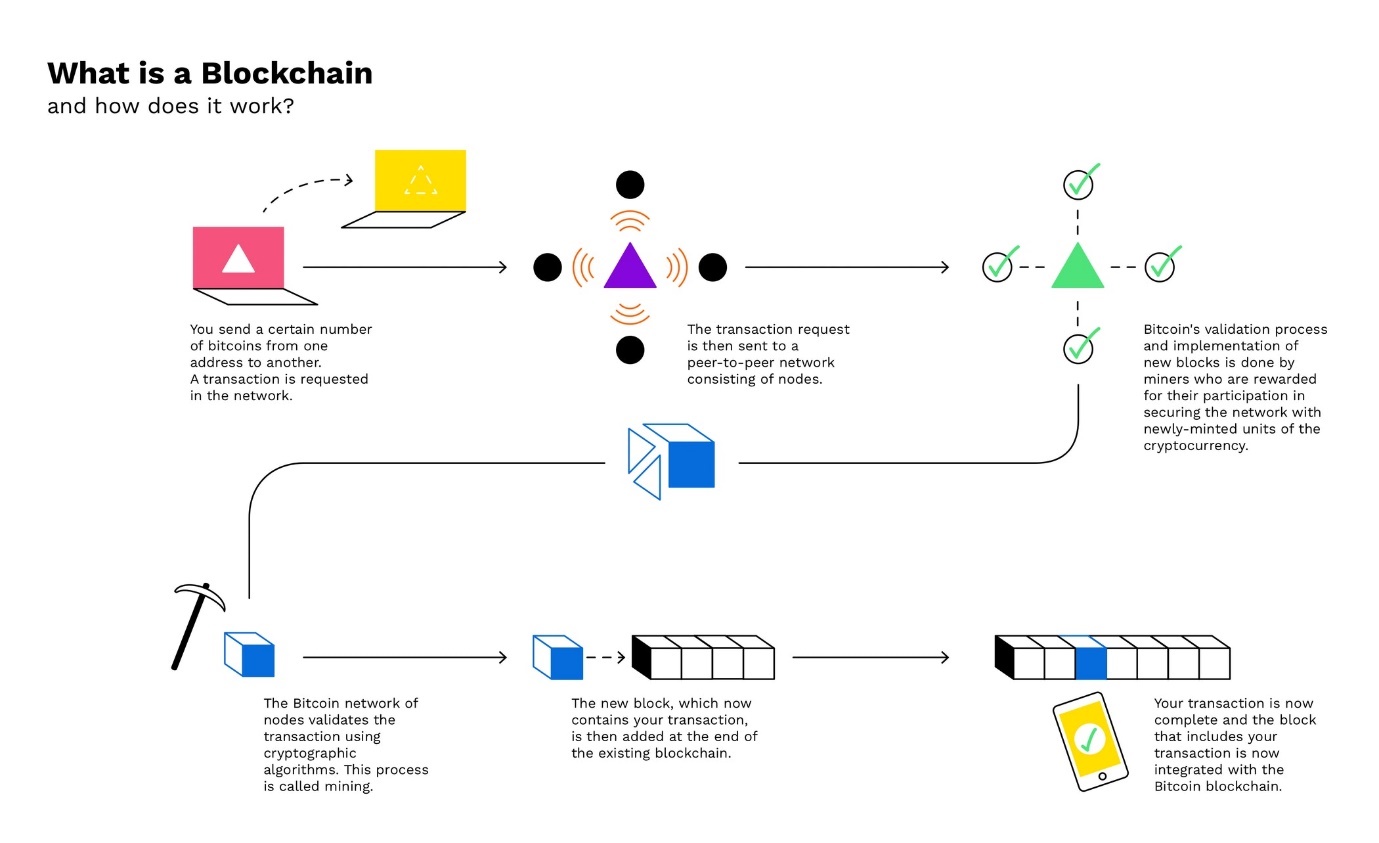
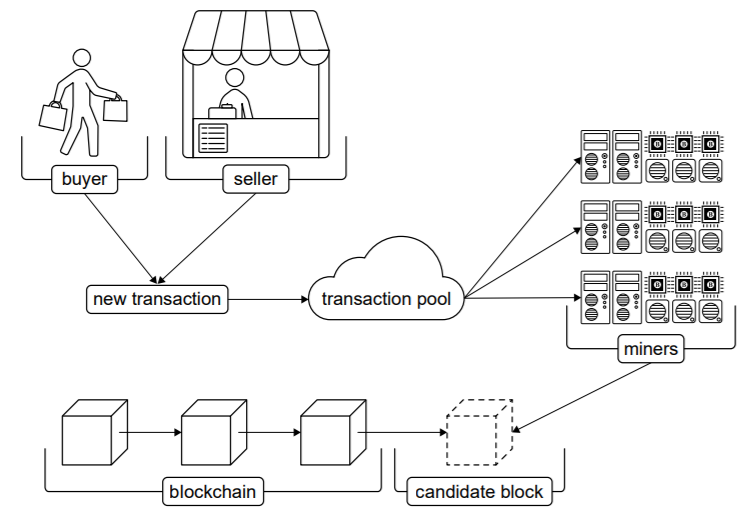
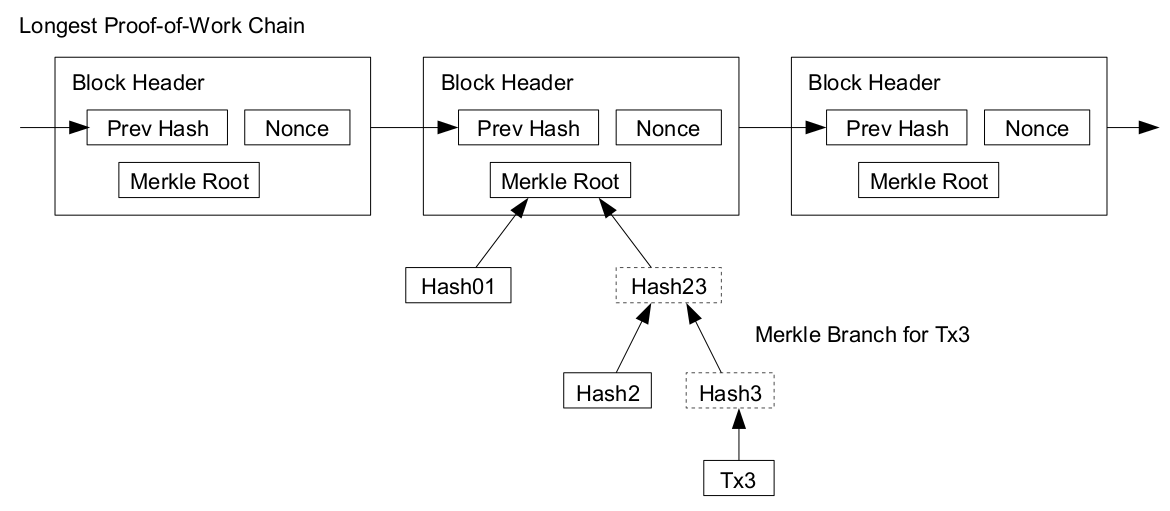
# Blockchain Process







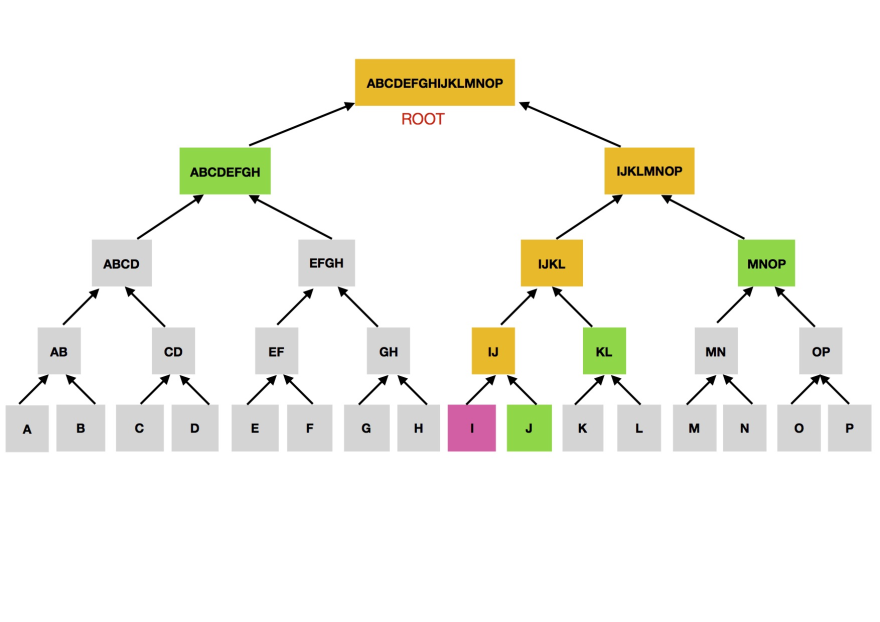
# Key Terms

Validation checks

* Digital signature to verify identity of sender
* Buyer has not already spent the currency used in the transaction (Double spent)

|  |  |
| --- | --- |
| Ledger | A financial record of transactions. A **blockchain** is a form of distributed ledger which is encrypted, decentralized and anonymous. |
| Cryptocurrency | Digital or virtual currency secured using cryptographic techniques such as **blockchain**. It is a decentralised system without a centralised bank or country to verify the legitimacy of transactions. Doing so decreases government influence and prevents corruption and financial instability. Secure cryptocurrencies make it nearly impossible to counterfeit or **double-spend** currency. The **ledgers** are distributed across a **P2P network** and uses a **distributed consensus** algorithm. |
| Digital Signature | Cryptographic method to bind a user to digital data. Verifies the identity of the sender. Using the **hash** of the data and private key of the sender (**Asymmetric key cryptography**), a digital signature is created and appended to the message to form a signed message. The receiver uses the sender’s public key and digital signature to generate a hash. If it matches the hashed original data, it is valid, and the sender is verified.  Digital signature guarantees authentication, **non-repudiation**, and integrity. |
| Non-repudiation | A property in information security in which the sender is unable to deny the authorship or validity of the signed message with a **digital signature**. It serves as a proof of integrity and origin of data. |
| Transaction Pool | Collection of validated (by **digital signature**) but unconfirmed (by **mining**) transactions. |
| Key-pair Generation | Used in asymmetric key cryptography to generate 2 mathematically linked numbers known as public and private keys. Can be used for **digital signature** verification. |
| PuTTYgen | RSA **key-pair generation** software which can be used for authentication in **asymmetric key encryption**. It generates random key-pairs in which **entropy** is the measure of true randomness. |
| Entropy | The measure of true randomness of a system. A n-bit number from a perfect random number generator has n bits of entropy. A high entropy is important for cryptographic uses as a system with low entropy would make guessing the output easy. |
| Self-Referential Data Structure | Data structure which contains a pointer to another structure of the same type (e.g. Linked List). A **block** in a **blockchain** is a self-referential data structure. |
| Peer-to-Peer Network | P2P, unlike client-server, is a distributed application architecture in which processes are decentralised. |
| Block | Stores transactions similar to a **ledger**. Each block consists of a **block** **header** and record of transactions using a **Merkle tree**. Permanent and cannot be removed from the **blockchain**. |
| Genesis Block | First block in a blockchain. Previous hash set to 0 (No data processed before this block). Serves as a verification – to ensure all blockchains start from genesis block. |
| Block Header | Contains metadata about the block – Previous block/previous block header’s **hash**, sender and receiver’s address, **Merkle root**, timestamp, **nonce**, **difficulty level** and **block height**. |
| Block Height | Number of blocks before current block |
| Blockchain | Growing list of **blocks** linked together using cryptography. |
| Cryptographic Hash Function | **One-way** mathematical cryptographic function which converts any input into a unique encrypted output of fixed length known as a hash. Cryptographic hash functions are more secure as they are **deterministic**, **non-invertible** and **collision resistant**. Examples include MD5, SHA1 and SHA2 (**SHA256**). |
| One-way function | Easy to compute output for every input but difficult to find the corresponding inputs given only the output |
| Determinism | For any given input, the hash function should produce the same result/output. This property is used to verify a specific input. |
| Non-invertibility (Pre-Image Resistant) | Output of the hash function must not reveal any information about the input. It must be nearly impossible to reverse the output to obtain the input. Having a fixed output length also prevents any correlation to be made between the output length and the input. |
| Collision Resistance | It must be nearly impossible for 2 different inputs to produce the same output hash. A hash function can never be entirely collision resistant as there are more input combinations due to varying lengths and a fixed output length. |
| SHA256 | **Cryptographic hash function** built using the Merkle-Damgård structure which produces a 256-bit hash digest (32 digits in ASCII, 64 digits in HEX) |
| Merkle Tree | A tree data structure in which each leaf node is a **hash** of individual transactions. Non-leaf nodes are hashes of the sum of multiple (usually 2) children. The top node’s hash in the Merkle tree is the Merkle root. |
| Merkle Proof | Method to verify if a transaction exists in a **block** using the **Merkle tree**. Shortens the verification process because instead of checking each transaction in a list, only log(n) hashes a required to prove a transaction is included in a **block** (Example below). When the final hash matches the **Merkle root** stored in the **block header**, it is verified. |
| Proof of Work | Proof of work must be done to add a block to the blockchain. **Miners** are required to compete against each other to complete transactions on the network. This process is known as **mining**. Once mining is completed, the block is added. |
| Miner | Miners validate new transactions, group them into **candidate** **blocks** and record them on the **blockchain**. Responsible for **mining**. |
| Candidate Block | Temporary block in which **miners** are attempting to **mine**. To participate in the mining competition, **miners** must create a candidate block using transactions from the **transaction pool**. If the candidate block is validated through **proof of work**, it is added to the **blockchain** and the block reward is given to the miner. |
| Mining | Involves finding an input, known as a **nonce**,which when **hashed** satisfies the **difficulty level** in the **block header** (e.g. a hash that starts with 5 zeros). Once this **hash** is found, a small amount of **cryptocurrency** is rewarded, typically to compensate for computational costs. The new **block** is appended to the **blockchain** once **distributed consensus** is reached before appending the new **block**. |
| Nonce | Number only used once. A number found during **mining** that when hashed, satisfies the **difficulty level** in the **block header**. |
| Distributed Consensus | Validates data among nodes in a distributed system. In **mining**, consensus occurs at more than 50.5% agreement assuming 0 network latency. An increase in network latency would increase the percentage required for consensus to be attained. Once consensus is attained for the **nonce** found, the new **block** may be added by the **miner** and the **miner** is rewarded with **cryptocurrency**. |
| 51% Attack | Occurs when a group of **miners** who control a total of more than 50% of the network’s hash rate of computing power. By doing so, these miners can always find the **nonce** faster than other miners, preventing other miners from completing blocks. Having control more than 50% of the computing power also makes the **blockchain** susceptible to the **double spend problem**.  Chance of this occurring is low as it is difficult once a large enough network has been established |
| Double Spend Problem | When an amount of **cryptocurrency** is spent twice. Occurs typically during a **51% attack**. This is done by creating a false **blockchain**. Cryptocurrency is spent and validated on the real blockchain. Cryptocurrency is then spent on the falsified chain. As the spender owns 51% of the computing power, this false blockchain is validated. The cryptocurrency “spent” on real world commodities on the real blockchain is invalidated and the falsified blockchain becomes the “legitimate” blockchain. |
| Takeover Attack | Attacks on a blockchain done by **miners** whose goal is not to simply get monetary rewards, but to destabilize the **blockchain**. This is done by subverting the **proof of work**. One example of a takeover attack is the **51% attack**. |
| Difficulty level | A number that regulates how time taken for **miners** to find the **nonce** and add new **blocks** to the **blockchain**. Difficulty level is incremented after adding a certain number of **blocks** depending on time taken. |
| Immutability | Unchangeability of transactions. This property makes it close to impossible for 3rd parties to manipulate, replace or falsify data on a blockchain, thus maintaining data integrity. |

## Merkle Proof



* Recall that each parent is the hash of the sum of its children
* Knowing the hashes in green allows the verification of transaction in pink

# Broader Questions

## What Makes Blockchain Secure

* 2 key properties: **Immutability** and **Distributed** **Consensus**
* Cryptography: **Hashing**
  + Used in **Digital Signature, Block Header, Merkle Tree, Proof of Work**
* Decentralized System
  + A single compromised node/**miner** cannot influence the whole **P2P network**
  + Inherent vulnerability to double spend problem
    - Comes in the form of 51% attacks
    - When community is large enough, difficult to control more than 50% of a network’s computing power

## Evaluation of Blockchain

### Advantages

* Distributed
  + See **distributed consensus**
  + Resistant to technical failure and malicious attacks
* Stable
  + Confirmed transactions are almost impossible to reverse
* Does not require trust
  + Verified by **miners**
  + Requires **distributed consensus**

### Disadvantages

* 51% Attacks
* Data and protocol modification
  + Hard fork is required in which previous versions become incompatible
  + New **blockchain** is created
* Private key ownership
  + Asymmetric cryptography
  + Lose private key 🡪 Money lost
  + Cannot be retrieved
* Inefficient
  + **Proof of work**
  + Only successful **miner’s** block is validated, other miner’s effort wasted
* Storage
  + As **blockchain** length increases, more memory taken
  + E.g. bitcoin 200GB

## Application of Blockchain

* Cryptocurrency
* Accounting
* Voting
* Quality Assurance (e.g. Food)
  + Identify origin, batch information, other food safety details

## Social and Ethical Issues

* Environment
  + **Mining** requires high energy costs
  + E.g. Bitcoin alone contributes 0.3% of global energy usage
* Privacy: Criminal Activity
  + Anonymity – no link between accounts and personal identity

<Anything about advantages, issues, ethics>

# Hacks Links

Original Bitcoin Paper: <https://bitcoin.org/bitcoin.pdf>

Bitcoin Protocol: <http://www.michaelnielsen.org/ddi/how-the-bitcoin-protocol-actually-works/>

3Blue1Brown Blockchain Video: <https://www.youtube.com/watch?v=bBC-nXj3Ng4&vl=en>

Blockchain Visualisation: <https://www.youtube.com/watch?v=_160oMzblY8>