

# Blockchain Technology SoS Reading Project

Pagoti Hemanth Naidu - 210050117

Mentor: Shikhar Agrawal

may 2023

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Demand & Supply of Bitcoin	4
1.2	Double Spend Problem	4
<b>2</b>	<b>Motivation</b>	<b>4</b>
2.1	Blockchains in High Level	4
2.2	Applications	5
<b>3</b>	<b>cryptocurrency</b>	<b>5</b>
<b>4</b>	<b>Peer to Peer</b>	<b>5</b>
4.1	Consensus Protocol	5
4.2	Popular P2P versions	6
4.2.1	Napster	6
4.2.2	Gnutella	6
4.2.3	Distributed Hash Table	6
<b>5</b>	<b>Hash functions</b>	<b>6</b>
5.1	Basic properties of Hash functions	6
5.2	Random Oracle	7
5.3	Properties of Cryptographic Hash functions	7
5.3.1	Collision Resistance	7
5.3.2	Hiding	7
5.3.3	Puzzle Friendliness	7
5.4	Proof of Work	7
<b>6</b>	<b>SHA256 and Merkle Trees</b>	<b>8</b>
6.1	Advantages of Merkle Trees	8
6.2	Proving to Light Client	9
6.3	Full Client using Merkle Trees	9
<b>7</b>	<b>Bitcoin Transactions</b>	<b>9</b>

7.1	Block Header contents . . . . .	9
7.2	Digital Signatures . . . . .	9
7.2.1	Generation of shared, public key pair . . . . .	10
7.2.2	sign the message . . . . .	10
7.2.3	verify the signature . . . . .	10
7.3	ECDSA . . . . .	10
7.4	Transactions . . . . .	10
7.4.1	Structure of bitcoin transactions . . . . .	10
7.4.2	Transaction fee . . . . .	11
7.4.3	Rules . . . . .	11
7.4.4	Mining . . . . .	11
7.4.5	Coinbase transaction . . . . .	11
7.4.6	Double spend prevention . . . . .	11
<b>8</b>	<b>Target Threshold . . . . .</b>	<b>12</b>
8.1	Network propagation . . . . .	12
8.2	How threshold is set . . . . .	12
8.3	Verification . . . . .	12
8.3.1	Rules for verification . . . . .	13
8.4	Deal with forks . . . . .	13
8.4.1	Chain weight . . . . .	13
<b>9</b>	<b>Confirmation Time . . . . .</b>	<b>13</b>
9.1	Double spend attack . . . . .	14
9.1.1	Inter arrival time of blocks . . . . .	14
9.1.2	Exponential distribution . . . . .	14
9.1.3	Poisson arrival process . . . . .	15
9.2	51% Attack . . . . .	15
<b>10</b>	<b>Satoshi's analysis . . . . .</b>	<b>15</b>
10.1	Meni Rosenfeld discrete analysis . . . . .	16
<b>11</b>	<b>Selfish mining . . . . .</b>	<b>18</b>
11.1	Majority is not enough . . . . .	18
11.2	Markov chain . . . . .	19
11.3	Discrete time markove chain analysis . . . . .	19
11.4	Rewards . . . . .	20
<b>12</b>	<b>Proof Of Stake . . . . .</b>	<b>21</b>
12.1	Sybil attack . . . . .	21
12.2	Nothing at stake . . . . .	21
12.3	Long range attack . . . . .	22
12.4	Initial distribution problem . . . . .	22
12.5	Bribe attack . . . . .	22
12.6	Pre computing attack . . . . .	23

<b>13 Hybrid PoW and PoS consensus . . . . .</b>	<b>23</b>
--	-----------

# 1 Introduction

Bitcoin was introduced in 2009 by Satoshi Nakamoto in response to the 2008 global financial crisis and the need for a decentralized, peer-to-peer electronic cash system. Nakamoto's vision was to create a currency free from central control, censorship-resistant, and capable of secure, borderless transactions.

## 1.1 Demand & Supply of Bitcoin

Bitcoin's supply is regulated by Satoshi Nakamoto's algorithm, limiting the total number of bitcoins that can ever exist to 21 million. The demand for bitcoin is determined by market forces and influenced by factors such as investor sentiment and adoption. Balancing supply and demand is essential for maintaining stability in the Bitcoin market and mitigating the risk of inflation.

## 1.2 Double Spend Problem

Satoshi Nakamoto's design of Bitcoin ensures that money can only be spent once through the concept of ownership and peer-to-peer networks. With public keys, everyone can know how much each owner possesses while maintaining anonymity. Hash functions are used for encoding transactions and mining purposes, while proof of work enables Bitcoin to function as a cryptocurrency. The tamper-proof nature of blockchain ensures that once transactions are added, they cannot be modified, and each block in the blockchain contains the hash of the previous block. Smart contracts are self-executing contracts with the terms of the agreement directly written into code, automating and enforcing the agreed-upon conditions without the need for intermediaries. They enable secure and transparent transactions in various applications, from finance to supply chain management.

# 2 Motivation

## 2.1 Blockchains in High Level

- Blockchain utilizes a tamperproof data structure that ensures data integrity and security.
- Starting with a genesis block, each subsequent block in the chain holds information in the form of bitstrings or binary representations.
- Once data is added to a block and appended to the chain, it becomes immutable and cannot be removed or altered.
- This feature of blockchain, along with its decentralized and distributed nature enables various applications such as cryptocurrency, smart contracts and trust.

## 2.2 Applications

- **Governance:** Land records, health records, transportation data, virtual currency, electronic wills, passport, identification, etc.
- **Commercial:** supply chains, auctions, gaming, sale of music, financial services, smart grid etc.
- **Disruptive:** Cryptocurrencies, Initial Coin Offerings

## 3 cryptocurrency

Bitcoin has no central trusted authority, cryptography brings in trust and peer-to-peer networks provide decentralization. Bitcoin satisfies the characteristics of acceptability, portability, durability, divisibility, and fungibility.

- **Acceptability:** Bitcoin is widely accepted as a form of payment and store of value.
- **Portability:** Bitcoin can be easily transferred and accessed across geographical boundaries.
- **Durability:** Bitcoin's digital nature makes it resistant to physical damage or deterioration.
- **Divisibility:** Bitcoin is divisible into small units, allowing for precise transactions of any value.
- **Fungibility:** Bitcoin units are interchangeable, meaning that each unit holds the same value and can be exchanged without distinction.

Each transaction is broadcasted among all so that double spend is avoided.

Consensus protocols are mechanisms used in blockchain networks to achieve agreement among participants on the validity of transactions and the order in which they are added to the blockchain and this is also known as proof of work.

## 4 Peer to Peer

A permissionless, distributed system allows an arbitrary number of participants to join or leave at any time, enabling seamless peer-to-peer transactions where individuals have the freedom to pay anyone within the network, fostering inclusivity and decentralization.

### 4.1 Consensus Protocol

- To establish consensus in a protocol, cryptography is employed to secure transactions and validate the integrity of the data.

- While a peer-to-peer network enables direct communication and information sharing among participants.
- The proof-of-work mechanism adds a layer of computational effort to validate transactions and prevent malicious behavior, and Merkle trees provide an efficient and verifiable way to store and verify the integrity of large sets of data within the blockchain.
- Together, these components form the foundation for a robust and trustless consensus protocol.

## 4.2 Popular P2P versions

### 4.2.1 Napster

Napster was one of the earliest peer-to-peer networks that facilitated file sharing. It used a centralized server to maintain an index of available files, allowing users to search and download files directly from each other, enabling widespread file sharing but faced legal challenges due to copyright concerns.

### 4.2.2 Gnutella

Gnutella is a decentralized peer-to-peer network where participating nodes connect directly with each other. It operates on a query flooding mechanism, where search queries propagate across the network, and peers respond with matching files, ensuring a distributed and scalable file sharing system.

### 4.2.3 Distributed Hash Table

DHT is a decentralized peer-to-peer network architecture that provides efficient lookup and storage of key-value pairs. It operates by partitioning the hash space among participating nodes, enabling direct lookup and storage of data based on keys. DHTs provide fault tolerance, scalability, and efficient resource utilization in distributed systems and are commonly used in applications like BitTorrent and distributed storage systems.

## 5 Hash functions

Each block in the blockchain contains hash of the previous block, the hash is recognised as an unique ID.

### 5.1 Basic properties of Hash functions

- Input to the hash function can be of any length.
- Output of the hash function should be of fixed size.
- Hash should be efficiently computed.

## 5.2 Random Oracle

It is like a black box that takes input and generates a random output based on that input. It provides a consistent and unpredictable response to any query. Random oracles are often used to model idealized cryptographic functions and serve as a building block for designing secure protocols and systems.

However, in practice, real-world hash functions are used as a substitute for random oracles.

## 5.3 Properties of Cryptographic Hash functions

### 5.3.1 Collision Resistance

Collision resistance, in the context of cryptographic hash functions, means that it is computationally infeasible to find two different inputs that produce the same hash output. It ensures that a small change in the input will result in a significantly different hash value, making it difficult to forge or manipulate data without detection.

The birthday paradox states that in a group of relatively few people, the probability of two individuals sharing the same birthday is surprisingly high.

### 5.3.2 Hiding

suppose  $r$  takes values from  $r_1, r_2, r_3, \dots, r_n$  with probability  $p(r_1), p(r_2), p(r_3), \dots, p(r_n)$ .

The minimum entropy is defined as follows

$$\min - \text{entropy} = \min_{i=1}^n -\log(x_i)$$

where a secret value  $r$  is chosen from a probability distribution with high min-entropy and combined with another value  $x$  to compute the hash  $h(x||r)$ , it is infeasible to find the original value of  $x$  given only the hash  $h(x||r)$ .

### 5.3.3 Puzzle Friendliness

Hash function is puzzle friendly if for every  $N$  bit output  $Y$  if  $K$  is chosen from a probability distribution with high minimum entropy, then it is infeasible to find  $X$  such that

$$H(K||X) = Y$$

in time significantly less than  $O(2^N)$ .

## 5.4 Proof of Work

- Miner selects pending transactions.
- Miner combines hash of previous block, nonce, and transactions.
- Nonce is of 16 bits, miner can include any number of transactions.

- Miner adjusts nonce until the resulting block hash meets the threshold for network consensus, creating the next block.
- Once a valid block is created it should be broadcasted to other miners for verification.

## 6 SHA256 and Merkle Trees

Here's a high-level overview of how SHA-256 works:

- **Padding:** The input message is padded with length of message to ensure it meets certain requirements for the SHA-256 algorithm. Padding includes adding bits to the message to make it a multiple of 512 bits.
- **Initialization:** An Initialization Vector (IV) is a predetermined set of initial values used to initialize the compression process in certain compression algorithms, ensuring consistent and reproducible compression results for the same input data.
- **Processing:** The padded message is divided into blocks of 512 bits, and the algorithm processes each block sequentially. The processing involves a series of logical and bitwise operations, including message expansion, data mixing, and bitwise operations such as AND, OR, and XOR.
- **Compression:** Each block is compressed using a combination of logical and bitwise operations, updating the state of the hash values.
- **Finalization:** Once all the blocks have been processed, the resulting hash value is obtained by concatenating the updated hash values.

Miners use Merkle trees (also known as Merkle hash trees) as an efficient and secure way to summarize a large set of transactions within a blockchain. These trees play a crucial role in the creation of the next block in the blockchain. Let's walk through the process:

- Miners gather valid transactions and sort them. They hash each transaction individually and combine the hashed values in pairs until a single root hash, known as the Merkle root, remains at the top of the tree.
- The Merkle root is included in the block header along with other block information. Miners attempt to find a nonce value that, when hashed with the block header, produces a hash that is less than the threshold.

### 6.1 Advantages of Merkle Trees

- Merkle trees enable efficient verification and tamper detection, making it difficult for a miner to modify a specific transaction without changing the Merkle root.
- If a miner want to add or Modify a transaction while solving the puzzle it can be done in  $O(\log(M))$  operations where M is no of transactions included.



- Light clients(New comer of Bitcoin) require fewer resources, enabling users with limited capabilities to participate in blockchain networks.

## 6.2 Prooving to Light Client

There is a situation that A is a light client and B need to prove that one of his transaction happened to C to A then B can provide A with a compact proof known as a Merkle proof.

This proof consists of the transaction, its corresponding Merkle path, and the Merkle root. A can then verify the validity of the transaction by using the Merkle proof to trace the transaction's inclusion in the Merkle tree and comparing the computed Merkle root with the one stored by A.

## 6.3 Full Client using Merkle Trees

If a full client want to discard any of the transaction he stored he can still prove that remaining transactions belonging to the same merkle tree which may not be happened if he stores in a naive way i.e concatenating all the transactions.

# 7 Bitcoin Transactions

## 7.1 Block Header contents

The block header in a blockchain typically contains the following elements:

- **Previous Block Hash:** Hash of the preceding block in the blockchain.
- **Merkle Root:** Hash of all the transactions included in the block.
- **Timestamp:** The time when the block was created, which is used after to find appropriate threshold.
- **Nonce:** A number used in the mining process to find a valid block hash.
- **threshold:** 4 Bytes are used to represent threshold also known as bits, to save space instead of storing 256 bits.

$$bits = b_1b_2b_3b_4$$

$$Threshold = b_2b_3b_4 * 256^{b_1-3}$$

## 7.2 Digital Signatures

Digital Signatures are hard to forge, easy to create and easy to verify. These signatures use the below three algorithms to complete the need:

### 7.2.1 Generation of shared, public key pair

This uses GenerateKeys algorithm which takes key size as the parameter and generates a key pair in which shared key is kept secret and public key is known to everyone.

### 7.2.2 sign the message

This uses Sign algorithm which takes message and shared key as the algorithm and encrypts the message, in this case message is typically a transaction.

### 7.2.3 verify the signature

This uses Verify algorithm which takes the message, signature and cipher text and verify the signature, this is done by everyone to verify transactions.

## 7.3 ECDSA

Elliptic Curve Digital Signature Algorithm is a cryptographic algorithm used for generating and verifying digital signatures, in which public key is 512 bits, shared key is 256 bits, signature is 512 bits and the message should be 256 bits.

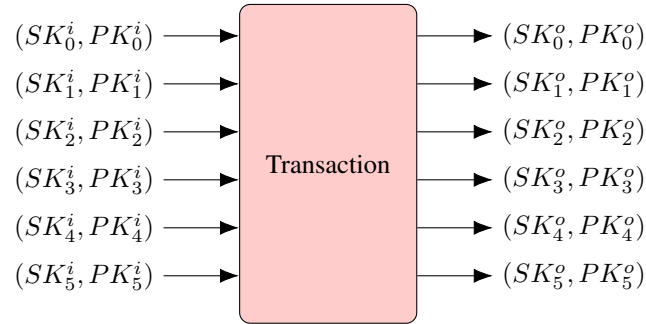
Hash the message using SHA256 as many of the messages are not of 256 bits.

## 7.4 Transactions

Assume A( $SK_A, PK_A$ ) want to pay B( $SK_B, PK_B$ ), A has to make a transaction himself and sign it with his shared key as discussed above.

Now the problem is how a miner verify this transaction is valid or not.

### 7.4.1 Structure of bitcoin transactions



Inputs are taken from earlier outputs of Bitcoin transactions, which consists hash of the transaction and index of the Block in which the transaction is present and some other feilds. Output pay Bitcoin to different public keys, each input may correspond to multiple outputs. Reaming Bitcoin can be payed to himself by generating a new key pair, the transactions should be signed by shared keys of all inputs of the transaction.

#### **7.4.2 Transaction fee**

Transaction fees in Bitcoin are small amounts of Bitcoin paid by the sender of a transaction to incentivize miners to include and confirm the transaction on the network. The fee amount varies based on network congestion and transaction size.

If a sender wants to pay 1% of his output as Transaction fee he decreases the output by 1%.

#### **7.4.3 Rules**

- Output should not be negative in a transaction.
- sum of inputs must be greater than or equal to sum of outputs in a transaction.
- Whoever miner solves the puzzle can claim all the fee of the transactions he included and the mining fee.

#### **7.4.4 Mining**

Miner selects some of the pending transactions, compute merkle root and adjusts nonce until the resulting block hash meets the threshold for network consensus, creating the next block.

#### **7.4.5 Coinbase transaction**

This is the first transaction in which the miner claims the transaction fee and the mining reward, this transaction will not have any inputs it can create bitcoin i.e mining reward, which decreases exponentially so that no inflation occurs. Miner can claim his reward if he wins the game by creating next block.

#### **7.4.6 Double spend prevention**

There are set of rules which should be followed by miners so that double spend can be prevented these follows:

- Any output of a transaction should be only spend once, such outputs are called UTXO(Unspent Transaction Output).
- If any miner includes invalid transaction and succeeded in creating new block every other miner should discard that block, if the block is in some chain the chain is also discarded.
- When a miner creates a block with all valid transactions every other miner should delete the transaction which are in the new block from their respective transaction pool.

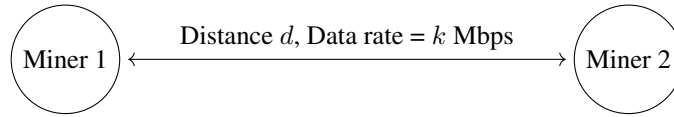
From the above miner can also change some arbitrary fields in the coinbase transaction to complete the proof of work.

The length of block header is 80 bytes, Satoshi managed to keep it small.

## 8 Target Threshold

If the threshold is set too small it takes miners so much time to complete the proof of work as it takes so many trails, if it is set too large then forks may appear instead of chain we will have trees. Target threshold should not be too large or small. Network delays while broadcasting block for verification should not be large because it may cause forks in the blockchain.

### 8.1 Network propagation



When miner 1 computes a block and wants to transmit to miner 2 the delay is  $\frac{d}{c} + \frac{b}{k}$ , where  $b$  is the length of block size. Miner 2 verifies the block and broadcasts it. So the total time taken for end-to-end propagation of the block in peer-to-peer network is given below:

$$\sum_{i=1}^n \frac{d_i}{c} + \frac{b}{k_i} + p_i$$

where  $p_i$  is the time taken for verification of the block, and the links are selected by the spanning tree algorithm for avoiding multiple transfers.

### 8.2 How threshold is set

There should be no situation like some miner solves proof of work while other miner broadcasts his block, it may lead to forks and double spend. So threshold is chosen such that each block takes 10 minutes to create.

The threshold cannot be constant because as the users of bitcoin increase the hashing power increases, so Satoshi decided to update threshold every 2 weeks based on hashing power. Users use ASIC hardware for solving proof of work. This 2 weeks is defined in terms of number of blocks, in 2 weeks roughly 2016 blocks were created.

So threshold is updated after every 2016 blocks as follows:

$$threshold = threshold * \frac{T}{1209600}$$

$T$  is Time taken for creation of prev 2016 blocks in sec,  $T$  can be found using timestamp in block header as follows:

$$T = timestamp(B_{2016n-1}) - timestamp(B_{2016(n-1)})$$

### 8.3 Verification

Ideally miner should follow **NTP** for the timestamp while solving proof of work.

### 8.3.1 Rules for verification

Miner accepts the received block if:

$$t_k > \text{median}(t_{k-1}, t_{k-2}, t_{k-3}, \dots, t_{k-11})$$
$$t_k < 2\text{hours} + \text{Networkadjusttime}$$

Network adjust time is defined as sum of miner local time according to NTP plus the median of the offsets with local times of his fellow peers.

## 8.4 Deal with forks

Satoshi's first rules were to select longest chain if forks are present, if multiple longest chains were present should select the one with less timestamp.

### 8.4.1 Chain weight

If some attacker manages to create longest chain privately and broadcast after he may kill the honest blocks to avoid this chain weight was used.

Chain weight uses the sum of expected work done for all the blocks in the chain, if it is built using fake timestamps he may get caught. expected work for creating  $i$ 'th block is calculated as follows:

$$B_i = \lfloor \frac{2^{256}}{\text{Threshold} + 1} \rfloor$$

expected work for creating entire chain is calculated as follows:

$$\sum_{i=1}^n B_i$$

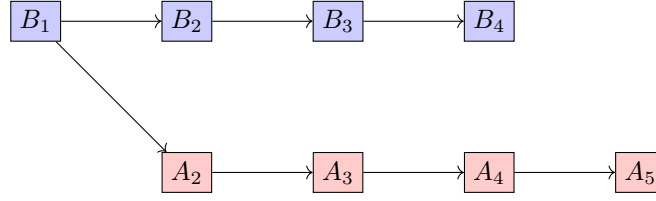
## 9 Confirmation Time

The confirmation time in blockchain refers to the amount of time it takes for a transaction to be validated and included in a block on the blockchain. The specific confirmation time can vary depending on the blockchain network and its consensus mechanism.

**Zero Confirmation:** A transaction that has been broadcasted but not yet added to a block or confirmed by the network.

**First Confirmation:** The first block in which a transaction is included, indicating its confirmation on the blockchain.

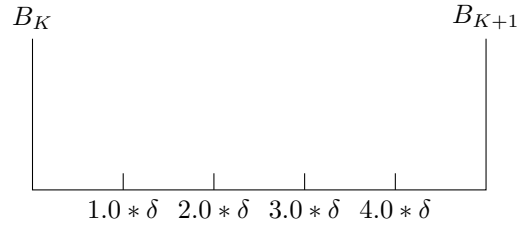
**Second Confirmation:** The second block in which a transaction is included, further validating its confirmation on the blockchain and increasing the level of confidence in its permanence.



## 9.1 Double spend attack

This attack was analysed by Satoshi himself, the above diagram shows the attack after  $B_1$ , so attacker creates another chain privately after  $B_1$  to create double spend and kill the chain created by honest miners.

He creates until the other user confirms, i.e if the user waits till 3rd confirmation and attacker broadcast the transaction between  $B_1$  and  $B_2$ , so the user checks if it is in  $B_4$  are not, so attacker releases the chain after  $A_5$  is created before  $b_5$ , otherwise he may not.



### 9.1.1 Inter arrival time of blocks

We want the distribution of separation time between each block, let us say that we divided time into intervals of  $\delta$  each and  $\beta$  is hashing power available.

The probability that a block is created in the time interval  $\delta$  is  $\beta\delta$ , let us denote the inter arrival time as  $I$  so,

$$P[I = n\delta] = (1 - \beta\delta)^{n-1} \beta\delta$$

$$P[I > n\delta] = (1 - \beta\delta)^n$$

set  $n\delta$  as  $x$

$$P[I > x] = \left(1 - \frac{\beta * x}{n}\right)^n$$

as  $n$  goes to  $\infty$

$$P[I > x] = e^{-\beta * x}$$

### 9.1.2 Exponential distribution

The PDF is

$$P[I = x] = \beta * e^{-\beta * x}$$

The mean or expected value of the distribution is

$$E[x] = \frac{1}{\beta}$$

This distribution satisfies memoryless property

$$P(X > t + s | X > t) = P(X > s)$$

This property states that the probability of the random variable X exceeding a given threshold t+s, given that it has already exceeded t, is equal to the probability of X exceeding s.

From this we can conclude that when the transaction is broadcasted to the miners we should wait for  $\frac{1}{\beta}$  which is roughly 10 minutes.

### 9.1.3 Poisson arrival process

The number of blocks created in an interval of T is suprisingly poisson distribution given as

$$P(A_T = n) = \frac{e^{-\beta * t} (\beta * t)^n}{n!}$$

## 9.2 51% Attack

let p be the fraction of hashing power with honest miners,  $q = 1 - p$  be the hashing power with attackers, if  $q > 0.5$  then attacker can always kill the chain of honest block. This attack is known as 51% attack.

The naive solution of this attack is to create checkpoints is that select a block which is deep enough in the chain must be present in the chain always, checkpoints are updated during a period of time.

It is very difficult to achieve more than 50% of hashing power. if attacker creates after time y and honest miner after x then probability of attacker winning the game is:

$$\begin{aligned} P[y < x] &= \int_0^\infty (1 - e^{-\beta * q * x}) \beta * p * e^{-\beta * p * x} dx \\ &= \int_0^\infty \beta * p * e^{-\beta * p * x} dx - \int_0^\infty \beta * p * e^{-\beta * x} dx \\ &= 1 - p \\ &= q \end{aligned}$$

## 10 Satoshi's analysis

We want to know the number of confirmations user should wait such that his transaction will be in the blockchain forever.

When the transaction is about to enter the block chain attacker may be ahead or behind the honest miners.

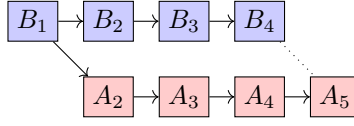


Figure 1: attacker is ahead of honest miners

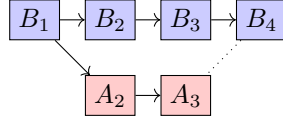
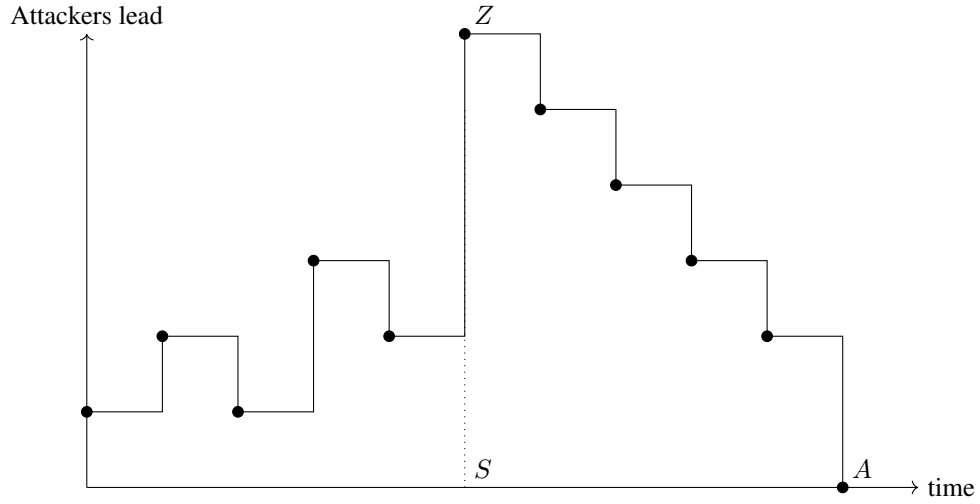


Figure 2: attacker is behind the honest miners

If attacker is behind the honest miners he should try create more blocks to kill the honest chain, if is ahead he will be wait until the transaction entered the block and releases the chain.

Satoshi assumed that network delay is zero and observed the attacker's lead with time.

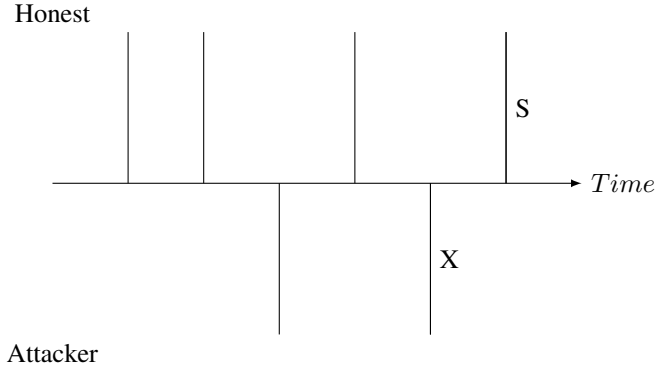


From the above picture  $Z$  can be in the range  $(-\infty, S)$ , where  $S$  is indicating  $S$ 'th confirmation, we also have to see after  $S$  if he can win at some point like  $A$  in above in future.  $S$ ,  $A$  location is variable it can happen anywhere and it affects the value of  $Z$ .

### 10.1 Meni Rosenfeld discrete analysis

We will be marking points at which blocks were created and observe it when  $S$  number of blocks are created by honest miners.





If the honest miners are ahead of  $Z$  blocks at the  $S$ 'th confirmation,

$$X = S - Z$$

we have to calculate the probability that honest miners have lead  $Z$  at  $S$ 'th confirmation,

$$P_{Z,S} = \text{Prob}((S - X = Z) | (\text{Honest} = S))$$

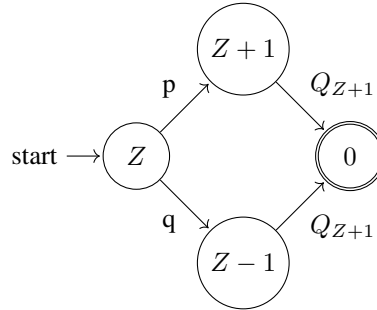
Also the attacker catching up the lead  $Z$  as  $Q_Z$ .

Probability that attacker catches up after  $S$  honest blocks,

$$R_S = \sum_{-\infty}^S P_{Z,S} * Q_Z$$

$$P_{Z,S} = \binom{2 * S - Z - 1}{S - 1} * P^S * Q^{S-Z}$$

For finding  $Q_Z$  we should perform random walk as there are many possibilities to catch up the lead, this can analysed by an simple automeata as below,



So for  $Z > 0$ ,

$$Q_Z = P * Q_{Z+1} + q * Q_{Z-1}$$

on solving,

$$Q_Z = \begin{cases} \left(\frac{q}{p}\right)^Z & \text{if } Z \geq 0 \\ 1 & \text{if } Z < 0 \end{cases}$$

The above  $Q_Z$  is only true if  $q < p$ , else  $Q_Z$  is always 1 which leads to 51% attack.

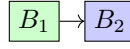


Figure 3: honest miners creates first

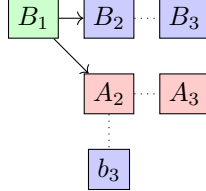


Figure 4: attacker and honest miners release simultaneously

## 11 Selfish mining

Selfish mining is a strategy where a miner intentionally withholds mined blocks to gain a competitive advantage and disrupt the fairness of the blockchain network. To perform selfish mining we should have an reasonable amount of hashing power.

### 11.1 Majority is not enough

This attack was analysed by Eyal and Siral around 2015.

If the block is created by honest miner before attacker creates a block, all the miners including attackers starts mining on this block as shown in figure 3.

If the attacker creates he waits and if the honest miner also creates some time after, if attacker has no lead he releases the block which leads all the attackers and some fraction of honest miners mining on his block, remaining fraction on honest block, as shown in figure 4. If  $B_3$  is created first attacker chain is killed, similarly if  $b_3$  or  $A_3$  is created the honest miners chain is killed.

So if attacker had a lead 2 or more he waits until lead becomes 1 and releases the chain and kill the honest chain, each attacker block gets lead but not the honest blocks in this case attacker releases the chain as  $B_4$  was created before  $A_6$  as shown in figure 5.

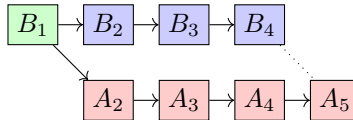
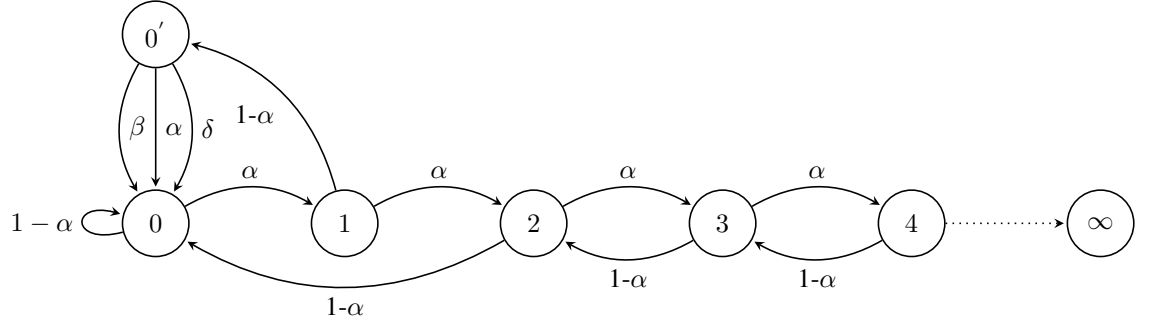


Figure 5: all the attacker blocks get into the chain

## 11.2 Markov chain

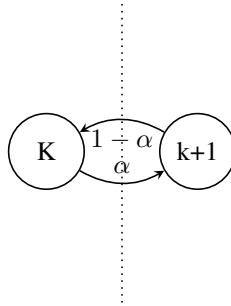


$$\beta = (1 - \alpha)(1 - \gamma)$$

$$\delta = (1 - \alpha)\gamma$$

According to the paper attacker gets one block in each  $(i, i + 1)$ , 2 blocks in  $(2,0)$  state change, honest miner may get 1,2 or 0 blocks in  $(0', 0)$  state change depending on who creates the block.

## 11.3 Discrete time markove chain analysis



From the above figure if we make an vertical cut the flow should be equal from both sides in steady state so for  $k \geq 2$ ,

$$P_k * \alpha = P_{k+1} * (1 - \alpha)$$

make cut between 1 and 0', 0 and 1,

$$P_{0'} = P_1 * (1 - \alpha)$$

$$P_0 = (P_1 + P_2) * (1 - \alpha)$$

on solving,

$$P_0 = \frac{\alpha - 2 * \alpha^2}{\alpha * (2 * \alpha^3 - 4 * \alpha^2 + 1)}$$

$$P_{0'} = \frac{(1 - \alpha) * (\alpha - 2 * \alpha^2)}{2 * \alpha^3 - 4 * \alpha^2 + 1}$$

$$P_1 = \frac{\alpha - 2 * \alpha^2}{2 * \alpha^3 - 4 * \alpha^2 + 1}$$

for all  $k \geq 2$ ,

$$P_k = \left(\frac{\alpha}{1 - \alpha}\right)^{k-1} * \left(\frac{\alpha - 2 * \alpha^2}{2 * \alpha^3 - 4 * \alpha^2 + 1}\right)$$

## 11.4 Rewards

$r_{pool}$  is defined as ratio of number of attacker blocks entered(A) int to final chain to the sum of all blocks generated by attacker( $A_0$ ) and honest miners( $H_0$ ).

$$r_{pool} = \frac{A}{A_0 + H_0}$$

$$r_{others} = \frac{H}{A_0 + H_0}$$

$$\frac{r_{pool}}{r_{pool} + r_{others}} = \frac{A}{A + H}$$

$$r_{others} = P_{0'} * (1 - \alpha) * \gamma + P_{0'} * (1 - \alpha) * (1 - \gamma) * 2 + P_0 * (1 - \alpha)$$

$$r_{pool} = P_{0'} * (1 - \alpha) * \gamma + P_{0'} * 2 * \gamma + P_2 * (1 - \alpha) * 2 + \sum_2^{\infty} P_i * \alpha$$

$$R_{pool} = \frac{A}{A + H}$$

$$R_{pool} = \frac{\alpha * (1 - \alpha)^2 * (4 * \alpha + \gamma(1 - 2 * \alpha)) - \alpha^3}{1 - \alpha * (1 + (2 - \alpha)) * \alpha}$$

Where  $\alpha$  is hashing power of attackers pool and  $\gamma$  is the fraction of honest miners who follow attackers after releasing attackers chain.

In the context of Bitcoin, a mining pool is a group of miners who come together to combine their computing power and resources in order to increase their chances of successfully mining new blocks and earning Bitcoin rewards. Here's how a mining pool typically operates:

1. Miners join the pool: Individual miners join a mining pool by connecting their mining hardware (such as specialized computers called ASICs) to the pool's mining server.
2. Solving hash puzzles: The mining pool's server assigns computational tasks to the connected miners, typically in the form of hash puzzles.
3. Work distribution: The pool's server distributes different variations of the hash puzzle to the connected miners.

4. Finding a block: Once a miner in the pool successfully solves a hash puzzle, they broadcast the solution to the pool's server.
5. Distribution of rewards: Rewards are distributed among pool participants based on the proportion of hashing power they contributed.
6. Continuous mining: Miners in the pool continue to receive new hash puzzles and work on solving them to find more blocks and earn rewards.

## 12 Proof Of Stake

### 12.1 Sybil attack

In a blockchain, a Sybil attack involves an attacker claims to have more hashing power than what he actually have.

In proof of work Sybil attack cannot be performed because someone cannot fake real time work done by them, but the problem with proof of work is it consumes more energy.

Proof of stake(PoS) is a consensus protocol unlike Proof of Work (PoW), which relies on miners solving complex mathematical puzzles, PoS selects validators to create and validate new blocks based on the amount of cryptocurrency they hold and "stake" as collateral.

Proof of work uses hash of previous block, nonce(IV), timestamp(T), merkle root of the transactions(MR) to solve the puzzle as follows,

$$Hash(Hash(B_k), IV, T, MR, ...) < Threshold$$

Proof of stake uses the hash of previous block, public key(a), balance with a, timestamp and merkle root is not included, but the threshold is varied for miners because people with more stake should have more chance, also timestamp should be near to local time.

$$Hash(Hash(B_k), a, T, ...) < Threshold * balance(a)$$

### 12.2 Nothing at stake

The "nothing at stake" problem is a concern in Proof of Stake (PoS) blockchains where validators have no financial disincentive to support multiple conflicting forks. Since PoS validators are not required to spend computational resources to mine blocks, they can potentially support all competing chains simultaneously. This lack of economic cost removes the incentive to converge on a single valid chain, leading to a lack of consensus. To mitigate this problem, PoS protocols implement penalties or slashing conditions to discourage validators from supporting multiple forks, ensuring network stability and security.

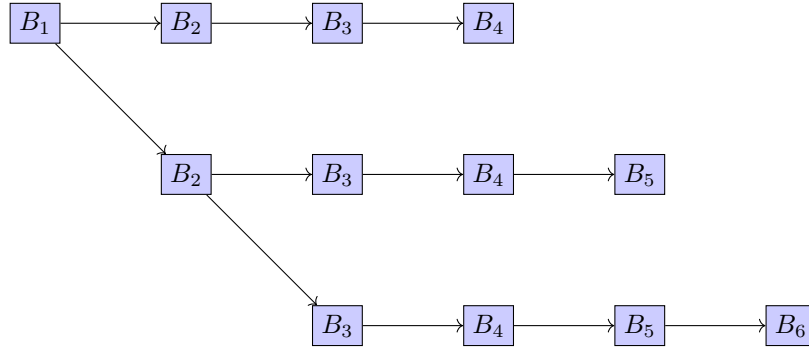


Figure 6: Nothing at stake attack

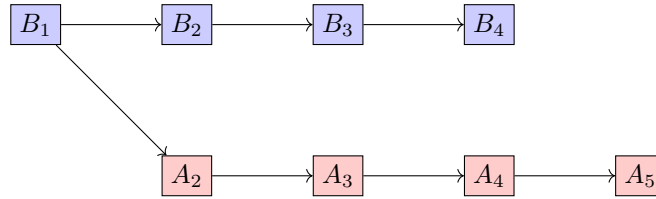


Figure 7: Long range attack killing blue chain after  $B_1$

### 12.3 Long range attack

A long-range attack is a security concern in blockchain systems, where an attacker attempts to rewrite the entire blockchain's history. By possessing a significant amount of stake or historical data, the attacker creates an alternative chain from an earlier point in time to surpass the current valid chain. This can invalidate previous transactions and compromise the integrity of the network as shown in figure 7. HAs stake goes up so attacker might approach real time in creating blocks i.e creating blocks at every 10 minutes and he can easily kill the entire chain easily.

### 12.4 Initial distribution problem

The initial distribution problem in Proof of Stake (PoS) refers to the challenge of fairly allocating cryptocurrency among participants at the launch of a PoS blockchain. Concentration of stake by early adopters can lead to centralization and disproportionate control. Strategies like fair token distribution and airdrops are used to promote decentralization and prevent dominance by a few stakeholders.

### 12.5 Bribe attack

A bribe attack is a type of attack where an adversary attempts to manipulate the behavior of participants in a blockchain network by offering them incentives or bribes. The

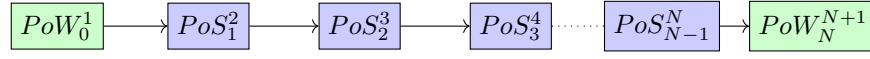


Figure 8: Hybrid solution

attacker aims to persuade participants to act in a way that benefits the attacker's interests. This attack can undermine the integrity and security of the blockchain consensus. Countermeasures include promoting transparency, strong governance mechanisms, and encouraging participants to act in the best interest of the network rather than personal gain.

### 12.6 Pre computing attack

A pre-computing attack in Proof of Stake (PoS) refers to a potential attack where an adversary tries to gain an unfair advantage by pre-computing a large number of block proposals or validations before they are actually required. By pre-computing blocks as he can change the transaction anytime in previous block, the attacker can quickly gain control over the blockchain and influence the consensus process.

## 13 Hybrid PoW and PoS consensus

Use proof of work but not too much of energy i.e don't use proof of work in every block, also use proof of stake for consensus protocol. The proposed hybrid solution is use PoW after every N blocks in between use PoS as shown in figure 8.

By using this the PoW energy per unit time is changed to  $\frac{M}{NT}$  from  $\frac{M}{T}$  where M is mining fee per block and T is average time per creating a block.