. DHL has a major presence in the US and is one of the largest shipping companies to embrace blockchain.

### 3. Retail loyalty rewards programs

### 4. Digital IDs

Illinois is at the forefront of experimental blockchain in government with the Illinois Blockchain Initiative. The state-funded initiative has already put in place measures to use a distributed blockchain ledger to enhance the security of birth certificates, death certificates, voter registration cards, social security numbers and much more.

### 5. Data sharing

### 6. Copyright and royalty protection

Mediachain uses smart contracts to get musicians the money they deserve. By entering into a decentralized, transparent contract, artists can agree to higher

royalties and actually get paid in full and on time. Streaming giant Spotify acquired Mediachain in April 2017.

## **7. Digital voting**

Voatz is a mobile voting platform that runs on blockchain. The encrypted biometric security system makes it secure to vote on a mobile device from anywhere in the world without fear of hacking or data corruption. West Virginia is one of the first states to use the company's platform to collect votes from eligible service people and travelers abroad during elections

## **8. Real estate, land, and auto title transfers**

Propy is a global real estate marketplace with a decentralized title registry system. The online marketplace uses blockchain to make title issuance instantaneous and even offers properties that can be purchased using cryptocurrency

## **9. Food safety**

From a food manufacturer perspective, Singapore food firm SunMoon Food sees blockchain technology as vital for food safety via its traceability through the processing chain. "Blockchain addresses consumer concern over food safety as it enables real-time tracking of a fruit's origin from farm to fork.

## **10. Immutable data backup**

Filecoin aims to become a decentralized storage network that allows users to buy and sell unused storage on an open market. The filecoin protocol is an incentive layer on top of peer-to-peer file system IPFS, built with a native token and distributed ledger.

## **11. Tax regulation and compliance**

## **12. Workers' rights**

The U.S. State Department announced last month that they are working together with Coca-Cola and two other companies to launch a project that will use blockchain to create a secure registry for workers to battle against forced labor worldwide.

### **13. Medical recordkeeping**

BurstIQ's big data blockchain contracts help patients and doctors securely transfer sensitive medical information. The smart contracts establish the parameters of what data can be shared and even displays details of personalized health plans for each patient.

### **14. Weapons tracking**

### **15. Wills or inheritances**

### **16. Equity trading**

### **17. Managing Internet of Things networks**

Filament creates software and microchip hardware that lets connected devices operate on blockchain. The Reno-based company's products encrypt ledger data, distribute real-time information to other blockchain-connected machines and allow for the monetization of those machines based on timestamps.

HYPR thwarts cybersecurity risks in IoT devices with its decentralized credential solutions. By taking passwords off a centralized server, while using biometric and password-free solutions, the company makes IoT devices virtually unhackable.

### **18. Expediting energy futures trading and compliance**

### **19. Securing access to belongings**

### **20. Tracking prescription drugs**

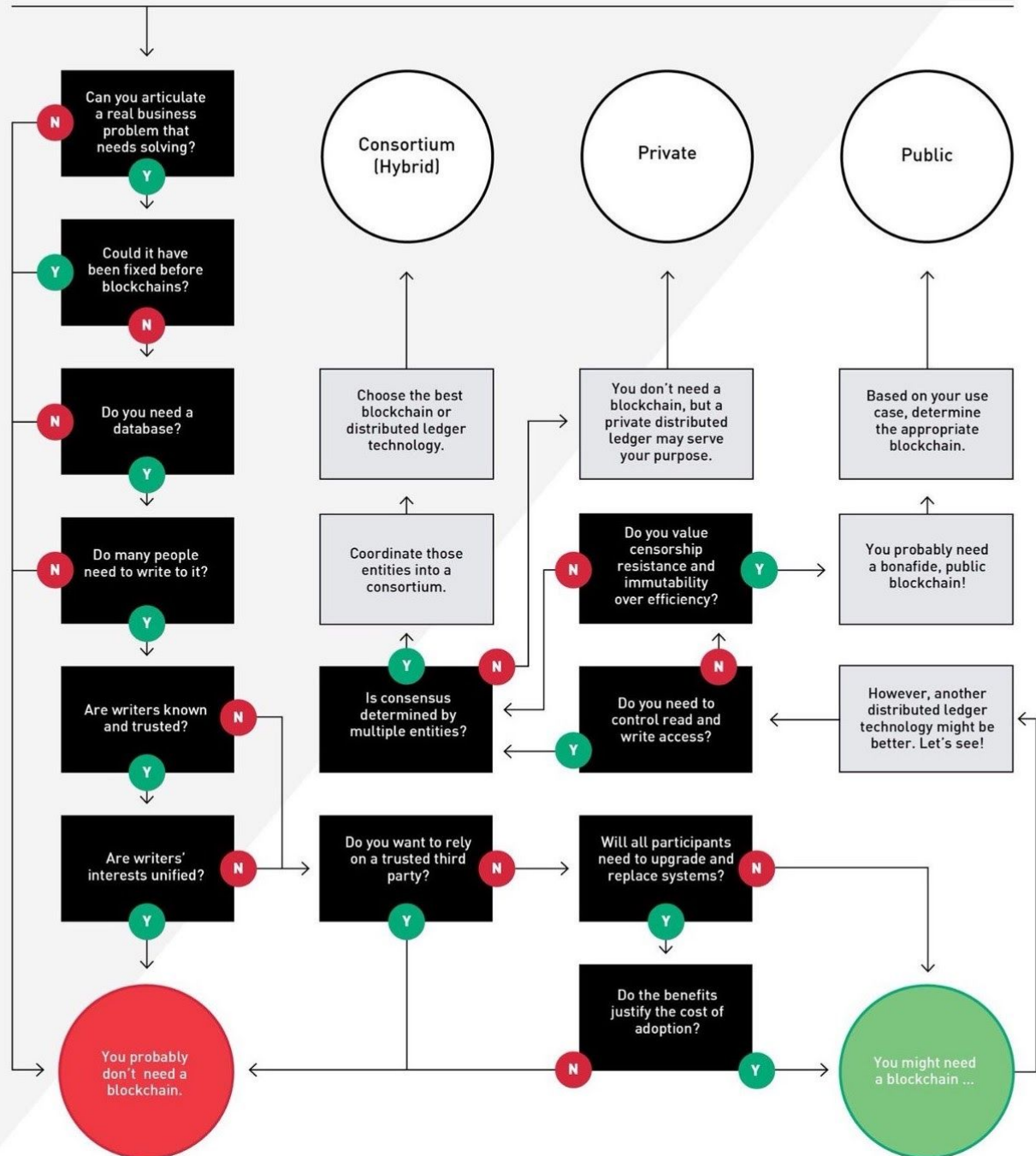
Medicea Technology Solutions is solving the problem of real-time information flow between multiple stakeholders in the pharmaceutical supply chain by using state-of-the-art mobile and cloud computing technology. By integrating the end consumers, retailers, stockists, and pharmaceutical companies, iMed India promises to cut the cost of medicine distribution and eliminate waste by 40% to 60%.

## Day 4

### Do you actually need a blockchain?

<https://medium.com/swlh/hyperledger-chapter-3-when-to-use-the-blockchain-technology-a5c414221bdf>

## DO YOU REALLY NEED A BLOCKCHAIN?



## Day 5

# Module 2: Deep dive into the Blockchain Environment

## 1.Transaction validation and Consensus mechanisms

In the bitcoin network, a transaction is performed between two accounts and the transaction is flooded across the entire network. This Flooding of transaction in the network is known as Gossip Protocol. All the nodes in the network validate the transaction and add it to their transaction pool in case the transaction is valid otherwise reject it. Each node has its own transaction pool that it maintains. All the transactions that are valid are placed inside the transaction pool.

In blockchain at a given point of time, it is not necessary for all the nodes to have the same transactions in their transaction pool. There could be various reasons for this to happen such as network latency, broken connection, less computational power etc. Bitcoin blockchain also defines the frequency at which blocks should be created inside the network which is set to 10 minutes in the current Bitcoin system.

The miner node takes the unconfirmed transactions in the mining pool and creates a block. Once a block is created, it gets flooded across the network using the gossip protocol. All the nodes receive the block and first validate the block if the block passes the validation they add it to their respective blockchain otherwise not.

Before going through the validation process we will have a look at the Block Anatomy.

A Block is divided into two a Header and Body.

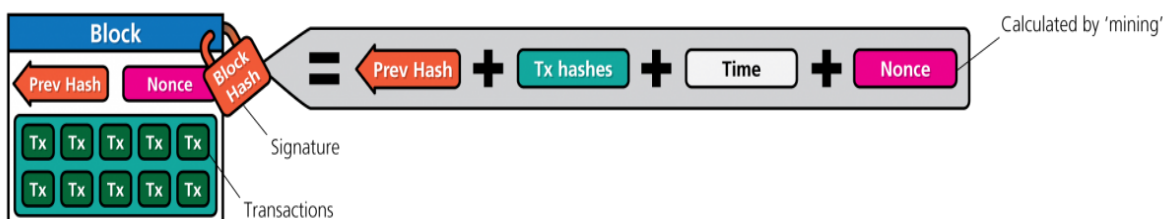
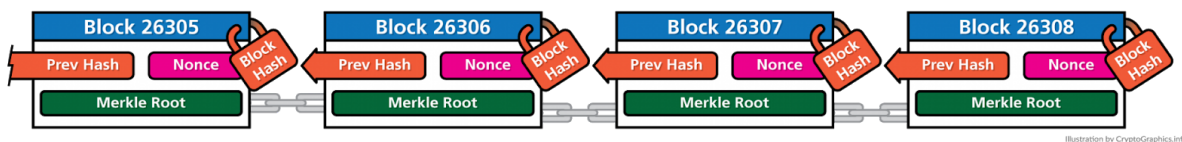


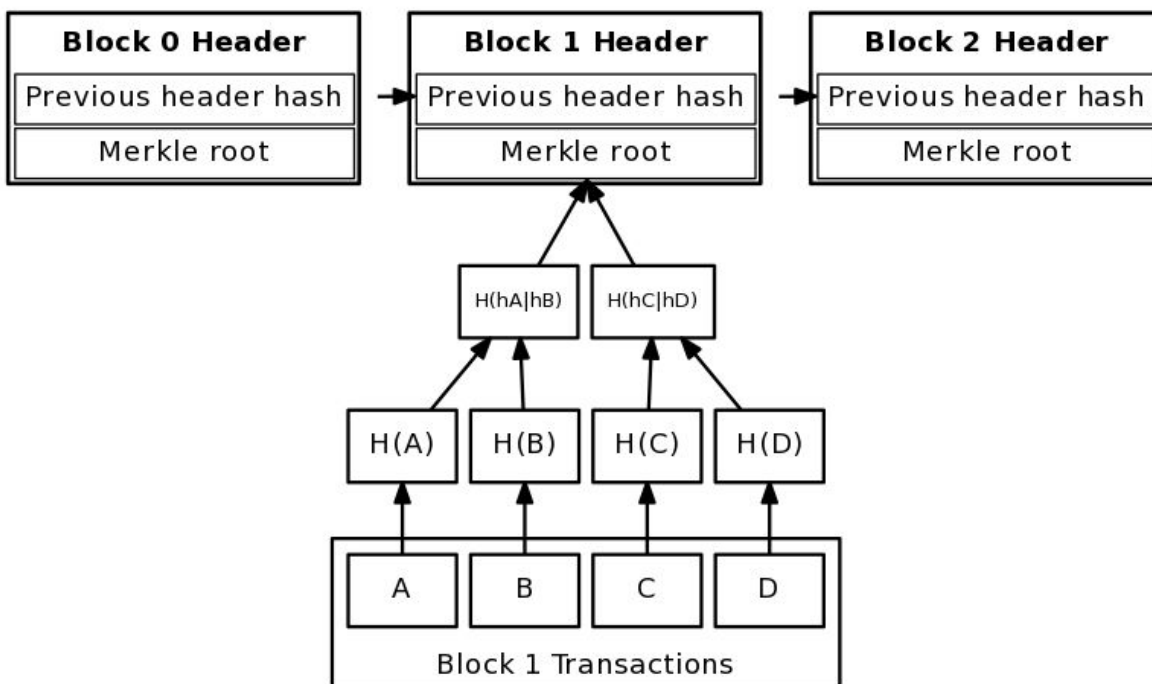
Illustration by CryptoGraphics.info

## A blocks header is comprised of the following components:

- i) **Merkle root** - aggregation of all the hash values of the transactions into a single hash value.
- ii) **Timestamp** - Timestamp of the block creation time.
- iii) **Nonce** - Random Value that is altered/updated to try different permutations to achieve the required difficulty level. You will learn more about this in the upcoming section.
- iv) **Transaction counter** - Count of the number of transactions in a block.



Now you must be thinking what the block body contains, So it contains the **Merkle Tree**.



Merkle tree connecting block transactions to block header merkle root

A Merkle tree stores all the transactions in a block by producing a digital fingerprint of the entire set of transactions. It allows the user to verify whether a transaction can be included in a block or not.

Merkle trees are created by repeatedly calculating hashing pairs of nodes until there is only one hash left. This hash is called the Merkle Root, or the Root Hash. The Merkle Trees are constructed in a bottom-up approach.

Every leaf node is a hash of transactional data, and the non-leaf node is a hash of its previous hashes. Merkle trees are in a binary tree, so it requires an even number of leaf nodes. If there is an odd number of transactions, the last hash will be duplicated once to create an even number of leaf nodes.

The above example is the most common and simple form of a Merkle tree, i.e., Binary Merkle Tree. There are four transactions in a block: TX1, TX2, TX3, and TX4. Here you can see, there is a top hash which is the hash of the entire tree, known as the Root Hash, or the Merkle Root. Each of these is repeatedly hashed, and stored in each leaf node, resulting in Hash 0, 1, 2, and 3. Consecutive pairs of leaf nodes are then summarized in a parent node by hashing Hash0 and Hash1, resulting in Hash01, and separately hashing Hash2 and Hash3, resulting in Hash23. The two hashes (Hash01 and Hash23) are then hashed again to produce the Root Hash or the Merkle Root.

## Day 6

### Mining Process

Before we move to the mining process we must learn something called **Hashcash**.

#### Video for more detail

<https://youtu.be/YUTwqG6e8LY>

Hashcash refers to a proof-of-work system that was created by Adam Back in 1997. His creation was initially meant to limit email spamming and DDoS attacks.

In more recent times, however, the system became popular for its involvement with Bitcoin and other cryptocurrencies, as an essential piece of the mining algorithm

Hashcash was originally meant to be a mechanism through which it would be possible to control and reduce the systematic abuse of shared, and available to all, internet resources such as email as well as anonymous remailers.

Hashcash used a cost function as an anti-DOS mechanism as it required malicious parties to use the processing power of their devices as a proof-of-work

If legitimate users want to send emails then they are required to compute a hash as proof that they have spent a reasonable amount of computing resources before sending the e-mail. Generating hashcash is a compute intensive process but does not inhibit a legitimate user from sending the e-mail because the usual number of emails required to be sent by a legitimate user is presumably quite low. On the other hand, if a spammer wants to send e-mails, usually thousands in number, then it becomes infeasible to compute hashcash for all e-mails, thus making the spamming effort expensive; as a result this mechanism can be used to thwart e-mail spamming. Hashcash takes a considerable amount of computing resources to compute but is easy and quick to verify. Verification is performed by the user who receives the email.

The process of mining means that a new block is created and added to the blockchain. Every miner creates a block from the transaction existing in its transaction pool.



- For a block to be valid and added to the network, the miner needs to solve a puzzle which is defined as below in the case of bitcoin blockchain.
- Hash of the block header < Value

Hash of the header is the hash of all the data present in the header appended together.

- In case the above condition is not met, the miners need to recompute the hash.
- To recompute the hash nonce is used whose value is changed in every iteration to arrive at a new hash. This is because all the other data in the block such as timestamp, Merkel root, number of transactions etc. are all fixed and can not be changed.
- To change the hash the only thing that can be changed is the nonce value. The difficulty to attain the acceptable hash in the network is termed as the difficulty of the network. Once the condition is satisfied, the block becomes a valid block.
- The miner who calculated a valid hash first is allowed to claim the creation of the block

A miner competes with other miners in the network to form a block. The miner which is able to first generate a block in the network is given a block reward. The block reward started from 50 BTC and currently, the miner earns a block reward of 6.25 BTC for every block.

Once a block gets created it is propagated to the entire network and all the peers don't add the block as such to their respective blockchains. Each peer node performs some validation on the incoming block.

The block header also contains a previous hash pointer of the block which precedes this block. Remember that blockchain is a chain of blocks that are connected by this previous hash pointer also known as the previous hash. The previous hash acts as a link between the blocks. Each node in the network checks whether the last block in its blockchain has the same hash as the previous hash pointer of the incoming block, it adds the block otherwise not

The blockchain is a trustless environment where nobody has to trust each other, and there is no central authority to ensure trust between the transacting parties. So how does blockchain ensure that valid transactions are performed in the network? Blockchain employs a set of validation checks known as consensus mechanisms for the validity of transactions.

In the blockchain, mathematical algorithms are used to verify transactions and ensure trust between transacting parties. Transacting parties have to trust the output achieved using these mathematical algorithms. These algorithms together are known as the consensus mechanism in the blockchain. The consensus can be for a transaction or for an entire block.

*The consensus for the transaction verification of transaction includes the checks for the following:*

**i) Sender Balance**

**ii) Valid Authority**

**iii) Valid signatures**

The consensus for a block is for the following condition:

$$\text{Hash(HDR)} < \text{Difficulty Level}$$

The consensus mechanism is difficult to achieve and easy to verify. The above-mentioned consensus mechanism is known as proof of work. In the proof of work consensus, miner nodes have to solve a computationally difficult problem to compute the hash for the current block. The network sets a difficulty target for the miners. This difficulty level plays a vital role in the consensus as well as the mining process.

*The difficulty value, also known as the target value, sets the difficulty level of the network. The difficulty level is used to regulate the mining of blocks in the network. The bitcoin network has a block creation time set to 10 mins and this value has to be maintained. The difficulty value adjusts with every block creation such that the value of block creation remains constant.*

The formula for calculation of difficulty level in bitcoin blockchain network is:

$$\text{Difficulty Level} = (\text{Previous Difficulty Level} * 20160) / (\text{Time taken to mine last 2016 block})$$

In case the time taken for the last block is greater than 10, the new value of difficulty level is lower than the previous one such that the blocks are created faster. In case the time taken for the last block is less than 10, the new value of difficulty level is greater than the previous one such that the blocks are created slower.

In proof of work mechanism, the consensus is achieved using an algorithm. Let us suppose miner node have computed the hash of the current block and have published it to the entire network. All other nodes in the network follow a set of rules to validate a block. Once all of the nodes have checked the validity of the block and found it to be valid, they give their consensus and the block is added to the blockchain. The functions performed while validating a node are given below:

- i) Check if a block with the same hash has already been received.
- ii) Check if there is a block in the already existing blockchain whose hash is previous.hash (current block).
- iii) Check if the hash of the current block satisfies the difficulty level.
- iv) Check if the timestamp of the current block is greater than the timestamp of the previous block.
- v) Update the current state of the blockchain new block number = parent block number + 1.
- vi) Broadcast updated blockchain to the network.

*Now there is a possibility that 2 miners mine simultaneously*

*It is possible that 2 miners find the valid nonce at the same time. This is because there can be more than one valid nonce which leads to a hash less than the target difficulty value. Also both of them can end up finding the same nonce at the same time. The new block is broadcasted to the network from both the miners. Few of them hear miner 1 as the winner while others may hear miner 2 as the winner. Those who hear miner 1 as the winner will add the block mined by miner 1 and those who hear miner 2 as the winner will add the block mined by miner 2. This leads to 2 different chains in the network. Now the next time miners may choose to work on either of the chains. Few will add blocks to chain 1, few will add blocks to chain 2. Eventually, the longest chain is deemed as a valid one and the transactions inside the blocks of invalid(shorter) chain are returned back to the transaction pool. These transactions are not cancelled. Just that they are returned back to the transaction pool to be picked up by the new blocks as their version of blockchain was deemed as invalid*

### **Proof of work faces three major challenges:**

**a. It is energy inefficient:** In proof of work millions of miners try to mine one block, and only one miner is successful. Rest of the energy spent by the miners goes to waste.

**b. It is computationally heavy:** Proof of work requires a lot of computation by the miners to mine one block successfully.

**c. 51% attack:** In case more than 51% of the nodes in the network are malicious the network could become unstable.

*Now the question arises is how 51% attack can cause the blockchain network not to behave properly. There is one more problem similar to 51% attack known as Byzantine Generals problem*

*The Byzantine Generals problem occurs when a malicious node propagates wrong messages or tampered transactions in the network that could compromise the security of data in the blockchain network. The blockchain network needs to be tolerant to such activities and needs to ensure that all the transaction data in the network is tamper free. Such networks are called as Byzantine Fault Tolerant Networks and they use the Byzantine fault tolerant consensus. For a network to be byzantine fault tolerant the number of malicious nodes in the network should be less than 1/3rd of the total nodes in the network. Whenever a node receives two conflicting messages, it goes for a majority vote and accepts the message which comes from the majority number of nodes. To ensure that the correct message is accepted by the nodes in the network the total number of malicious nodes needs to be less than 33.33% or 1/3rd of the network. Proof-of-work is a BFT consensus mechanism provided the 51% attack and pool mining does not happen.*

# Day 7

List of more consensus mechanism and how they work

## 1. Proof of Stake (PoS):

This is the most common alternative to PoW. Ethereum is shifting from PoW to PoS consensus. In this type of consensus algorithm, instead of investing in expensive hardware to solve a complex puzzle, validators invest in the coins of the system by locking up some of their coins as stake. After that, all the validators will start validating the blocks. Validators will validate blocks by placing a bet on it if they discover a block which they think can be added to the chain. Based on the actual blocks added in the Blockchain, all the validators get a reward proportionate to their bets and their stake increases accordingly.

In the end, a validator is chosen to generate a new block based on their economic stake in the network. Thus, PoS encourages validators through an incentive mechanism to reach an agreement.

## 2. Proof of Burn (PoB):

With PoB, instead of investing into expensive hardware equipment, validators 'burn' coins by sending them to an address from where they are irretrievable. By committing the coins to an unreachable address, validators earn a privilege to mine on the system based on a random selection process. Thus, burning coins here means that validators have a long-term commitment in exchange for their short-term loss.

Depending on how the PoB is implemented, miners may burn the native currency of the Blockchain application or the currency of an alternative chain, such as bitcoin. The more coins they burn, the better are their chances of being selected to mine the next block.

While PoB is an interesting alternative to PoW, the protocol still wastes resources needlessly. And it is also questioned that mining power simply goes to those who are willing to burn more money.

## 3. Proof of Capacity:

In the Proof of Capacity consensus, validators are supposed to invest their hard drive space instead of investing in expensive hardware or burning coins. The more hard drive space validators have, the better are their chances of getting selected for mining the next block and earning the block reward.

#### **4. Proof of Elapsed Time:**

PoET is one of the fairest consensus algorithms which chooses the next miner using fair means only. It is mainly used in a permission Blockchain network where permission is required for accessing the network.

In this, every individual on the network is supposed to wait for a random amount of time. The participant who has finished waiting for the given set of time will get a chance to be on the ledger to create a new block.

#### **5. Proof of Activity:**

There also exist other consensus algorithms like Proof of Activity, Proof of Weight, Proof of Importance, Leased Proof of Stake, etc. It is therefore important to wisely choose one as per the business network requirement because Blockchain networks cannot function properly without the consensus algorithms to verify each and every transaction that is being committed.

#### **List of consensus alogs:**

<https://medium.com/hackernoon/consensuspedia-an-encyclopedia-of-29-consensus-algorithms-e9c4b4b7d08f>

#### **Building your own blockchain in javascript**

<https://www.codementor.io/@savjee/implementing-proof-of-work-blockchain-in-javascript-part-2-k9ozymkqw>

# Day 8

## Blockchain Terms: Block, Gas, Gas Price, Merkle Patricia Tree

1. **Node** - user or computer within the blockchain architecture (each has an independent copy of the whole blockchain ledger)
2. **Transaction** - smallest building block of a blockchain system (records, information, etc.) that serves as the purpose of blockchain
3. **Block** - a data structure used for keeping a set of transactions which is distributed to all nodes in the network
4. **Chain** - a sequence of blocks in a specific order
5. **Miners** - specific nodes which perform the block verification process before adding anything to the blockchain structure
6. **Consensus (consensus protocol)** - a set of rules and arrangements to carry out blockchain operations
7. **Gas** - Gas is the unit of calculation that indicates the fee for a particular action or transaction. For example, if your transaction is to simply add two numbers, that is 3 units of work. If it is multiplication, that would be 5 units of work and so on. **This unit of work is called gas.**
8. **Gas Price** - The Gas Limit is the maximum amount of Gas that a user is willing to pay for performing this action or confirming a transaction. For example, you can say "I am willing to pay 3 Gwei per gas" for my transaction. If your transaction takes 100000 gas and you set the gas price to 3Gwei, you end up paying 300000 Gwei for your transaction.
9. **Gas Limit** - The price of Gas (Gas Price) is the amount that the user is willing to spend on each unit of Gas. As a developer, you don't want to blindly execute a transaction and realize your transaction took hundreds of dollars worth of Ether. To avoid this situation, you can specify a **gas limit** which indicates the maximum amount of gas you are willing to buy to execute your transaction.

Don't confuse this with the Block gas limit. Block gas limit is the maximum cap applied to each block in Ethereum. Currently, a block can only include transactions whose total sum of gas is less than 8 million. The reason you have a limit for each block is to prevent someone from just writing an infinite loop in which case the transaction wouldn't complete its execution resulting in the block not getting mined.

## Gas Explained:

To better understand how gas works in Ethereum, let's use an analogy. Suppose you are going on a road trip. Before you do so you go through these steps:

- You go to the gas station and specify how much gas you want to fill up in your car.
- You get that gas filled up in your car.
- You pay the gas station the amount of money you owe them for the gas.

Now, let's draw parallels with Ethereum.

Driving the car is the operation that you want to execute, like executing a function of a smart contract.

- The gas is well....gas.
- The gas station is your miner.
- The money that you paid them is the miner fees.

All the operations that users want to execute in ethereum must provide gas for the following:

- To cover its data aka intrinsic gas.
- To cover its entire computation.

**10. Nonce** - Nonce is the central part of this Proof of Work. The Nonce is a random whole number, which is a 32-bit (4 byte) field, which is adjusted by the miners, so that it becomes a valid number to be used for hashing the value of block. **Nonce is the number which can be used only once.** Once the perfect Nonce is found, it is added to the hashed block. Along with this number, the hash value of that block will get rehashed and creates a difficult algorithm.



## Day 9

# Cryptographic Primitives: Symmetric Cryptography, Asymmetric Cryptography, Hashing Function

### Cryptography

Asymmetric cryptography, also known as public-key cryptography, is one of the key components of blockchain technology. This form of cryptography allows everyone to verify the integrity of transactions, protect funds from hackers and much more. But how does it work?

To understand asymmetric cryptography it is important to first understand the meaning of cryptography.

Cryptography is a method of using advanced mathematical principles in storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. Encryption is a key concept in cryptography—It is a process whereby a message is encoded in a format that cannot be read or understood by an eavesdropper. The technique is old and was first used by Caesar to encrypt his messages using Caesar cipher. A plain text from a user can be encrypted to a cipher text, then sent through a communication channel and no eavesdropper can interfere with the plain text. When it reaches the receiver end, the cipher text is decrypted to the original plain text. — [SSL2BUY](#)

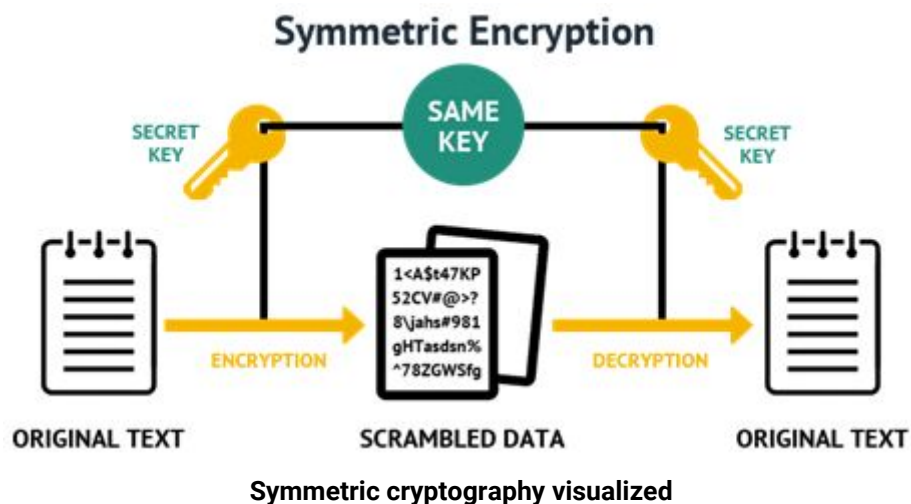
Asymmetric cryptography is one of these methods and is a more advanced version of symmetric cryptography, which we will explain first.

### Symmetric cryptography

Symmetric cryptography is a 'simple' form of cryptography which uses a single key to encrypt and decrypt data. This key can be almost anything, ranging from a number to a word to a random string of characters. This key is then used to encrypt the data after which the data can get sent across a network safely. To decrypt the data the receiver needs the key (the same one that the sender used to encrypt the data).

To get a better idea of how this works, we have created a visualization of the process below. First of all the sender encrypts a message with the shared key, the sender then sends the message without having to worry that anybody without the

shared key is able to read the message. The receiver then receives the encrypted message and decrypts it (with the same shared key).



The main downside of this cryptography method is that the key needs to be shared with everyone who needs to access the data, which can often be quite hard.

## Asymmetric Cryptography

Now you are familiar with symmetric cryptography, let's dive into asymmetric cryptography. Asymmetric cryptography is similar to symmetric cryptography, but is a bit more complex and also has a solution to the main downside of symmetric cryptography.

The main distinction from symmetric cryptography is the usage of keypairs. Asymmetric cryptography uses keypairs, instead of a shared key, in order to encrypt and decrypt data.

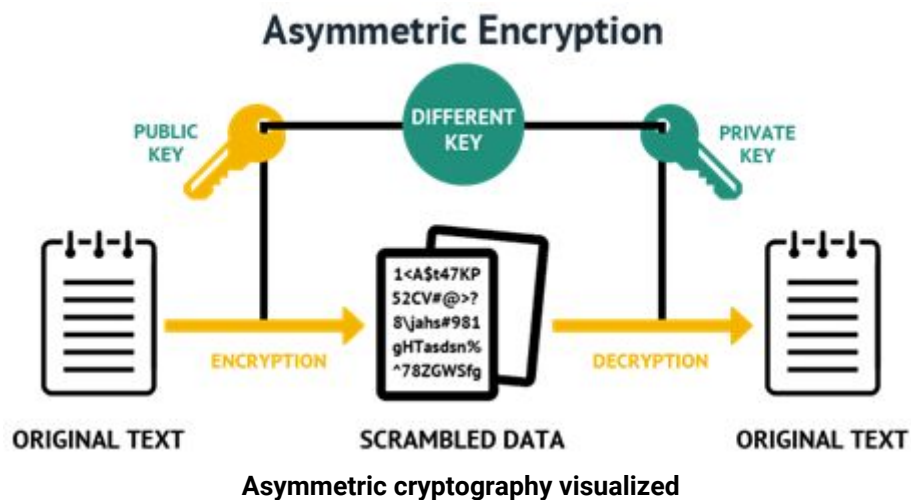
Keypairs consist of 2 parts, a public key and a private key.

A public key can be seen as a username, it is available to everyone, can be shared with everyone, and everyone can view the history of the account with that username. The username is tied to a password (private key), but there is absolutely no way to derive the password (private key) from a username. It is also not possible to authorize any action on the account with just the username.

A private key can be seen as a password to an account with a certain username. It is not publicly available and should not be shared with anyone. The private key is used

to authorize actions on the accounts. Unlike with 'normal accounts', to access the account, or to authorize any action on the account, only the private key is needed.

In the graphic below you can see how these keys work in practice, when sending a message to somebody securely. First of all the sender encrypts the message with the public key of the receiver, the sender can then send the (encrypted) message safely, as the only way to view the message is to decrypt it with the corresponding private key which only the receiver has. The receiver then receives the message and is able to decrypt it using the private key.

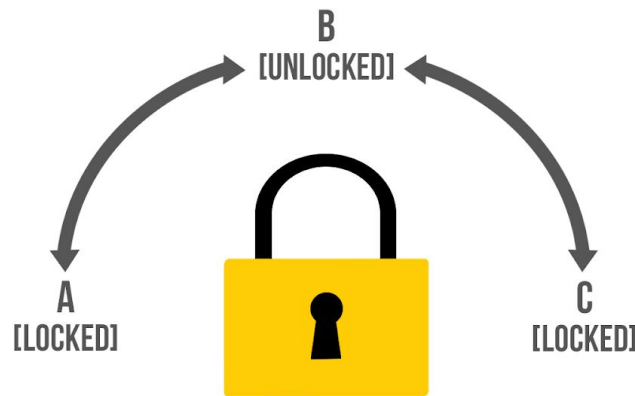


Due to the usage of keypairs asymmetric cryptography is a (much) safer way to encrypt data and make sure only those who are supposed to receive it are able to receive it. These keypairs also allow themselves to be used for authentication purposes, which we will talk more about below.

## Digital Signatures

Digital signatures are essentially signatures that provide integrity using asymmetric cryptography. They are widely used in many protocols for authentication purposes and have already proven to be both very useful and secure.

Digital signatures are incorruptible and easily verifiable thanks to their usage of asymmetric cryptography. Since they use asymmetric cryptography (and a private key is only linked to a single person) digital signatures also have the quality of non-repudiation, meaning they can be as legally binding as a normal signature. But how can you verify the integrity of certain messages, or how could you ensure others are able to verify the integrity of your message?



**Public and private key functionality visualized**

The answer to these questions is rather simple and can be explained with the image above. The lock in the image above has 2 keys, the public key, which can only turn counterclockwise, and a private key, which can only turn clockwise. We assume the public key of each lock is widely available.

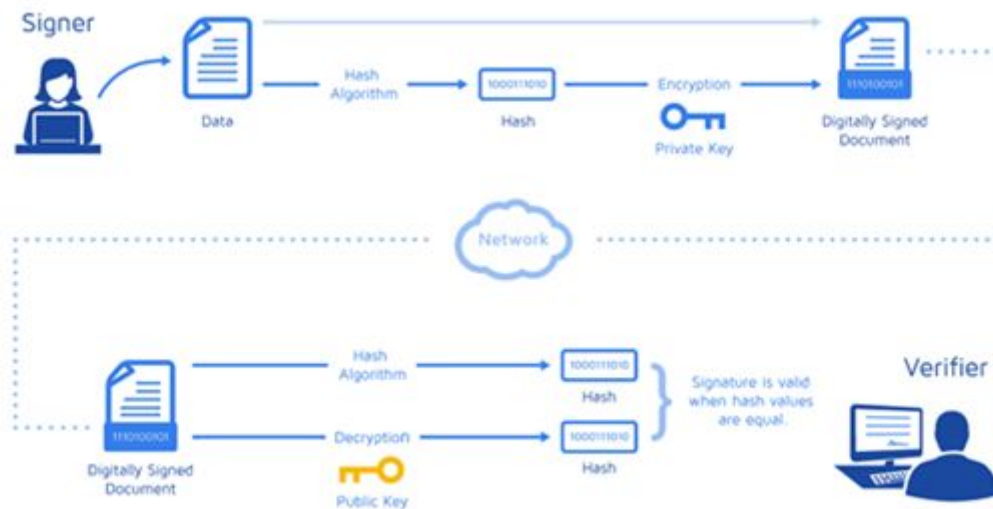
Let's say Alice wants to send an unencrypted message to Bob, but Bob wants to ensure Alice has sent the message. Alice puts the message in a box and then puts the lock on the box. She then locks the lock using her own private key (from B to C). After locking the box she sends it to Bob. Bob then receives the box and can verify the integrity (and sender) of the message using Alice's public key. If Alice's public key does not open the box the integrity of the message and box should be considered violated.

If you encrypt ("lock") something with your private key, anyone can decrypt it ("unlock"), but this serves as a proof you encrypted it: it's "digitally signed" by you. —Panayotis Vryonis

If Alice wants to send an encrypted message to Bob, and only Bob should be able to both decrypt the message and verify the identity of the sender, she would have to take some extra steps. First of all Alice puts the message in a box and puts a lock on the box, she then locks the box using her own private key (from B to C). She then puts the (locked) box in a new box and puts a new lock on the box. She locks the new lock using Bob's public key (from B to A). After Alice is done she sends the box to Bob. Once Bob receives the box he can open the outer box with his private key, he can then open the inner box with Alice's public key. If the inner box can not be opened with Alice's public key the integrity of the message and box should, yet again, be considered violated.

These 'locks' are called digital signatures.

Note that in reality both a signature generation algorithm, which takes a key and data, and a signature verification algorithm, which takes a message and signature, are needed to do the things explained above.



## The usage in blockchains

While both asymmetric cryptography and digital signatures have been popular ever since they were invented, they are probably most known for their implementations in blockchains. Since blockchains are essentially widely distributed ledgers, it is of the essence that the cryptography used is both reliable and functional.

Most cryptocurrencies use keypairs (and thus asymmetric cryptography) to manage 'addresses' on the blockchain. The public key is the address, which 'holds' the tokens and it can be viewed by anyone. The private key is used to access the address and authorize actions for the 'address'. Below is an example of an arbitrary address on the Ethereum blockchain.

Etherscan - Sponsored slots available. [Book your slot here!](#)

## Overview

Balance: 9.564060650332366351 Ether

Ether Value: \$1,026.22 (@ \$107.30/ETH)

Transactions: 2693 txns

## Misc:

Address Watch: [Add To Watch List](#)

Token Balance: View (\$101.07)

14

## Transactions Internal Txns Erc20 Token Txns Comments

Latest 25 transactions from a total of 2693 transactions

TxHash	Block	Age	From		To	Value	[TxFee]
<a href="#">0xb6f303574e50353...</a>	<a href="#">7158419</a>	58 secs ago	<a href="#">0x2b83315e60f04a0...</a>	IN	<a href="#">0xdfc38911f6e0bfdd...</a>	0.009839748 Ether	0.000095152
<a href="#">0xc213d6db8e7997...</a>	<a href="#">7158255</a>	48 mins ago	<a href="#">0xdfc38911f6e0bfdd...</a>	OUT	<a href="#">0x75cc366804b984...</a>	0.0003 Ether	0.000095152
<a href="#">0x51527d8563d2b4...</a>	<a href="#">7158249</a>	50 mins ago	<a href="#">0xdfc38911f6e0bfdd...</a>	OUT	<a href="#">0xf1d35ebfcd8744...</a>	0.0003 Ether	0.000095152
<a href="#">0x0b1f079031b4db5...</a>	<a href="#">7158180</a>	1 hr 14 mins ago	<a href="#">0xdfc38911f6e0bfdd...</a>	OUT	<a href="#">0x433be9fad1995d9...</a>	0.0003 Ether	0.0000785004
<a href="#">0xd8bb98ae0a6d82...</a>	<a href="#">7158090</a>	1 hr 42 mins ago	<a href="#">0xdfc38911f6e0bfdd...</a>	OUT	<a href="#">0x3d89ff3e4fb996df...</a>	0.0003 Ether	0.000095152
<a href="#">0xfecac61281e1df7...</a>	<a href="#">7158090</a>	1 hr 42 mins ago	<a href="#">0xdfc38911f6e0bfdd...</a>	OUT	<a href="#">0xdac508443d8244...</a>	0.0003 Ether	0.000095152
<a href="#">0xd37f3bb01d07971...</a>	<a href="#">7157997</a>	2 hrs 7 mins ago	<a href="#">0xdfc38911f6e0bfdd...</a>	OUT	<a href="#">0x183d2fd28719006...</a>	0.0003 Ether	0.00010538084

## Arbitrary Ethereum address

Digital signatures are also widely used in cryptocurrencies. They can be used to sign transactions more safely (offline) and are also used in multisignature contracts and wallets. These multisignature contracts and wallets require digital signatures from multiple (different) private keys before any action can be executed.

## Day 10

### SHA-256

Presently SHA-256 is the most secure hashing function. This function expresses the possible combinations or values that results from the given input data. SHA stands for Secure Hashing Function, and 256 expresses the numerical quantity of the fixed bit length. This means that the target is correct 256 bit, and as mentioned, Bitcoin uses a 65-hexadecimal hash value.

Using the SHA-256 function makes it (nearly) impossible to duplicate a hash. That's because there are just too many combinations to try and process. Therefore, a significant amount of computational work is required. So much so that Bitcoins are no longer mined with personal computers. Presently, to mine, you need to rely on Application-Specific Integrated Circuits or ASICs. Achieving this target has a probability of  $2^{256}$ . If you remember your exponents, you will deduce this is an incredibly difficult variable to hit.

Furthermore, using this hash function means that such a hash is intentionally computationally impractical to reverse. The intentional result is that it requires a random or brute-force method to solve for the input.

#### A Case in Point

Consider the following, if I have 1 six-sided dice, I have a 1 in 6 chance of rolling a 6. However, the more sides my dice has (say 256 sides), the more my chances of rolling a 6 decrease. That's 1 in 256, which is still better than your odds of using brute-force on an extent hash.

A hash rate is then the speed at which hashing operations take place during the mining process. If the hash rate gets too high and miners solve the target too quickly, this increases the potential for a collision. When that happens, the difficulty of the hash needs adjusting accordingly. For example, at present, a Bitcoin is mined/hashed about every 10 minutes.

#### Collision Resistance

Due to the complexity and sensitivity of SHA-256, reversing the hash sequence in an effort to find the original input data is basically impossible. The difficulty of meeting SHA-256 means that this hash is extremely secure because it is "collision resistant."

Collision resistance expresses the likelihood of two different networks solving the same hash at the same chance is minuscule.

Therefore, given the possible permutations of SHA-256, the probability of a collision is negligible. Below is a comparison of two different hash outcomes. The first only uses the single hash function (SHA-1), while the second uses the double hash function (SHA-256). And as you can see, the double hash function produces a much more complicated hash and as a consequence is far more collision resistant.

www.thesslstore.com.cer	(application/x-x509-ca-cert) - 2042 bytes
SHA-1	2377b8642eac9f622b826f7a83500704f3cc3488
SHA-256	0feb43ff09dbea9487421fd3930527828703995e9f1d95d01656e3474cff2493

Here are a few examples of other cryptographic hash functions and when collision resistance broke, and it will become evident why SHA-256 is currently the favored hash:

- **MD 5:** It produces a 128-bit hash.
- Collision resistance was broken after  $\sim 2^{21}$  hashes.
- **SHA 1:** Produces a 160-bit hash.
- Collision resistance broke after  $\sim 2^{61}$  hashes.
- **SHA 256:** Produces a 256-bit hash.
- This is currently being used by Bitcoin.
- **Keccak-256:** Produces a 256-bit hash.
- Currently used by Ethereum.

If you want to more deep down into how SHA-256 works then you can explore the below link

<https://medium.com/bugbountywriteup/breaking-down-sha-256-algorithm-2ce61d86f7a3>



## ECDSA

Elliptic Curve Digital Signature Algorithm or ECDSA is a cryptographic algorithm used by Bitcoin to ensure that funds can only be spent by their rightful owners.

### A few concepts related to ECDSA:

**i) Private Key:** A secret number, known only to the person that generated it. A private key is essentially a randomly generated number. In Bitcoin, someone with the private key that corresponds to funds on the [block chain](#) can spend the funds. In Bitcoin, a private key is a single unsigned 256 bit integer (32 bytes).

**ii) Public Key:** A number that corresponds to a private key, but does not need to be kept secret. A public key can be calculated from a private key, but not vice versa. A public key can be used to determine if a signature is genuine (in other words, produced with the proper key) without requiring the private key to be divulged. In Bitcoin, public keys are either compressed or uncompressed. Compressed public keys are 33 bytes, consisting of a prefix either 0x02 or 0x03, and a 256-bit integer called  $x$ . The older uncompressed keys are 65 bytes, consisting of constant prefix (0x04), followed by two 256-bit integers called  $x$  and  $y$  ( $2 * 32$  bytes). The prefix of a compressed key allows for the  $y$  value to be derived from the  $x$  value.

**iii) Signature:** A number that proves that a signing operation took place. A signature is mathematically generated from a [hash](#) of something to be signed, plus a private key. The signature itself is two numbers known as  $r$  and  $s$ . With the public key, a mathematical algorithm can be used on the signature to determine that it was originally produced from the hash and the private key, without needing to know the private key. Resulting signatures are either 73, 72, or 71 bytes long (with approximate probabilities of 25%, 50%, and 25%, respectively—although sizes even smaller than that are possible with exponentially decreasing probability).

### What is double-spending?

As the term suggests, double-spending means spending the same money twice. With physical cash such as coins and notes, this simply isn't possible and therefore isn't an issue.

Let's look at this example: You go to Dmart and buy groceries worth Rs.100. You pay in cash and hand over a Rs.100 note. As soon as the cashier puts the cash in the register, you can't re-spend it unless you physically steal it.

## Why is double-spending an issue for digital money?

Digital money is different from cash. When you make a transaction with digital cash, you are broadcasting the transaction to all the 'nodes' in the network (nodes are computers that run the software on which the currency is supported). These nodes need to receive and confirm the transaction, which takes time. Hence the problem: what's to stop someone copying a transaction and rebroadcasting it before it's been confirmed on the network?

To prove that no attempts to double-spend have occurred, the blockchain provides a way for all nodes to be aware of every transaction. With bitcoin, all transactions are publicly announced to all nodes. They can then agree on a single history of the order in which they were received. Bitcoin's solution to double-spending is that if the majority of the nodes agree on which transaction was first to be received, later attempts to double-spend are irrelevant.

- Double-spending occurs when a blockchain network is disrupted and cryptocurrency is essentially stolen. The thief would send a copy of the currency transaction to make it look legitimate, or might erase the transaction altogether.
- Although it is not common, double-spending does occur. What is much more likely, however, is cryptocurrency being stolen from a wallet that wasn't properly secured.
- The most common method of double-spending is when a blockchain thief will send multiple packets to the network, reversing the transactions so that it looks like they never happened.

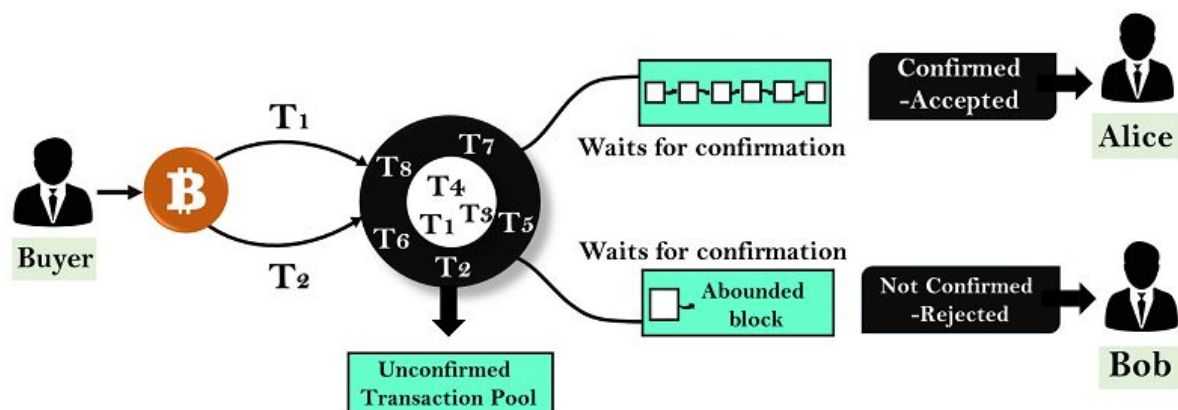
## Bitcoin's Timestamp Server Explained

Satoshi Nakamoto's white paper proposed the use of a timestamp server as a solution to the double-spending problem. This server takes a hash of a block of transactions and then broadcasts this hash to all the nodes in the bitcoin network. This timestamp proves that all the data in the hash couldn't have been created *after* the hash was published (obviously). As each timestamp includes the previous timestamp in its hash, this forms an immutable (unchangeable) record of the order in which transactions took place. Each timestamp reinforces the ones before it.

## There's a record of every bitcoin transaction ever made

### An example of the bitcoin network in action

Let us suppose you have 1 BTC and try to spend it twice. You made the 1 BTC transaction to Alice. Again, you sign and send the same 1 BTC transaction to Bob. Both transactions go into the pool of unconfirmed transactions where many unconfirmed transactions are stored already. The unconfirmed transactions are transactions which are not picked by anyone. Now, whichever transaction first got confirmations and was verified by miners, will be valid. Another transaction which could not get enough confirmations will be pulled out from the network. In this example, transaction T1 is valid, and Alice will receive the bitcoin.

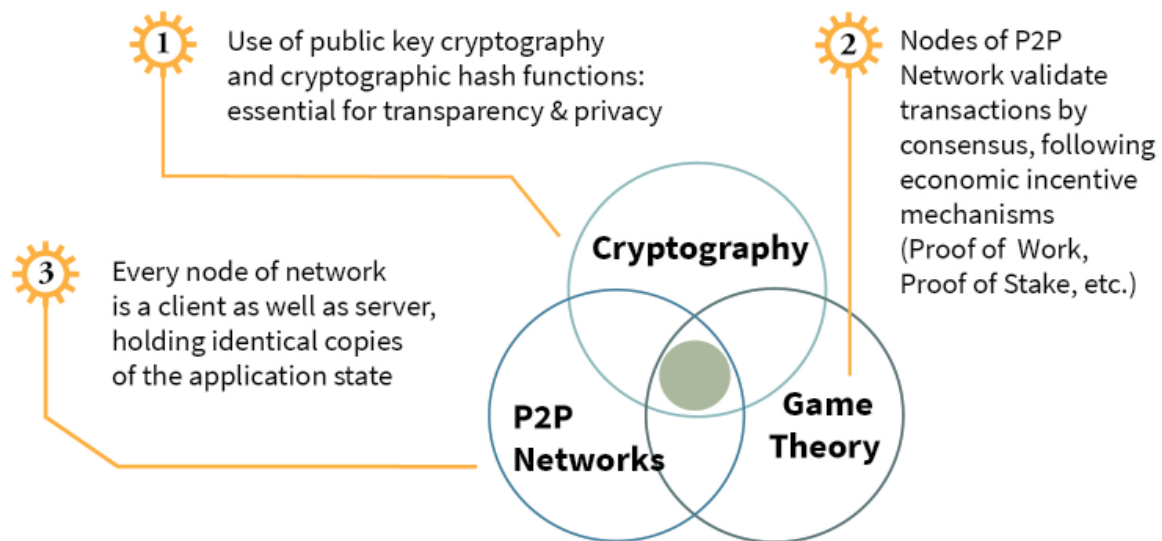


## Nodes and Blockchain Forks (soft and hard fork)

Firstly understand that blockchain is a protocol upon which a software will be built upon

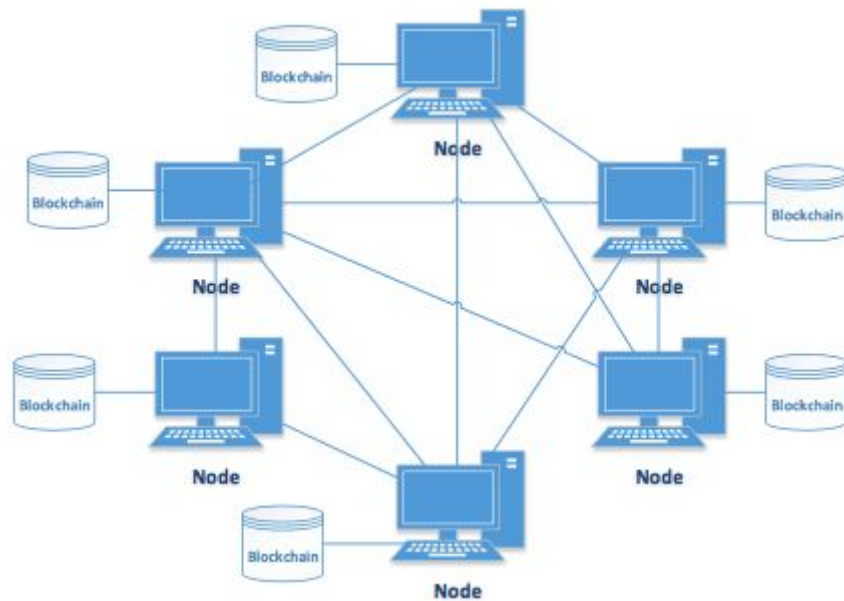
Till now we understood about game theory/mining and Cryptography let's go to the networks

## *Behind the Blockchain Protocol*



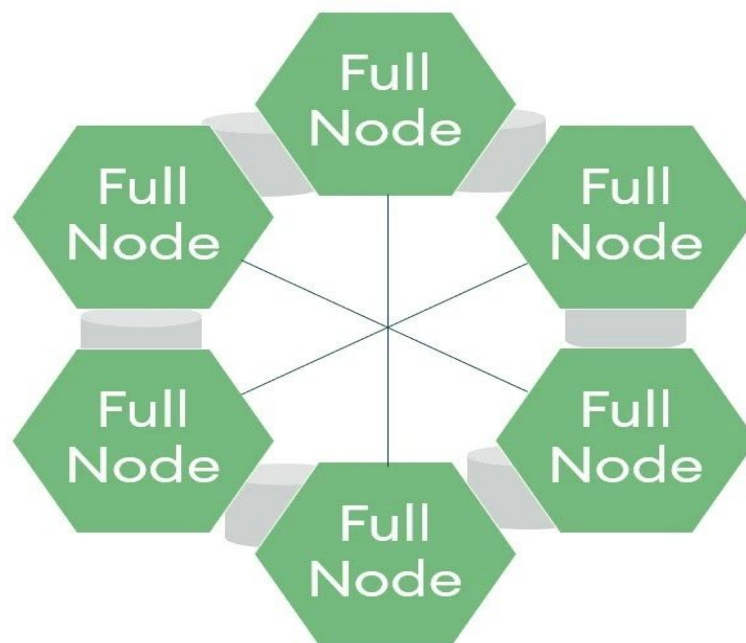
From the Book "**Token Economy**" by Shermin Voshmgir, 2019  
Excerpts available on <https://blockchainhub.net>

So that means the journey of understanding blockchain doesn't finish after knowing how the chain is made, But you need to explore the elements in the blockchain network which manages the transactions.



[Image Source](#)

Blockchain node can be **full-node** or **light-client**



[Image Source](#)

Majorly there are three different types of nodes in any blockchain network

- i) **Full Node**
- ii) **Light Nodes**
- iii) **Mining Nodes**

And how do they differ ?

	Can propose new blocks	Send new transactions	Holds wallet balance information	Holds the complete data history of the blockchain
Mining Nodes	Yes	No	No	No
Full Nodes	No	Yes	Yes	Yes
Light Nodes	No	Yes	Yes	No

[Image Source](#)

### There is no rule that a miner node cannot hold a full node and light nodes with him

For Mining to happen a miner always has a full node with him without which he/she could never win the consensus

### FullNodes:

A full node stores all information held on a blockchain and acts as a core server across decentralised blockchain networks. Each block in a blockchain is verified, authenticated and stored by all the full nodes in a network.

Each full node will store a copy of all blockchain transactions which means full nodes are data-heavy. Due to this, they require more advanced computing power and energy and thus, are expensive. It's estimated that the Bitcoin network has [over 10,000 operational full nodes](#).

Full nodes are essential to the overall security and validity of a blockchain network and have specific responsibilities which differentiate them from other types of nodes. Two key differentiating features include:

- **Validation of signatures** in each block transaction: once a new block is about to be added to a blockchain stored with them, a full node inspects each digital signature to authenticate the transaction. A digital signature is usually the private key the sender of a transaction uses to sign each transaction.
- **Validation of mined block** full nodes have the authority and decision making influence to reject new transactions or blocks. This includes if other nodes on the network have validated the incoming node. Reasons for rejecting newly formed transactions could include incorrectly formatted blocks or a duplication of a transaction (potentially fraudulent transactions).

A full node is a program that fully validates transactions and blocks. Almost all full nodes also help the network by accepting transactions and blocks from other full

nodes, validating those transactions and blocks, and then relaying them to further full nodes.

Most full nodes also serve lightweight clients by allowing them to transmit their transactions to the network and by notifying them when a transaction affects their wallet. If not enough nodes perform this function, clients won't be able to connect through the peer-to-peer network—they'll have to use centralized services instead.

Many people and organizations volunteer to run full nodes using spare computing and bandwidth resources—but more volunteers are needed to allow Bitcoin to continue to grow.

A full Bitcoin node can be established through [different software implementations](#), but the most used and popular one is the [Bitcoin Core](#). These are the minimum requirements to run a Bitcoin Core full node:

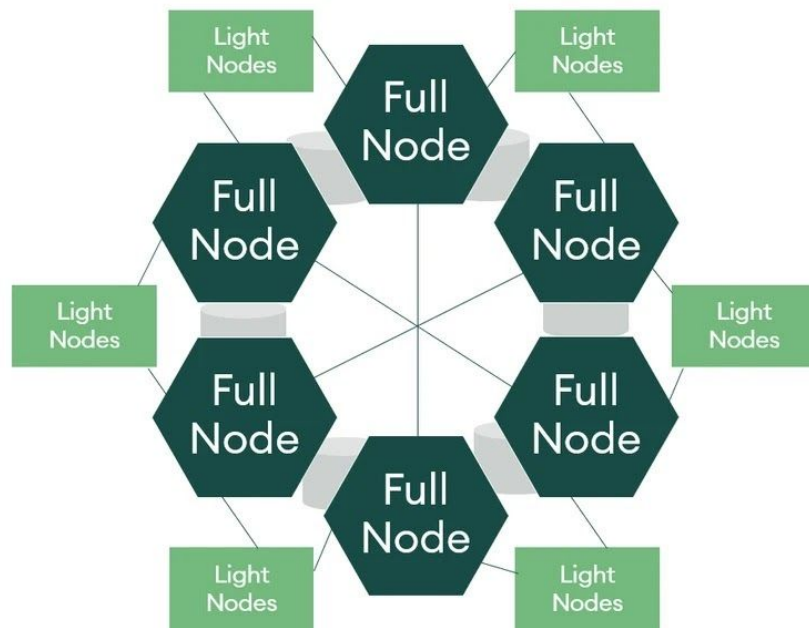
- Desktop or laptop with a recent version of Windows, Mac OS X, or Linux.
- 200GB of free disk space.
- 2GB of memory (RAM).
- High-speed internet connection with upload speeds of at least 50 kB/s.
- An unmetered connection or a connection with high upload limits. Online full nodes may reach or exceed an upload usage of 200 GB/month and a download usage of 20 GB/month. You will also need to download ~200GB when you first start your full node.
- Your full node should run at least 6 hours a day. Even better if you run it continuously (24/7).

Many volunteer organizations and users are running full Bitcoin nodes as a way to help the Bitcoin ecosystem.

So do you want to act as a node but can't buy hardware and bandwidth?

Introducing you to light nodes!

## Light Nodes:



[Image Source](#)

Light nodes have a similar purpose to full nodes however instead of holding a complete history of a blockchain, they typically hold a block header which seeks to support and query the validity of previous transactions. The block header is a detailed summary of a specific block and includes information relating to a particular previous block it is connected to. Information stored in the block header include: the timestamp of the block and a unique identifying number (also known as a nonce).

light nodes are connected to full nodes (commonly referred to as their parent nodes) and allow access for light nodes to verify transactions which have been included in a specific block. Unlike full nodes, light nodes do not store a copy of a blockchain fully history and rely totally on full nodes to provide them with validated data.

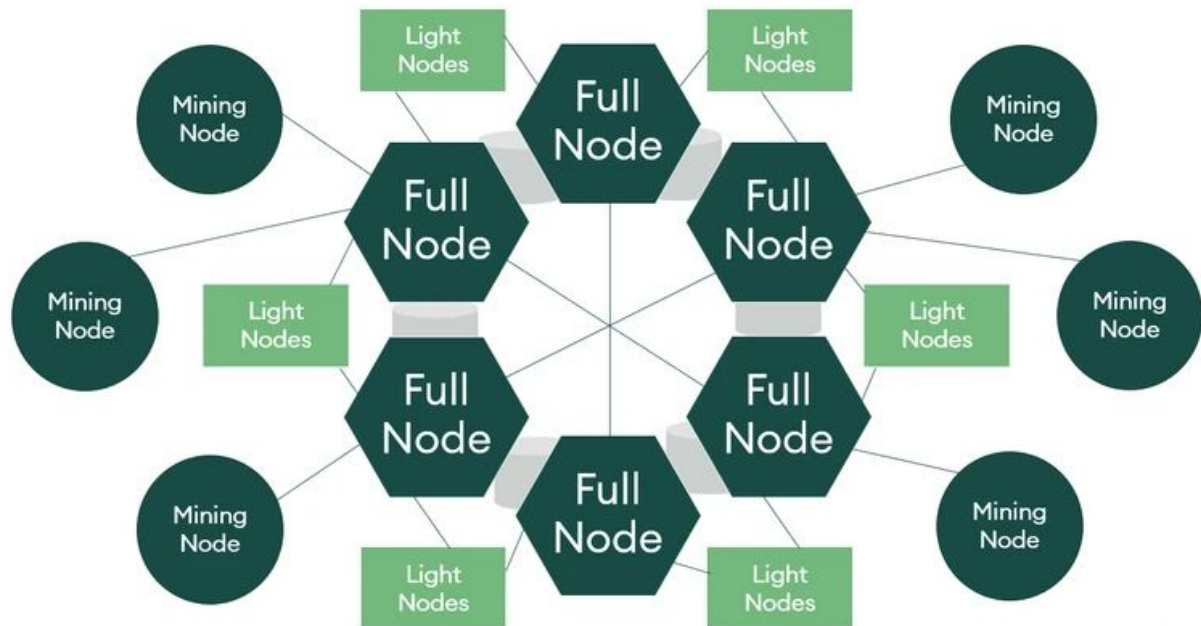
Using light nodes supports a blockchain to further decentralise and grow a network. As light nodes hold and process less data than full nodes, they require much fewer resources to maintain and run. This allows a blockchain network to grow more sustainably compared to full nodes. Examples of light nodes include desktop or online wallets.

In short, light nodes / SPV is the method through which a user can check whether some transactions were included or not in a block, without having to download the entire block data. Thus, SPV clients rely on the information provided by other full nodes (supernodes). The lightweight clients work as communication endpoints and are used by many cryptocurrency wallets.  
[\[https://academy.binance.com/en/articles/what-are-nodes\]](https://academy.binance.com/en/articles/what-are-nodes)

Now let me introduce you to the nodes which safeguard the blockchain the mining nodes!



## Mining Nodes:



[Image Source](#)

Mining nodes are nodes which produce blocks for the blockchain. You may be familiar with the term “bitcoin miners”; miners are classified as nodes. The role of these miners is to complete an action such as finding a nonce that satisfies for the current network difficulty (typically done through the brute force use of high-performance computer equipment). The first to broadcast their results and have attained validation from the full nodes is granted the right to add to the blockchain. Readers can revisit how mining works in our previous article [here](#).

In response to solving cryptographic problems, the miner is typically rewarded through the issuance of cryptocurrencies or tokens.

Mining nodes are only responsible for creating blocks to add to the blockchain, they are not responsible for the maintenance or validity of future blocks (unlike full nodes). Mining nodes offer users the opportunity to work with others and increase the rate of receiving rewards over a period of time.

It is worth mentioning that the process of mining consumes energy and miners typically have high start-up costs in purchasing the computer power required. This has led to the popularity of mining pools which exist to pool hashrate from multiple sources/users.[11]

While the solo miners’ full nodes make use of their own copy of the blockchain, pool miners work together, each one contributing to his own computational resources (hashpower). In a mining pool, only the administrator of the pool is required to run a full node - which can be referred to as a pool miner’s full node.

Before trying to mine a block, a miner needs to gather pending transactions that were previously accepted as valid by the full nodes. Next, the miner creates a candidate block (with a group of transactions) and tries to mine that block. If a miner manages to find a valid solution for their candidate block, they broadcast it to the network so that other full nodes can verify the validity of the block. Therefore, the consensus rules are determined and secured by the distributed network of validating nodes and not by the miners

Wait!

There are still more number of nodes which have similar duties/powers

<https://nodes.com/>

The above link explains it perfectly !

So how is this decentralized ? There are different types of nodes here ! Are they powerful?

Well , Miners have the power to validate the blocks i.e power to validate a transaction in a block and send it to the network for synchronizing and achieving consensus.

Full nodes have the ability to reject or add this verified block to the existing chain in their node.

On what basis do the full nodes reject or add the verified blocks?

Based on the protocol the full node follows!  
(Protocol implemented by a software)

## Day 11

### What is a fork?

A software fork occurs at a point where software is copied and modified. The original project lives on, but it's now separate from the new one, which takes a different direction. Suppose that the team of your favorite cryptocurrency content website had a major disagreement with how to proceed.

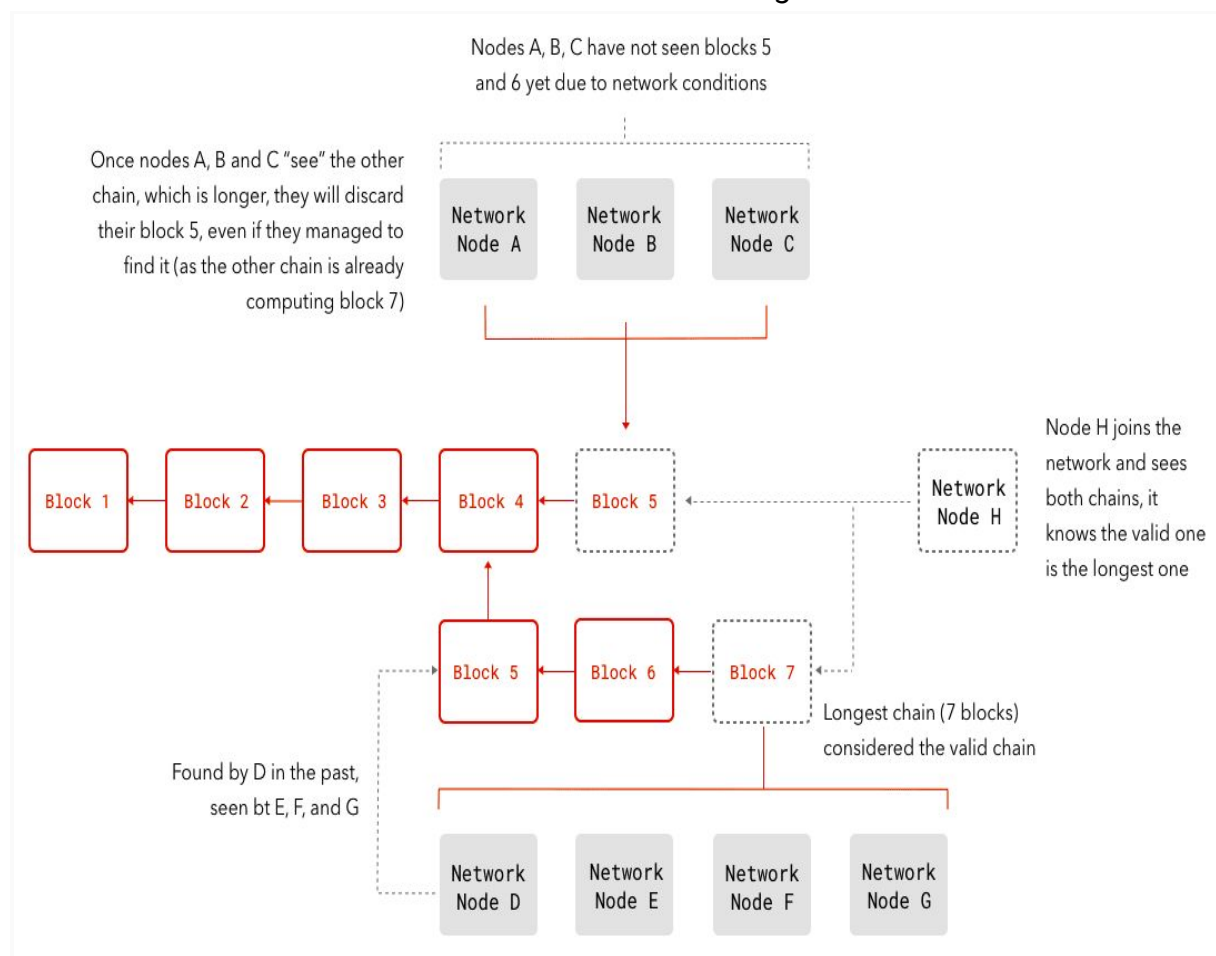
One part of the team might replicate the site on a different domain. But going forward, they would post different types of content than the original.

The projects build off a common ground and share a history. Just like a single road that later splits into two, there's now a permanent divergence in their paths.

Note that this kind of thing happens a lot in [open-source](#) projects, and has been happening for a long time before the appearance of [Bitcoin](#) or [Ethereum](#). However, the distinction between hard forks and soft forks is one almost exclusive to the blockchain space. Let's discuss those a bit more.

The below figure shows unplanned fork which is usually automatically handled by all the softwares that implements blockchain protocol

Forking in each and every Blockchain is different, based on the design architecture and use case of the chain. We shall look into a more generic scenario.



[Image Source](#)

The above scenario occurs whenever due to network latency or many reasons

1. Anytime two miners find a block at nearly the same time.
2. Developers seek to change the rules the software uses to decide whether a transaction is valid or not.

So what if some full nodes follow new protocols and the other full nodes following old protocols ?

Welcome to the world of **Forking!**

This is not a spoon ! But more than it ;)

Well it's an art of dividing!

**i) Soft fork**

**ii) Hard fork**

Whenever there is disagreement among miners about an existing protocol, they make way for a new protocol.

Adding to it if full nodes support the miners who want the new protocol to be implemented then it leads to a soft or hard fork.

Before going further we need to understand a concept that a fork happens until there is a division , If both the parties agree

### **Hard Fork:**

Hard forks are backward-incompatible software updates. Typically, these occur when nodes add new rules in a way that conflicts with the rules of old nodes. New nodes can only communicate with others that operate the new version. As a result, the blockchain splits, creating two separate networks: one with the old rules, and one with the new rules.

A hard fork is a permanent divergence from the previous version of the Blockchain, and nodes running previous versions will no longer be accepted by the newest version. A hard fork is a radical change to the protocol that makes previously valid blocks or transactions invalid. Any transaction on the forked (newer) chain will not be valid on the older chain. All nodes and miners will have to upgrade to the latest version of the protocol software if they wish to be on the new forked chain. This essentially creates a fork in the Blockchain, one path which follows the new, upgraded Blockchain, and one path which continues along the old path.

Hard Fork is usually done only when there is enough support from the mining community. Only when the majority of miners give positive signals towards the upgrade or fork, the developers of the chain starts work on the upgraded code. Typically, the support should come from 90 to 95 percent of the miners.

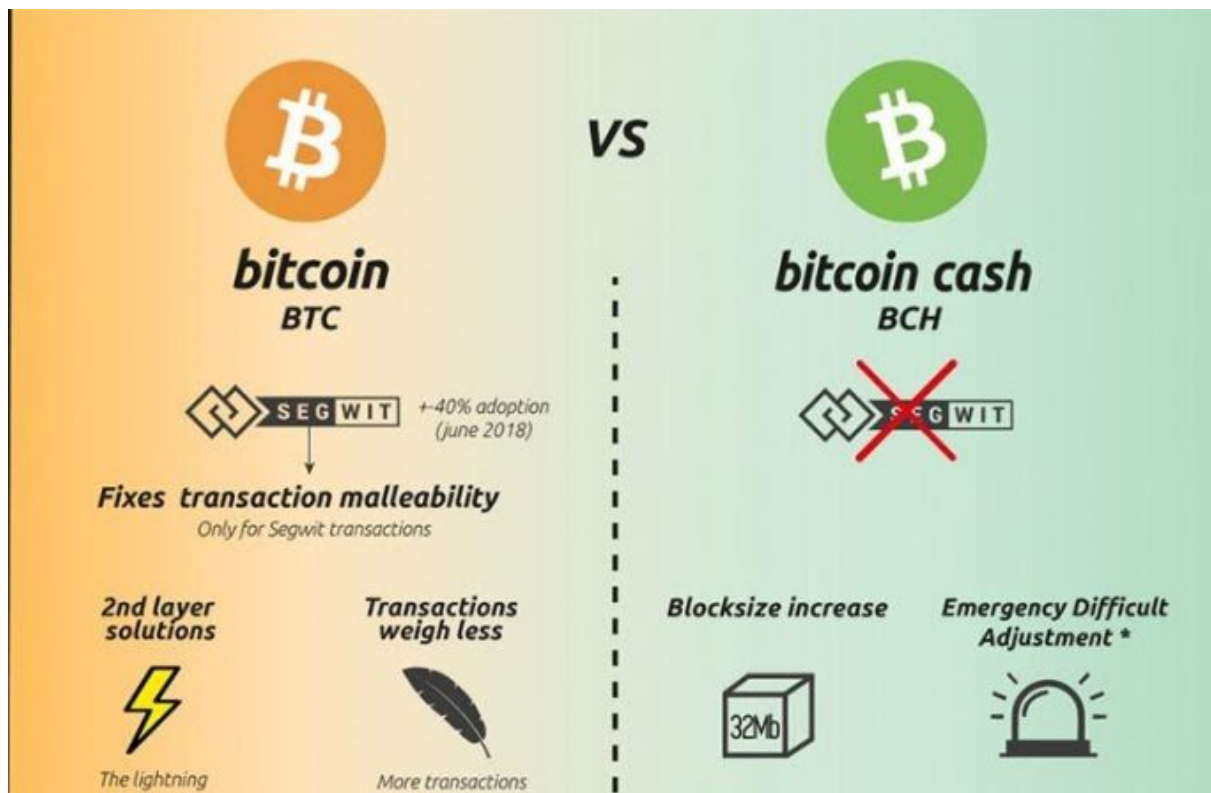
### **Bitcoin to Bitcoin Cash:**

A good example of a Hard Fork was when Bitcoin Cash came into existence. Previously Bitcoin was the dominant player in the cryptocurrency game. But as transaction times slowed and fees started to rise, the future of Bitcoin came into question.

Bitcoin was starting to become more of a store of value, and less and less an actual usable currency. This issue led to a political divide inside of the Bitcoin community. One half of the community wanted Bitcoin to remain unchanged. They felt that any changes to the Bitcoin platform would deviate from its original vision. While the other half of the community felt that Bitcoin needed to evolve if it was going to fulfill its dream of becoming the world's first decentralized currency.

So, fix the debate, the community agreed to have a Hard Fork or splitting Bitcoin into two separate asset classes. Bitcoin, and Bitcoin cash. Bitcoin would remain unchanged from its original vision and stay a store of value, while Bitcoin Cash would become much faster and cheaper to use currency. The result is there are now two completely separate types of Bitcoins you can own. Each with its own value and its own prices

Note that the nodes which are following the new protocol will be orphaned out.



[Image Source](#)

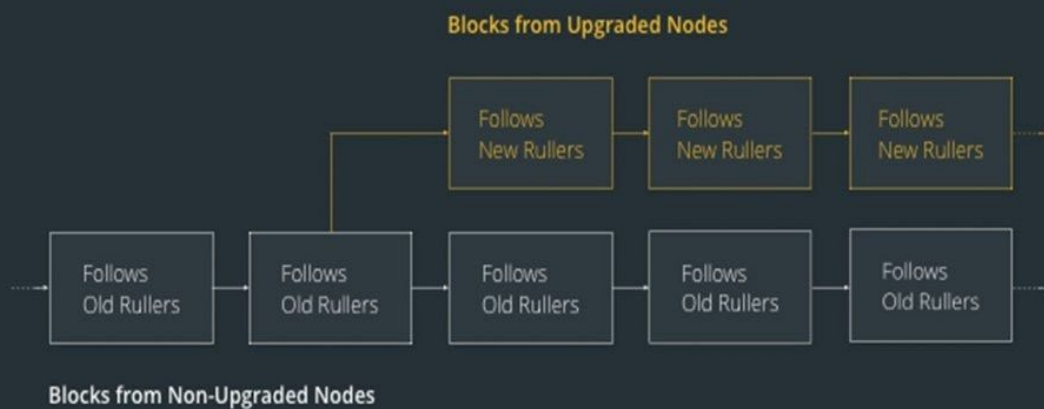
Another example of hard fork is ethereum hard fork from **ethereum** to **ethereum classic**

<https://medium.com/swlh/the-story-of-the-dao-its-history-and-consequences-71e6a8a551ee>

The above incident explains about the **hard fork** occurred due to **DAO**.

A latest hard fork Byzantium Hard Fork is about introducing more quality and security(In ethereum based\*)

## HARD FORK NODES



[Image Source](#)

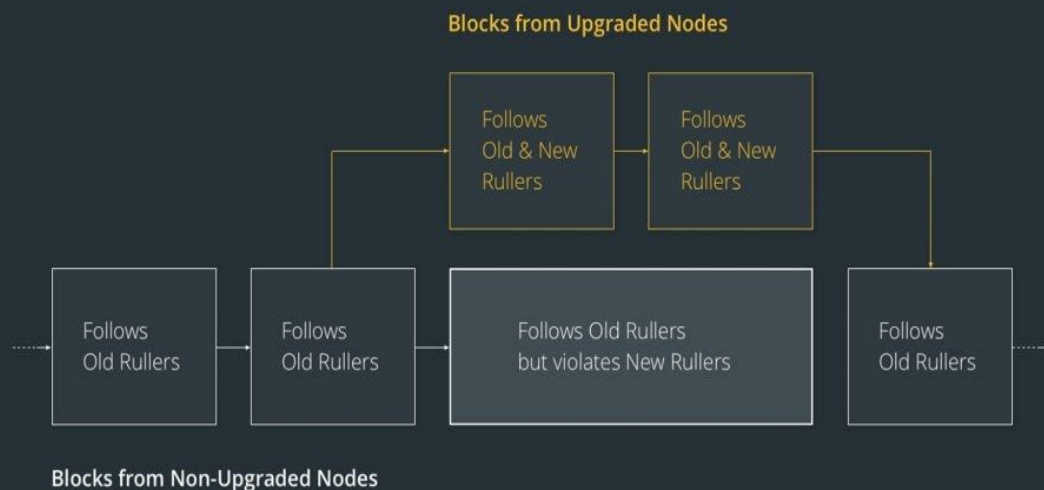
Please note that in the above image ,both the chains cannot communicate with each other i.e non - compatible separate chains with miners and full nodes following different protocols.

Just imagine if the both the chains and divided but still communicate with each other i.e act as a single chain?

Introducing you to the Soft fork ! yeah it's very soft ;)

**Soft fork:**

## SOFT FORK NODES



[Image Source](#)

A soft fork is a backward-compatible upgrade, meaning that the upgraded **nodes can still communicate with the non-upgraded ones**. What you typically see in a soft fork is the addition of new improvements that don't clash with the current protocol. Soft forks do not lead to network splits. All nodes still accept and communicate with other nodes that aren't implementing those rules, but may filter out some of the information they pass.

'A soft fork is said to happen when a change to the software protocol keeps it backward compatible' -

What the above statement means is that the new forked chain will follow the new rules and will also honor the old rules. The original chain will continue to follow the old rules. This kind of fork requires only a majority of the miners upgrading to enforce the new rules, as opposed to a hard fork which requires (almost) all nodes to upgrade and agree on the new version.

New transaction types can often be added as soft forks, requiring only that the participants for e.g. sender and receiver and miners understand the new transaction type. This is done by having the new transaction appear to older clients as a "pay-to-anybody" transaction (of a special form) and getting the miners to agree to reject blocks including this transaction unless the transaction validates under the new rules. A soft fork can also occur at times due to a temporary divergence in the Blockchain when miners using non-upgraded nodes violate a new consensus rule their nodes don't know about.

There are two kinds of soft forks, such as a miner-activated soft fork (MASF), which means that the new rules are accepted when the majority of miners' upgrade, and a user-activated soft fork (UASF), when the appliance of new rules happens without the miners' support.

<https://medium.com/swlh/hard-forks-and-soft-forks-what-are-they-and-whats-their-difference-91163ac77095>



## Soft Fork vs Hard Fork:

Difference	Description	Soft Fork	Hard Fork
Implementation Wise	What are the barriers to implementing a soft and hard fork?	(i) Works at the network level  (ii) Requires 51% of mining hash power (depends on the blockchain type)  (iii) Does not require nodes, exchanges or users to upgrade	(i) Works at the protocol level  (ii) Hashpower is irrelevant  (iii) Nodes, exchanges and users must all upgrade
Network Splits	It is where users cannot safely transact with each other because they are referencing different forks.	(i) Will not cause a network split. Users, nodes, and exchanges can be sure they are transacting safely  (ii) Miners may find themselves mining on the "non-forked chain" thus wasting hashpower	(i) Nodes, exchanges, users, and miners, who fail to fork will be split from the rest of the network

## Consequences of Forking:

Fundamentally, both of the above types of forks serve different purposes. Contentious hard forks can divide a community, but planned ones allow the freedom to modify the software with everybody in agreement.

Soft forks are a gentler option. Generally speaking, you're more limited in what you can do as your new changes can't conflict with the old rules. That said, if your update can be crafted in such a way that it remains compatible, you don't need to worry about fragmenting the network.