Blockchain Introduction

Plan of Attack

- What is a Blockchain?
- Understanding SHA256 Hash
- Immutable Ledger
- Distributed P2P Network
- How Mining Works (Part 1: The Nonce)
- 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

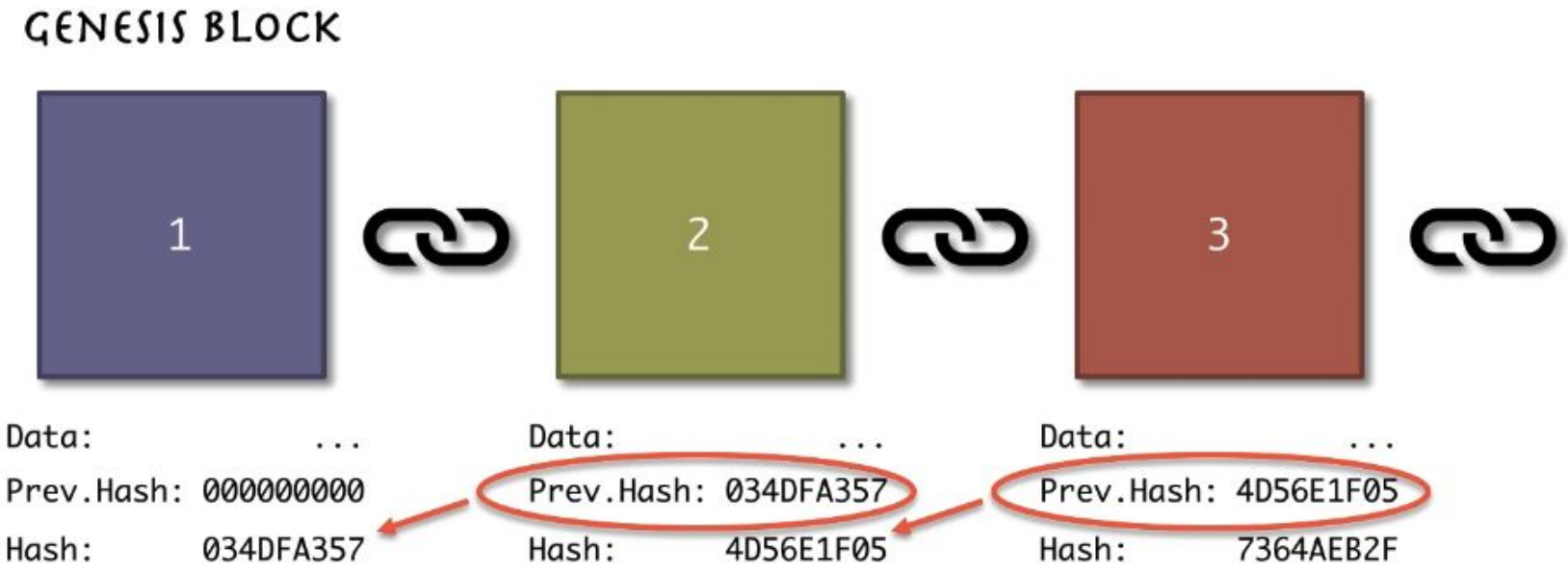
Block: #3

Data:
Kirill -> Hadelin 500 hadcoins
Kirill -> Ebay 100 hadcoins
Hadelin -> Joe 70 hadcoins

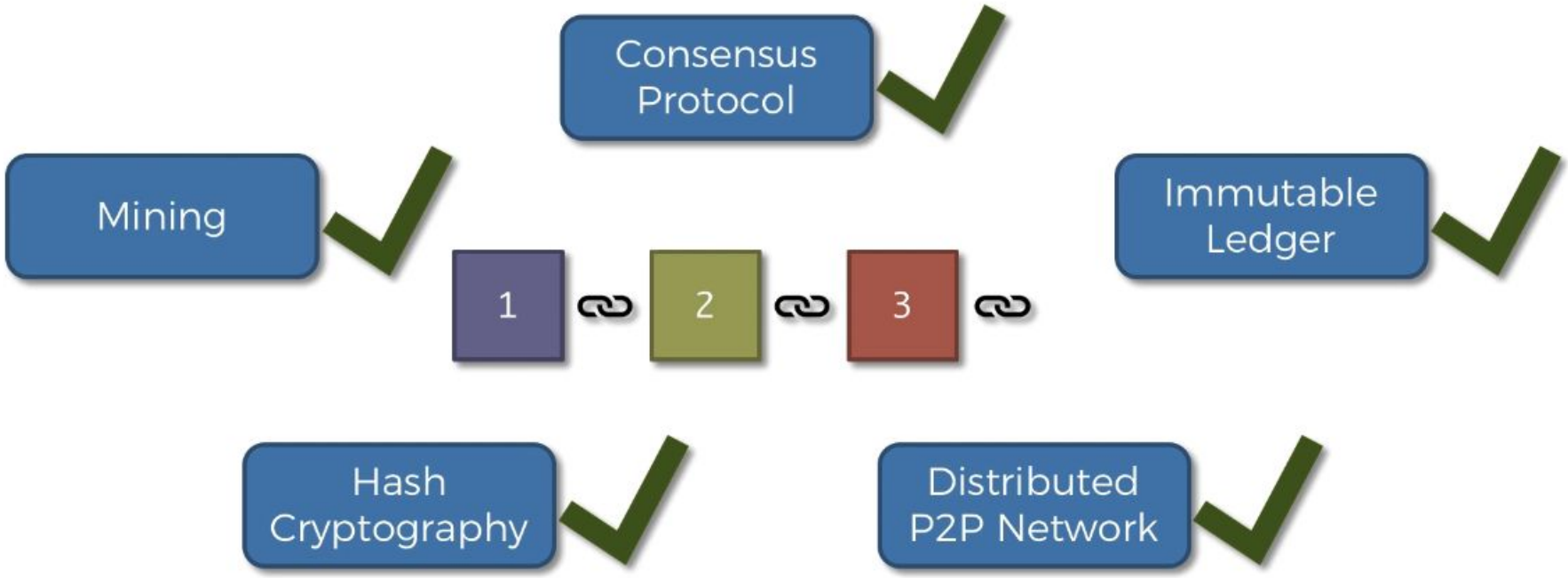
Prev.Hash: 0000DF2E57FB432A

Hash:

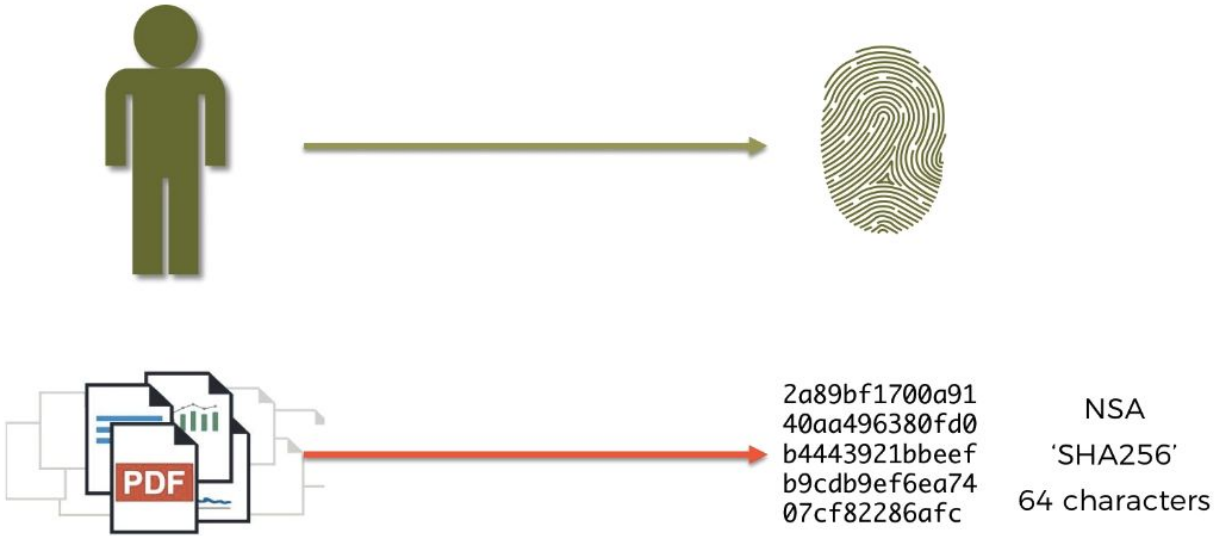
A chain of blocks



Introductory plan



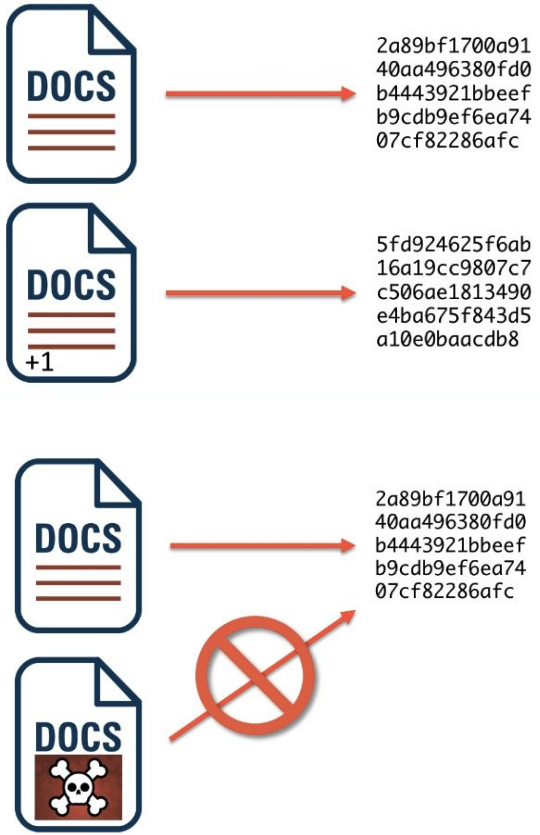
Understanding SHA256 Hash



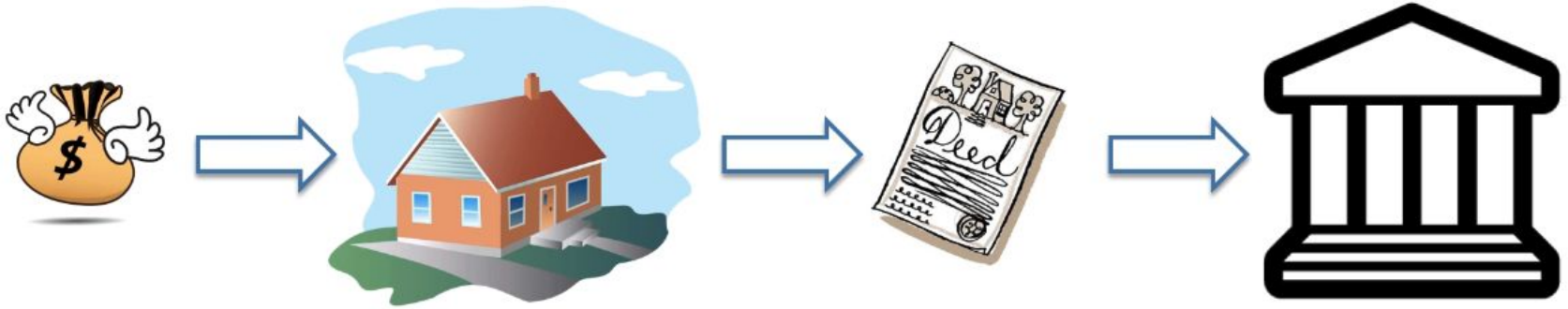
https://github.com/anders94/blockchain-demo/

The 5 requirements for Hash algorithms

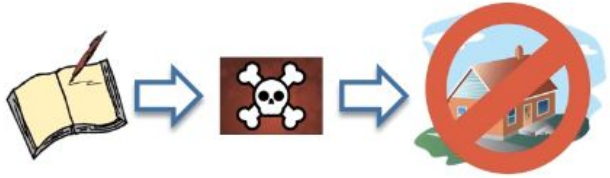
1. One-Way
2. Deterministic
3. Fast Computation
4. The Avalanche Effect
5. Must withstand collisions



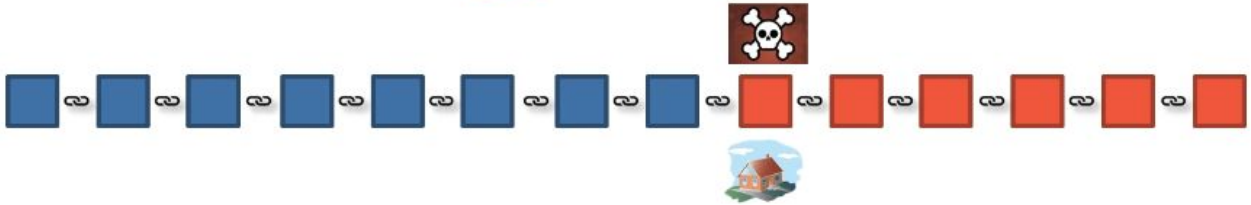
Immutable Ledger



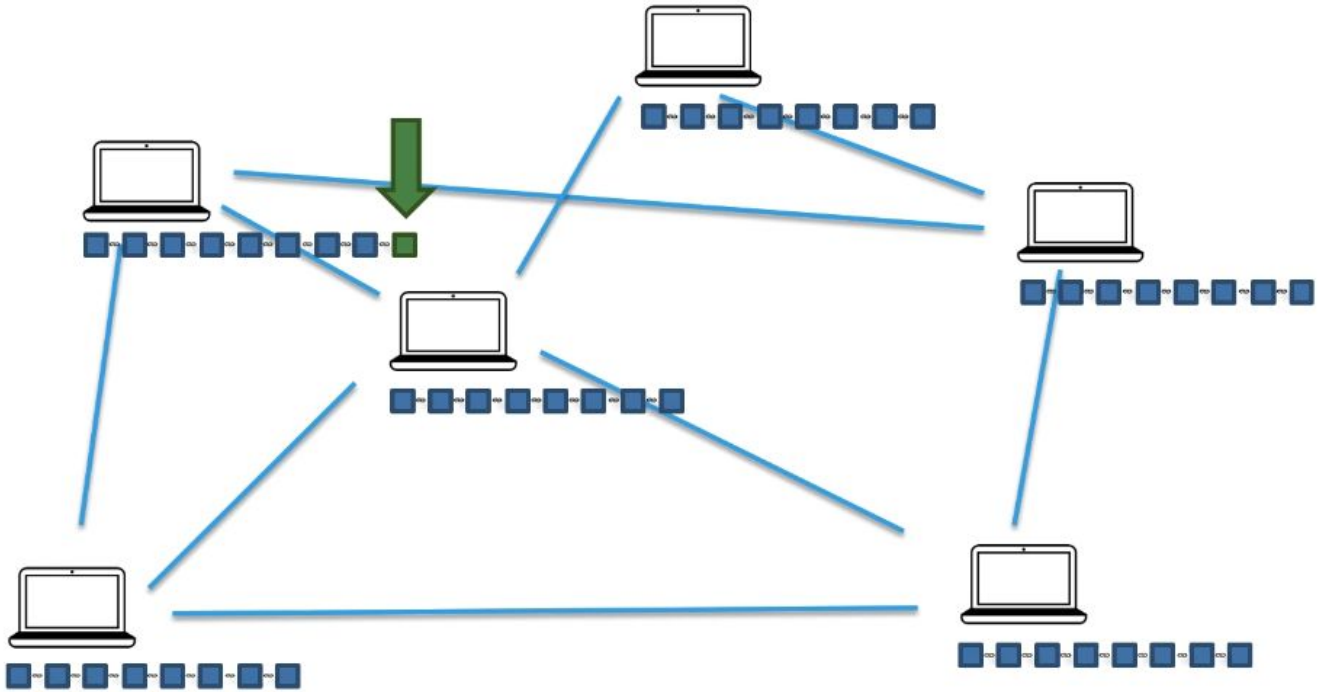
Traditional Ledger



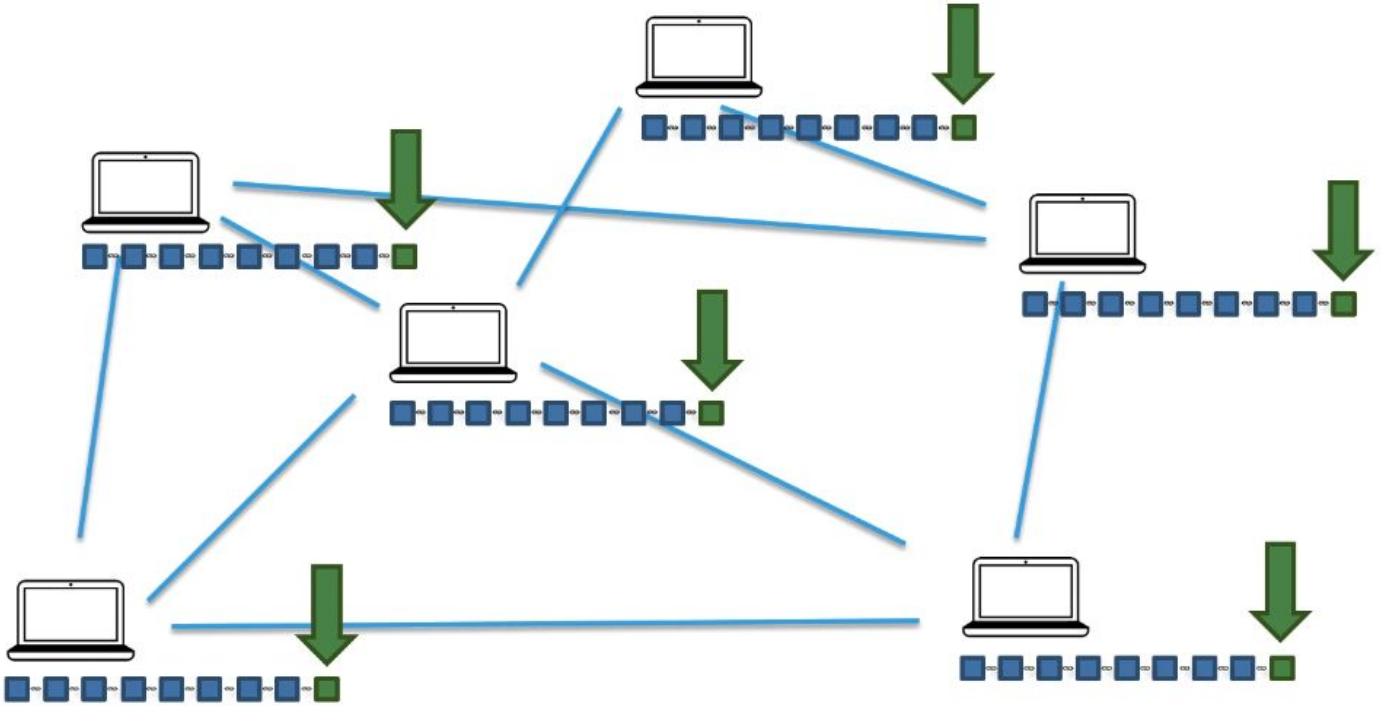
Blockchain



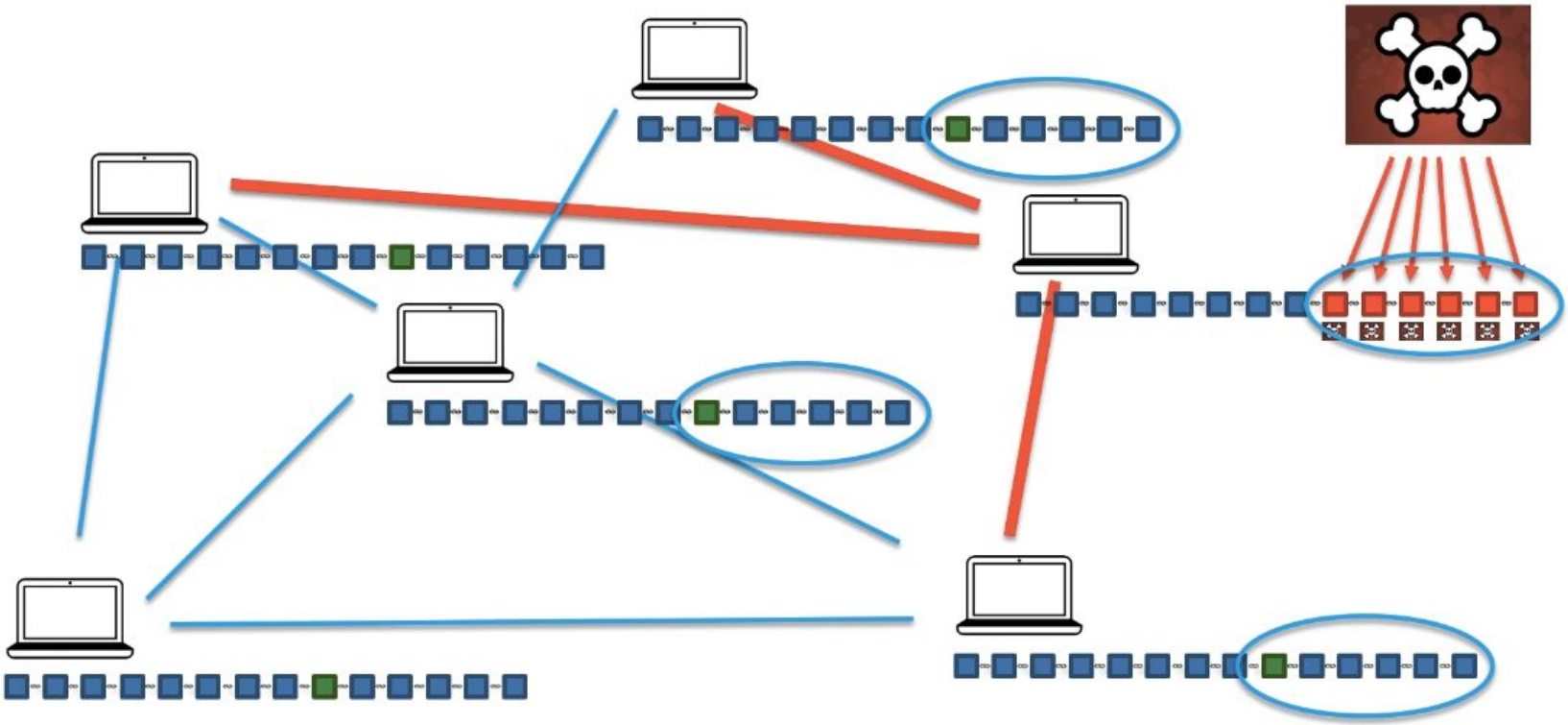
Distributed P2P Network



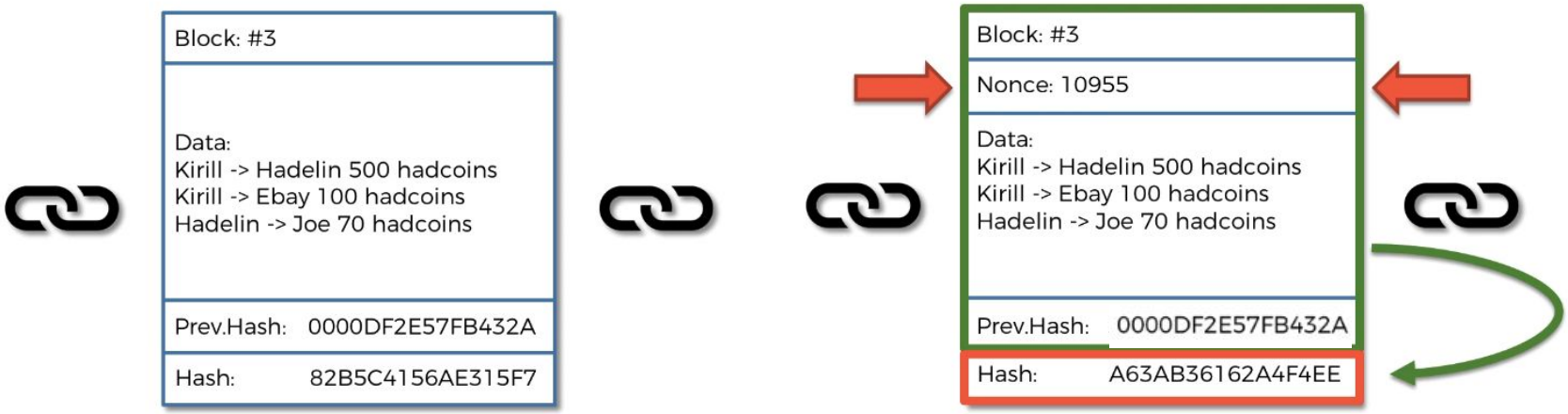
Broadcasting



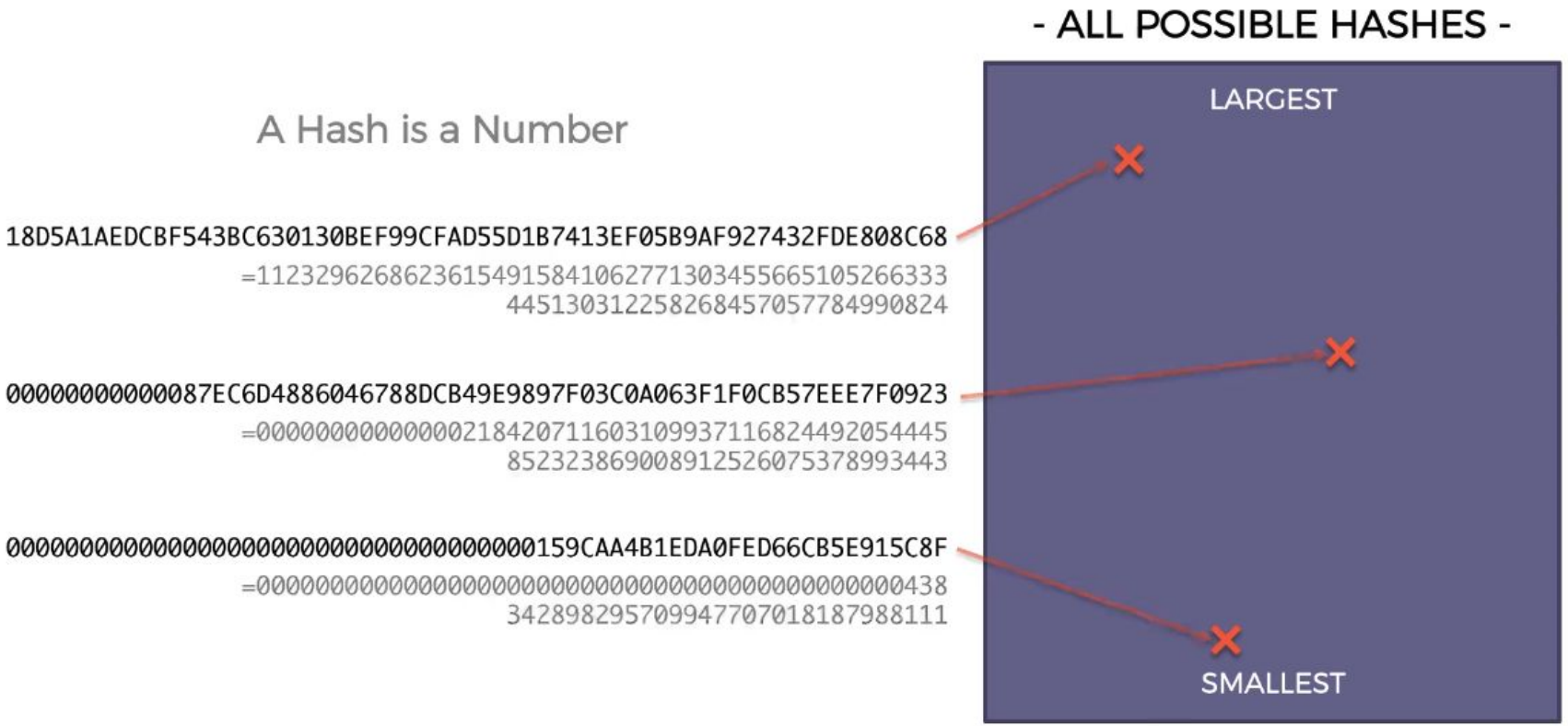
Trust in a trustless environment



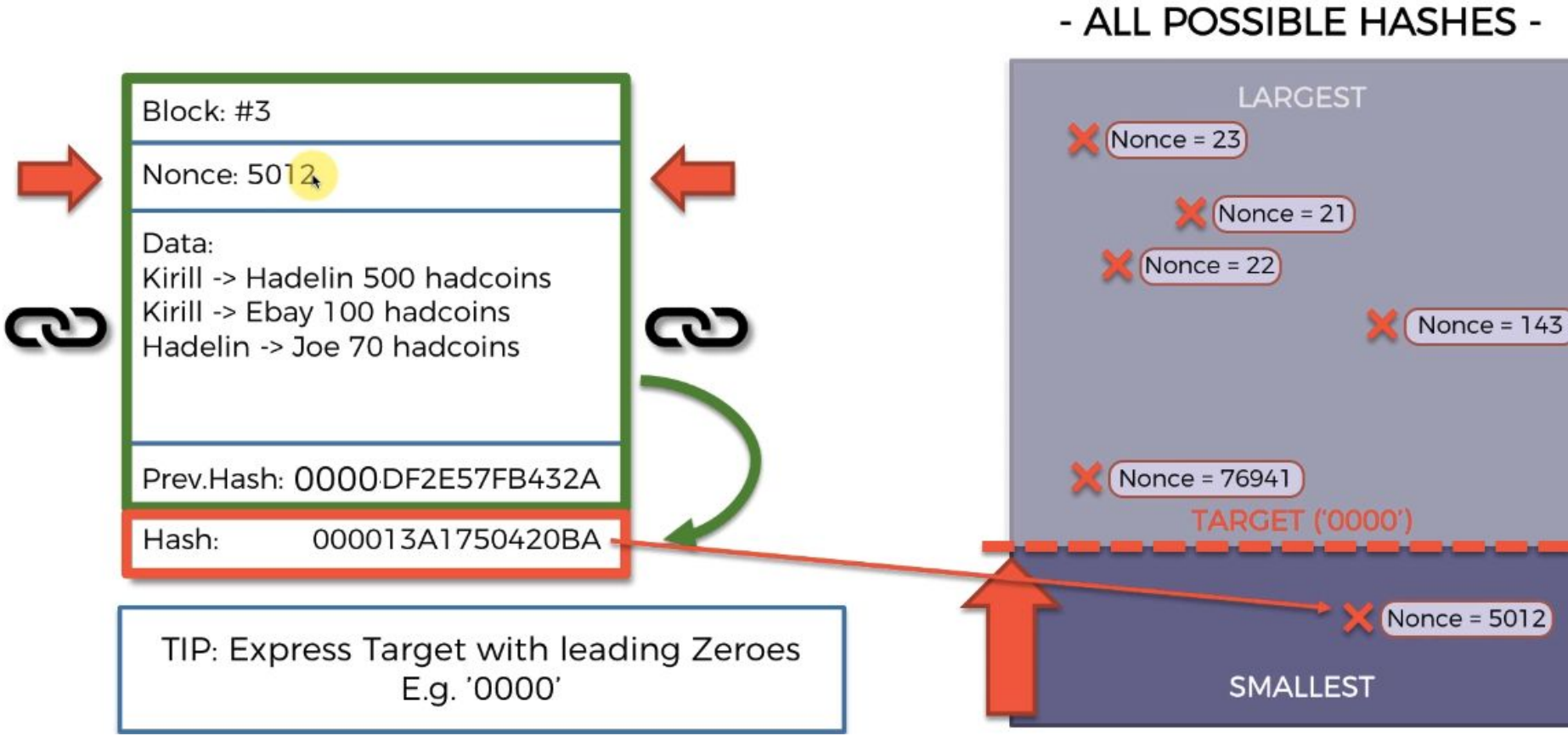
How Mining Works



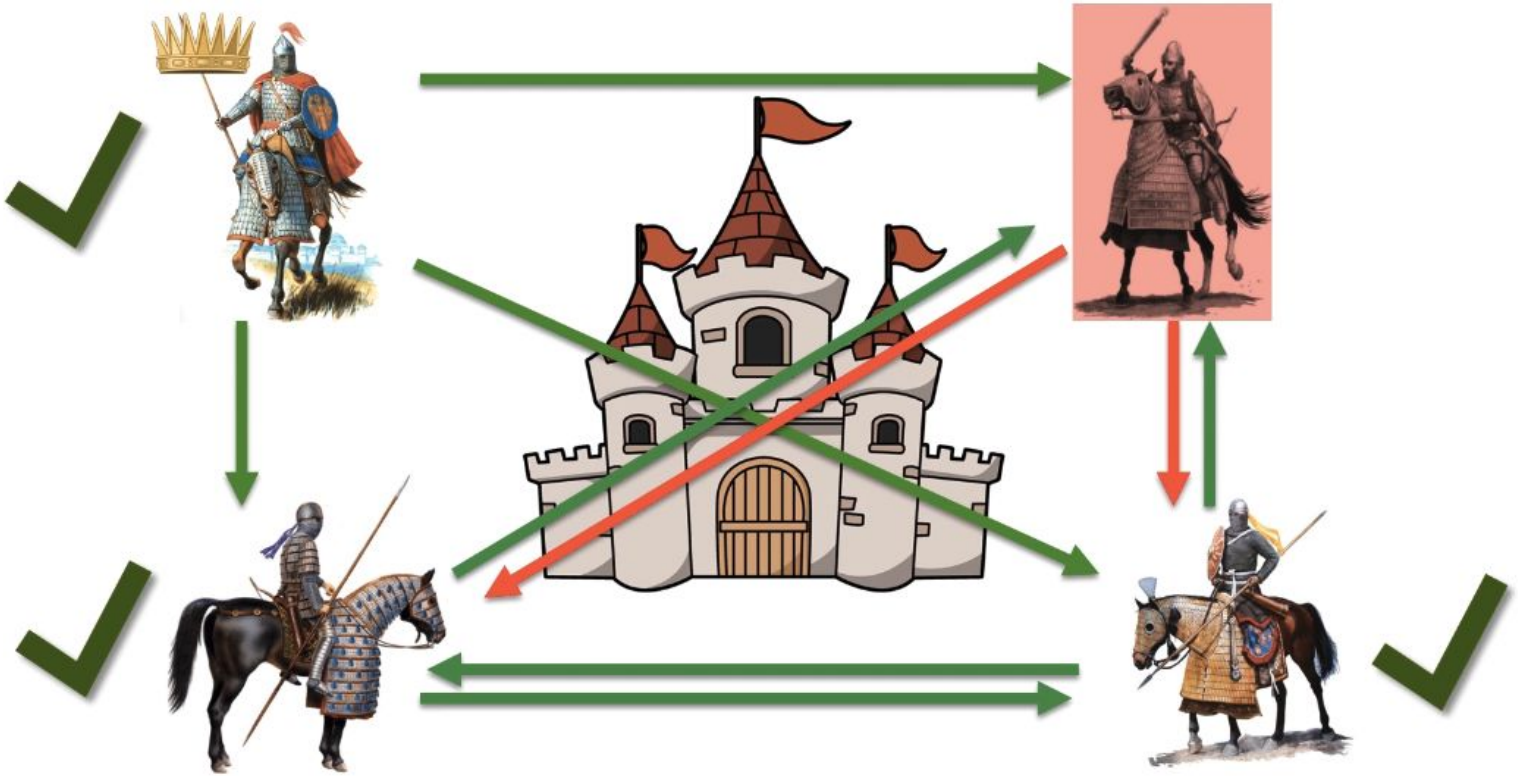
Cryptographic puzzle

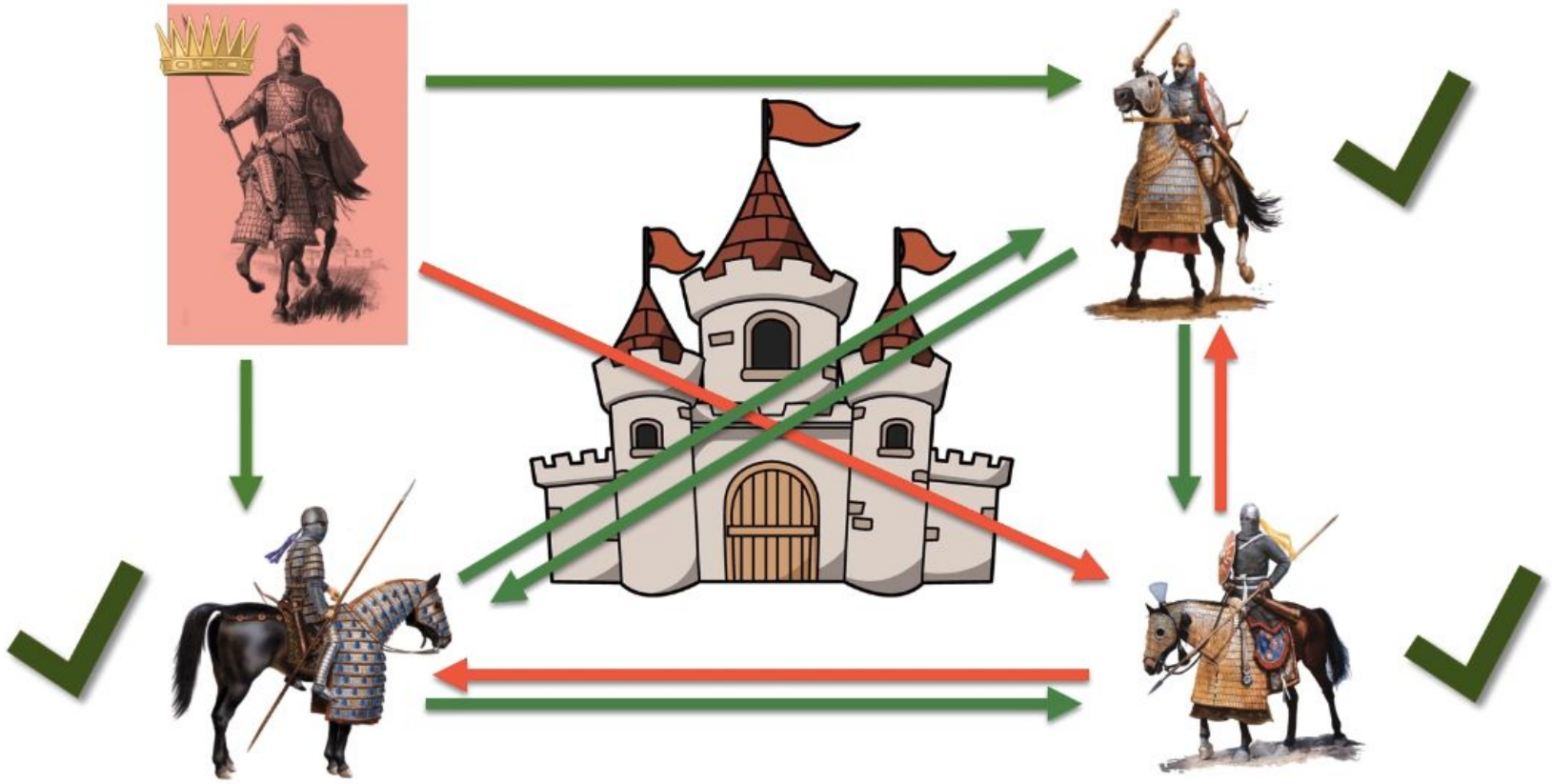


PoW : only exhaustive brute force works

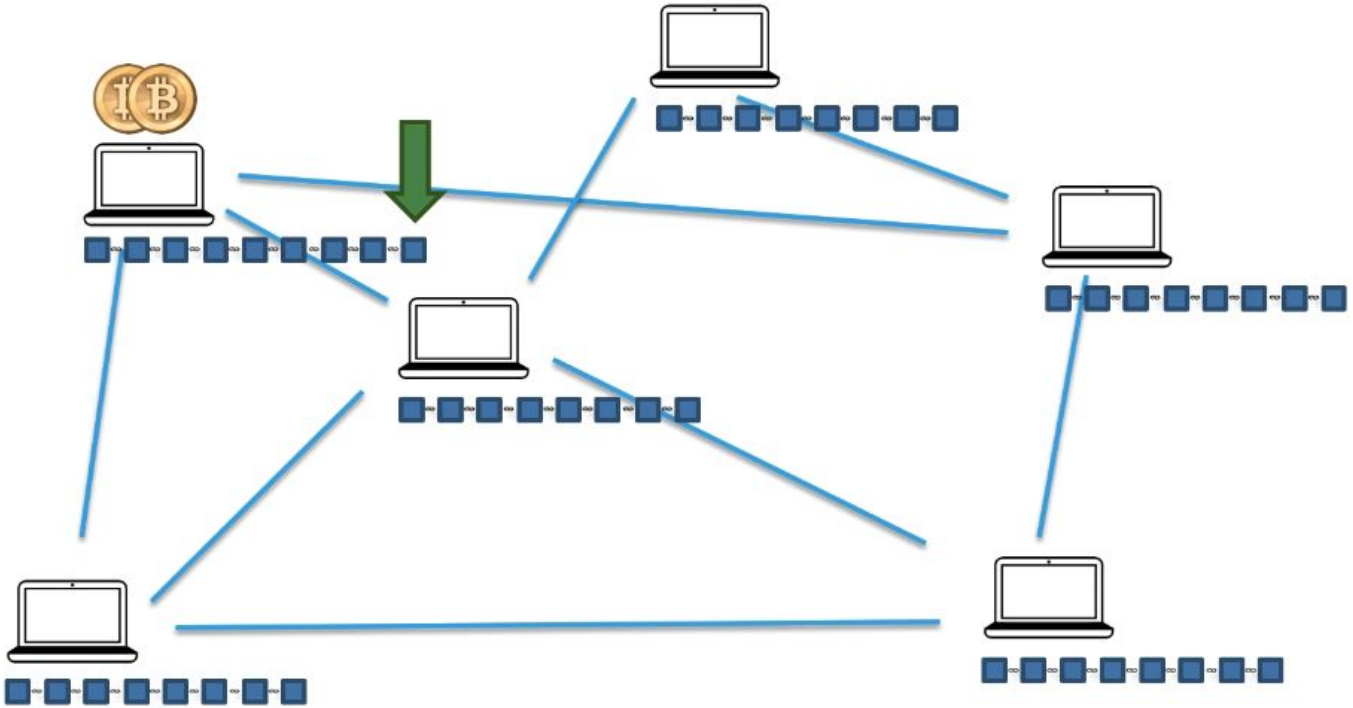


Byzantine Fault Tolerance : achieving consensus with traitors

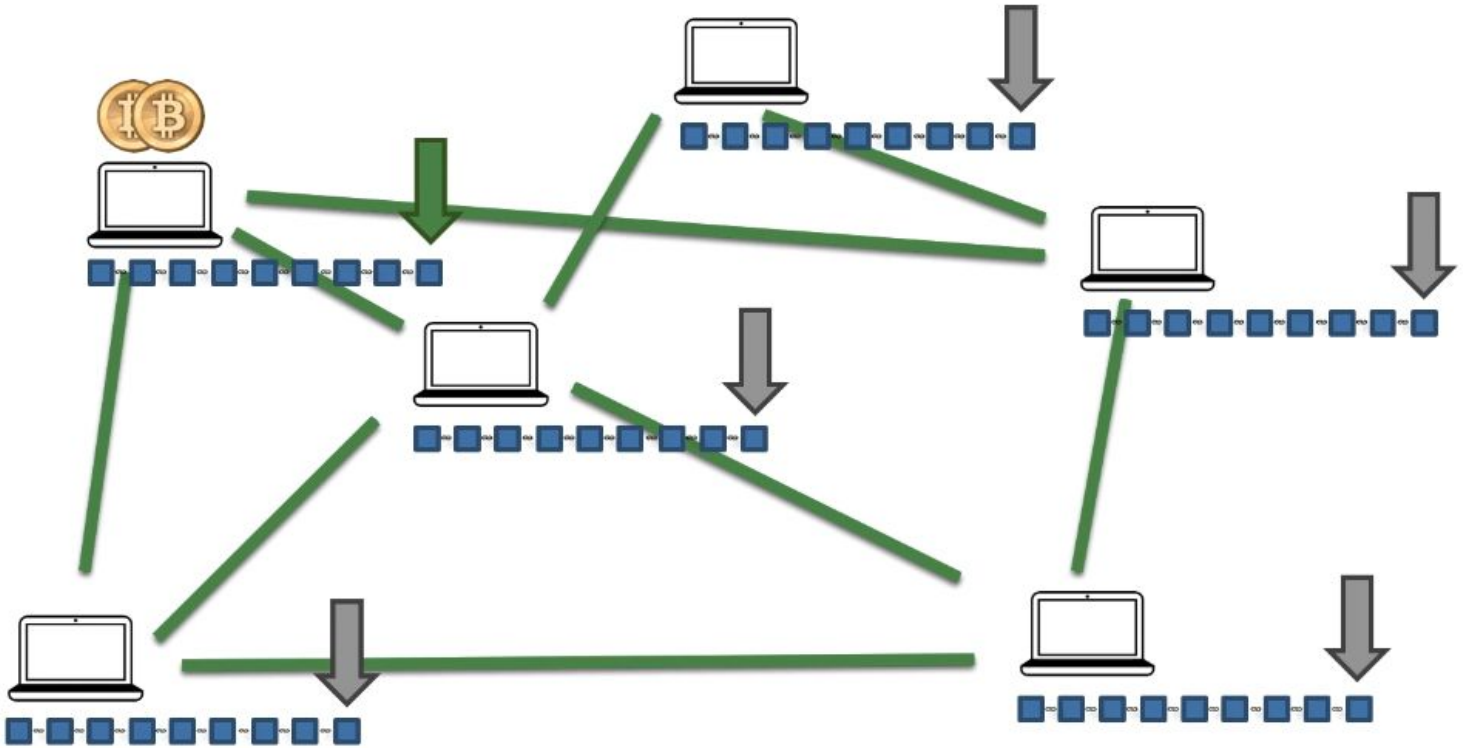




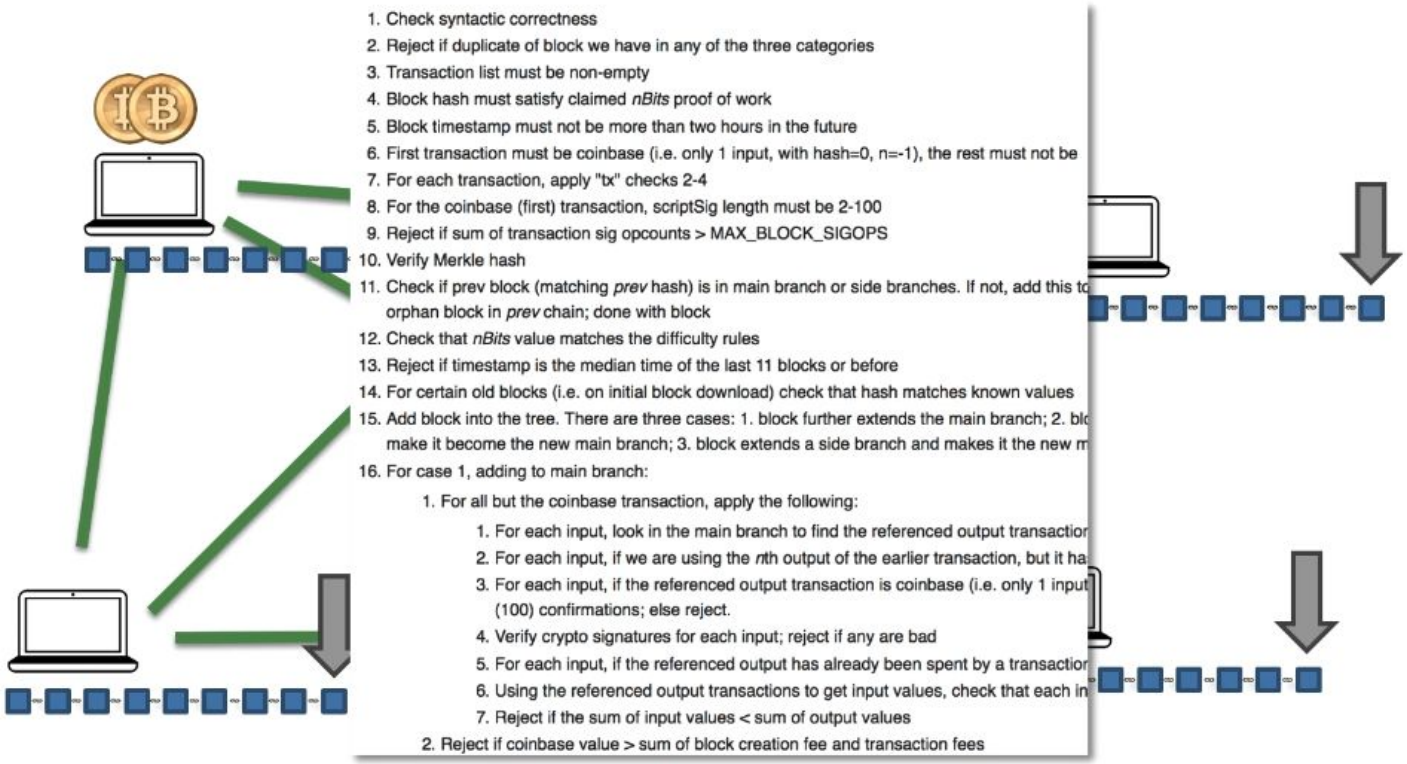
Consensus Protocol



Consensus protocol

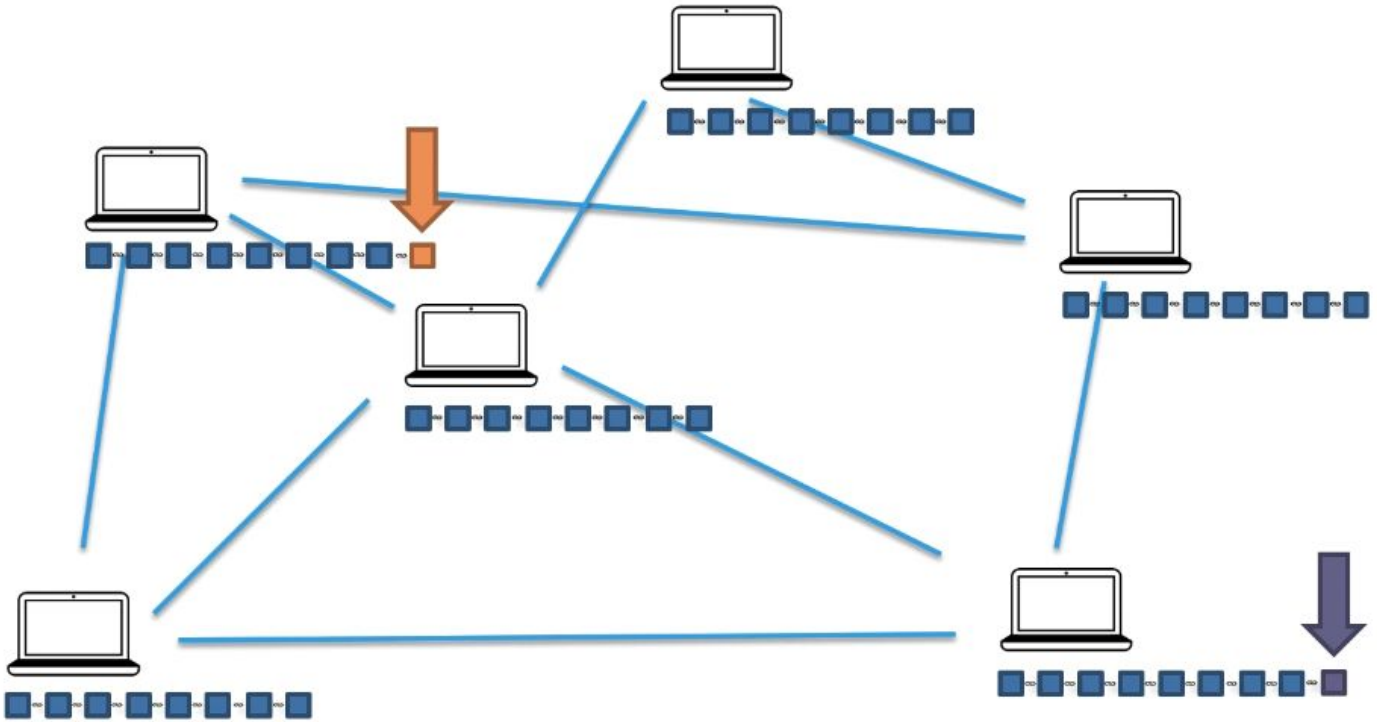


Series of check

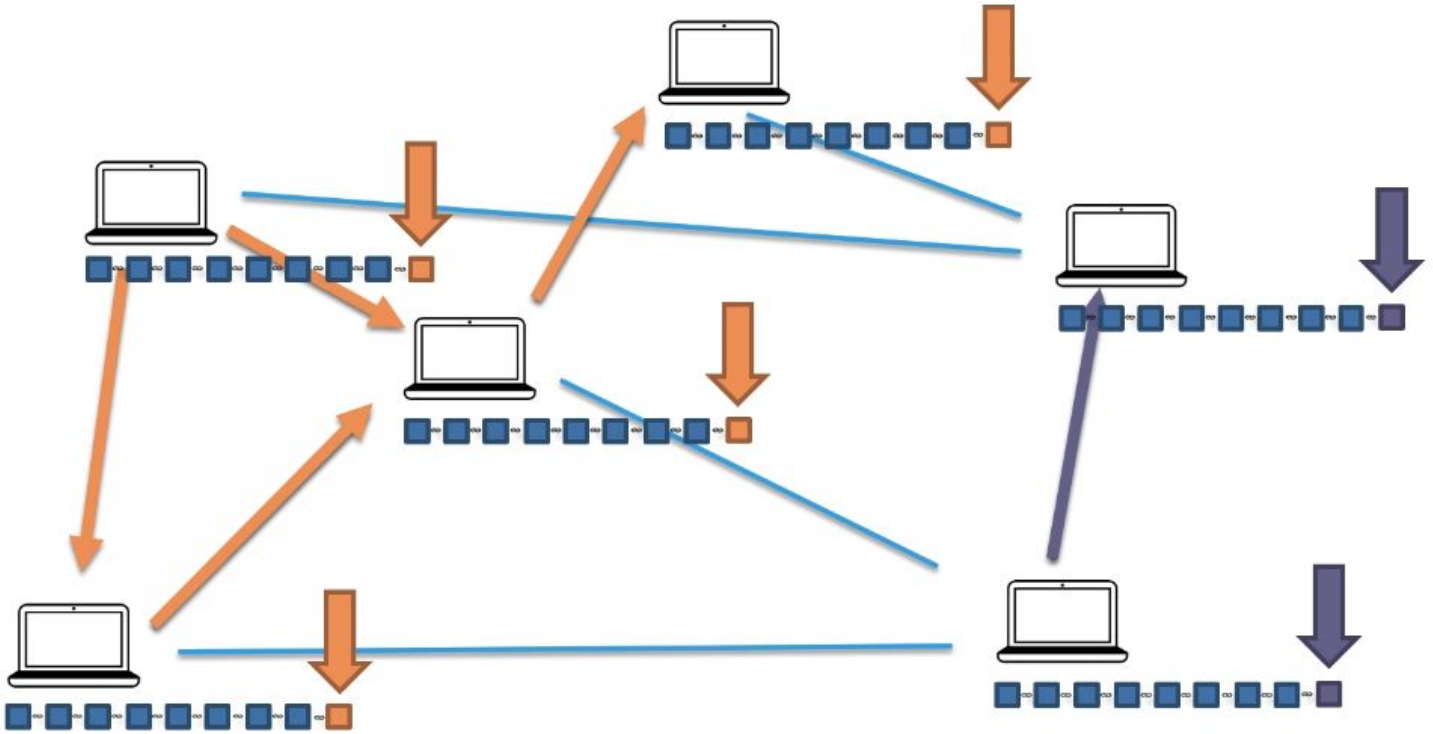


1. Check syntactic correctness
2. Reject if duplicate of block we have in any of the three categories
3. Transaction list must be non-empty
4. Block hash must satisfy claimed *nBits* proof of work
5. Block timestamp must not be more than two hours in the future
6. First transaction must be coinbase (i.e. only 1 input, with hash=0, n=-1), the rest must not be
7. For each transaction, apply "tx" checks 2-4
8. For the coinbase (first) transaction, scriptSig length must be 2-100
9. Reject if sum of transaction sig opcounts > MAX_BLOCK_SIGOPS
10. Verify Merkle hash
11. Check if prev block (matching *prev* hash) is in main branch or side branches. If not, add this to orphan block in *prev* chain; done with block
12. Check that *nBits* value matches the difficulty rules
13. Reject if timestamp is the median time of the last 11 blocks or before
14. For certain old blocks (i.e. on initial block download) check that hash matches known values
15. Add block into the tree. There are three cases: 1. block further extends the main branch; 2. blo make it become the new main branch; 3. block extends a side branch and makes it the new m
16. For case 1, adding to main branch:
    1. For all but the coinbase transaction, apply the following:
        1. For each input, look in the main branch to find the referenced output transactio
        2. For each input, if we are using the *n*th output of the earlier transaction, but it ha
        3. For each input, if the referenced output transaction is coinbase (i.e. only 1 input (100) confirmations; else reject.
        4. Verify crypto signatures for each input; reject if any are bad
        5. For each input, if the referenced output has already been spent by a transactio
        6. Using the referenced output transactions to get input values, check that each in
        7. Reject if the sum of input values < sum of output values
    2. Reject if coinbase value > sum of block creation fee and transaction fees

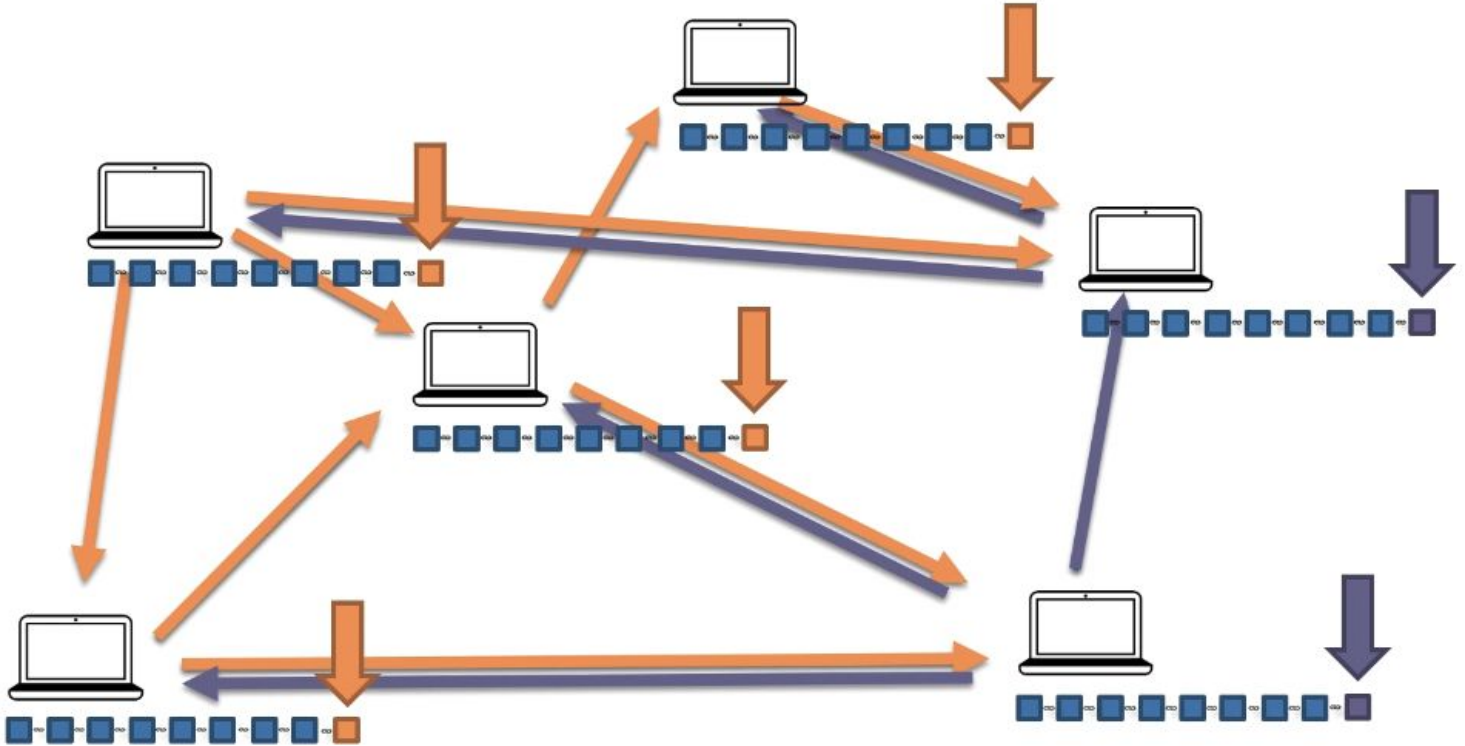
PoW to avoid fraud



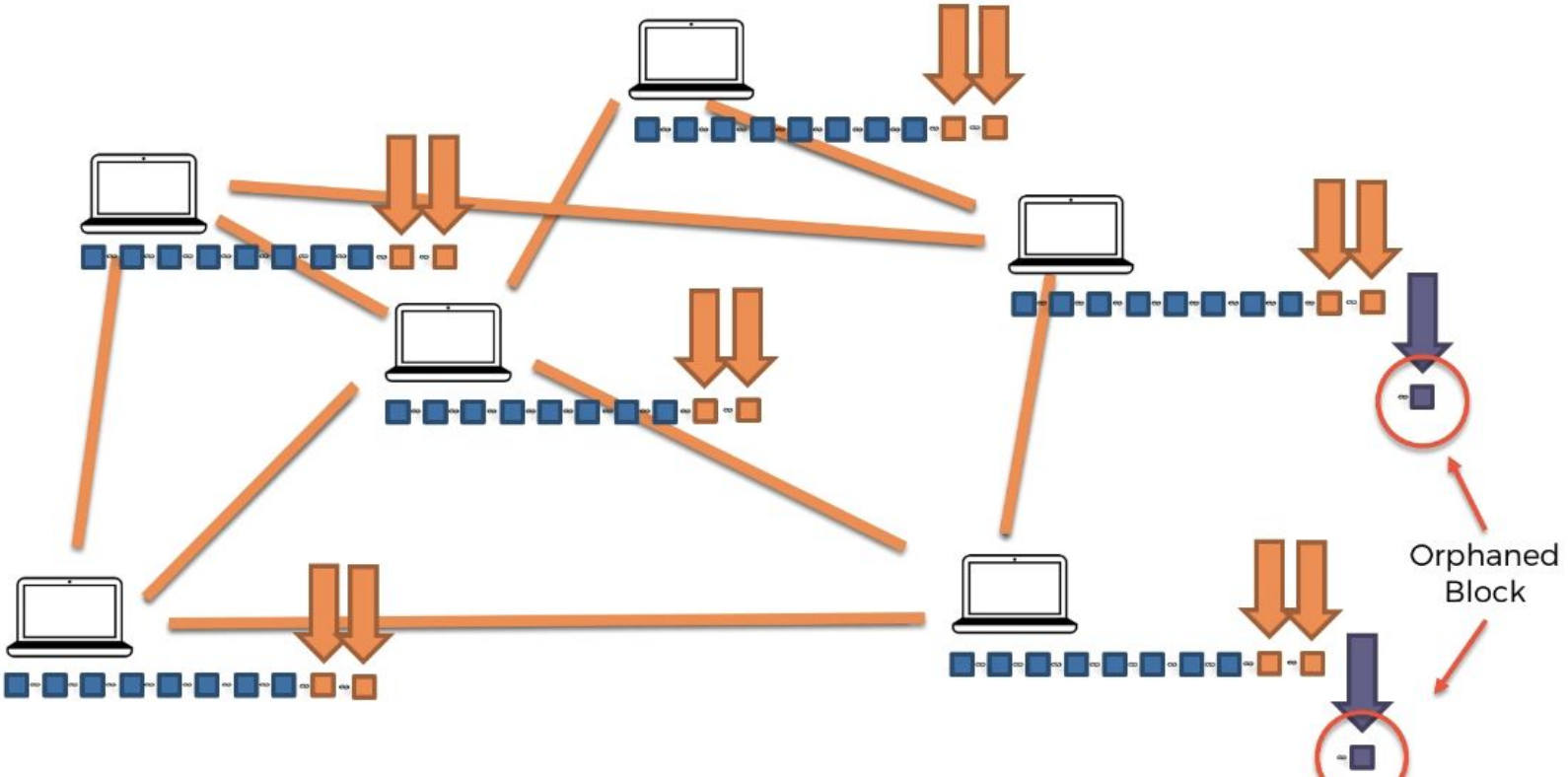
PoW to avoid fraud



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



Hashing power will define the longest chain hence the truth



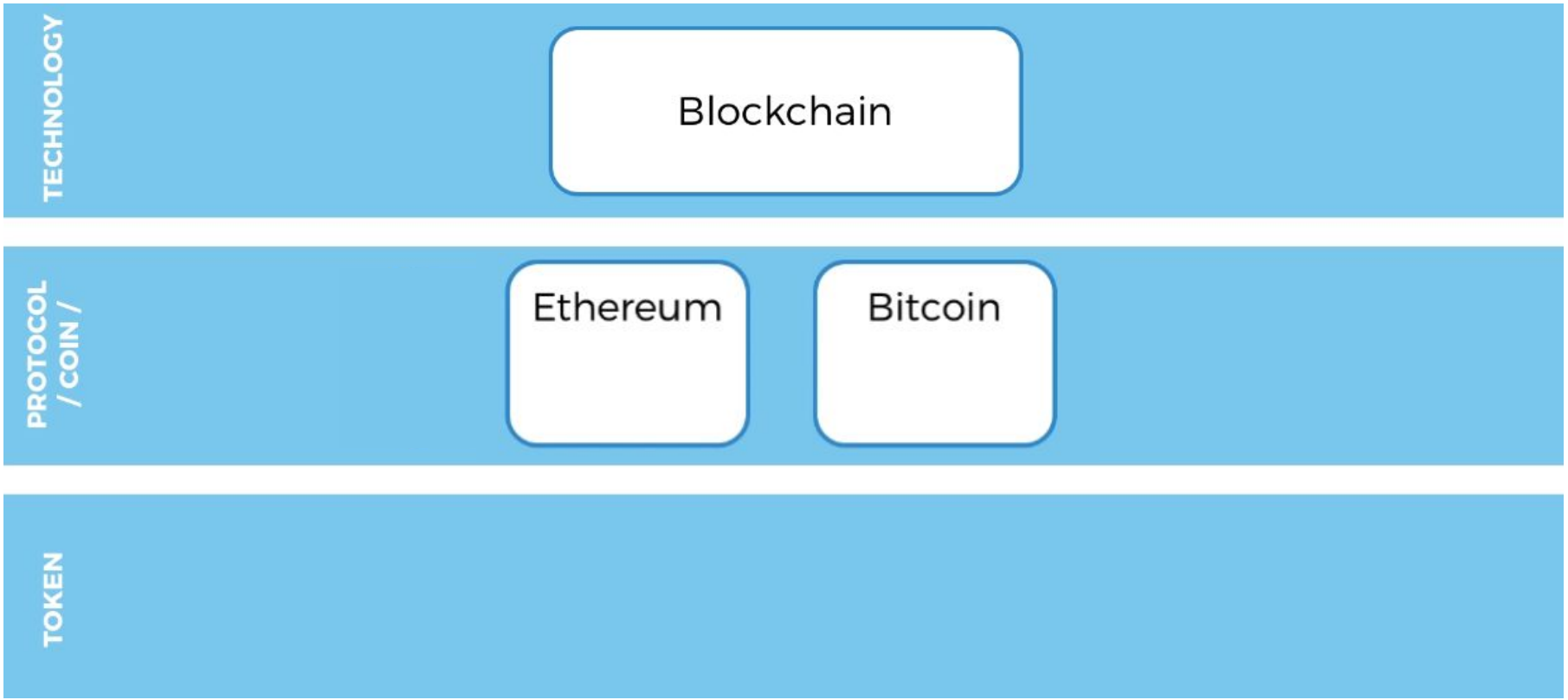
Blockchain demo !

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

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

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

Technological stack



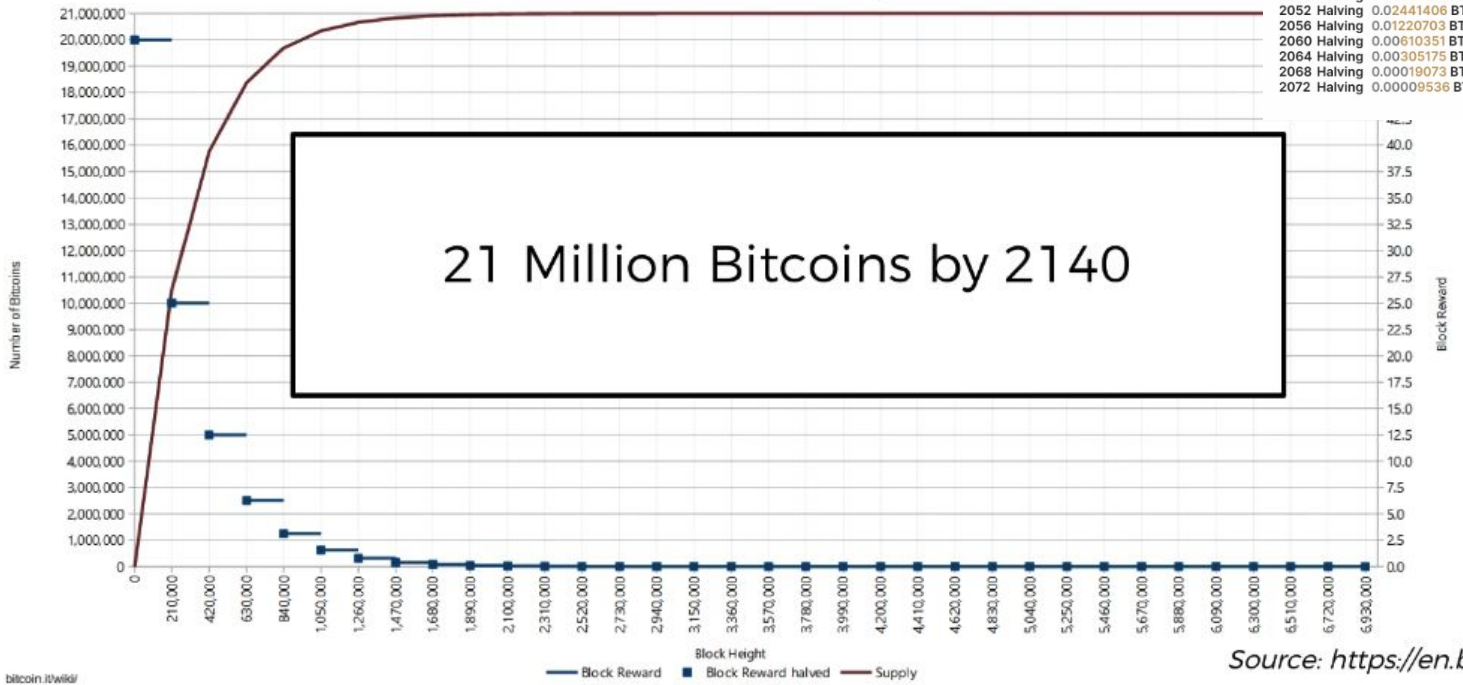
Bitcoin Monetary Policy

The Halving

Bitcoin Halving Schedule
(From Start To 1 Satoshi)

Genesis Block - 2009 - 50 BTC

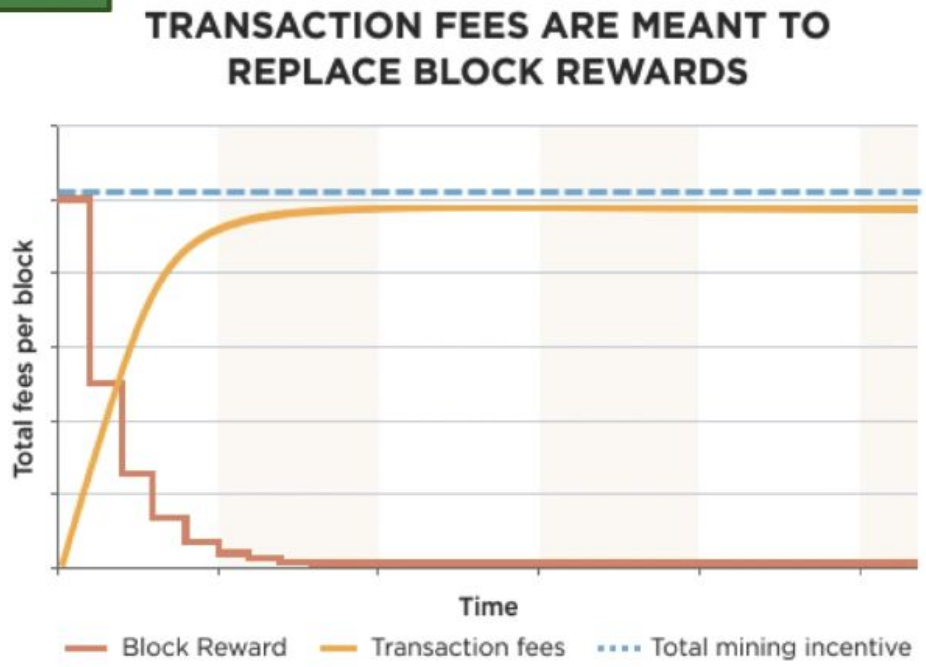
Year		Block Subsidy	Year		Block Subsidy
2012	Halving	25 BTC	2076	Halving	0.00038146 BTC
2016	Halving	12.5 BTC	2080	Halving	0.00019073 BTC
2020	Halving	6.25 BTC	2084	Halving	0.00009536 BTC
2024	Halving	3.125 BTC	2088	Halving	0.00004768 BTC
2028	Halving	1.5625 BTC	2092	Halving	0.00002384 BTC
2032	Halving	0.78125 BTC	2096	Halving	0.00001192 BTC
2036	Halving	0.390625 BTC	2100	Halving	0.00000596 BTC
2040	Halving	0.1953125 BTC	2104	Halving	0.00000298 BTC
2044	Halving	0.09765625 BTC	2108	Halving	0.00000149 BTC
2048	Halving	0.04882812 BTC	2112	Halving	0.00000074 BTC
2052	Halving	0.02441406 BTC	2116	Halving	0.00000037 BTC
2056	Halving	0.01220703 BTC	2120	Halving	0.00000018 BTC
2060	Halving	0.00610351 BTC	2124	Halving	0.00000009 BTC
2064	Halving	0.00305175 BTC	2128	Halving	0.00000004 BTC
2068	Halving	0.00019073 BTC	2132	Halving	0.00000002 BTC
2072	Halving	0.00009536 BTC	2136	Halving	0.00000001 BTC



Source: https://en.bitcoin.it/wiki/

Bitcoin Monetary Policy

The Halving



Source: https://bitsonblocks.net

Understanding mining difficulty

Current target = 000000000000000005d97dc0000000000000000000000000000000000000000

18 zeros

Let's do some estimations:

Probability:

Total possible 64-digit hexadecimal numbers: $16 \times 16 \times \ldots \times 16 = 16^{64} \approx 1.1579 \times 10^{77} \approx 10^{77}$

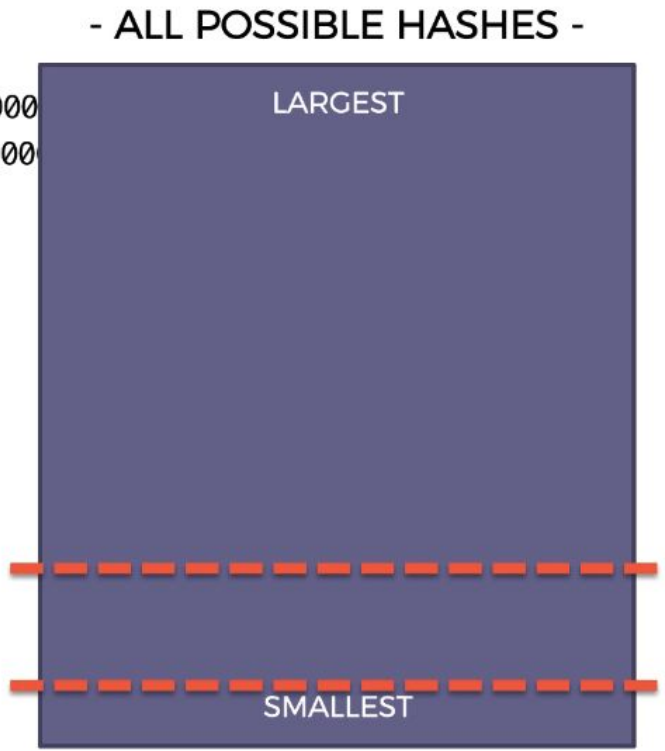
Total valid hashes (with 18 leading zeros): $16 \times 16 \times \ldots \times 16 = 16^{64-18} \approx 2.4519 \times 10^{55} \approx 2 \times 10^{55}$

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

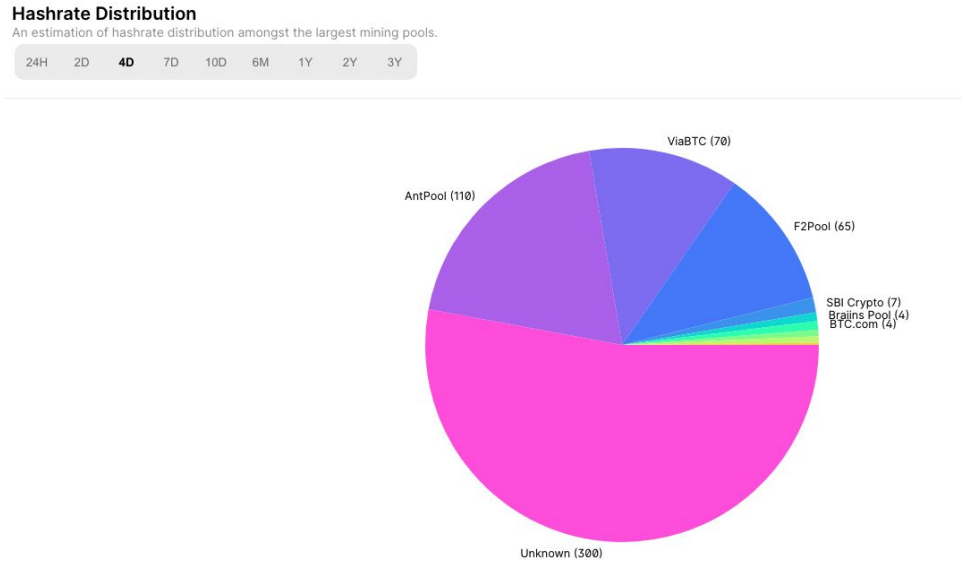
Difficulty = current target / max target
Curr target = `00000000000000000005d97dc00000000000000000000`
Max target = `00000000FFFF0000000000000000000000000000000000`

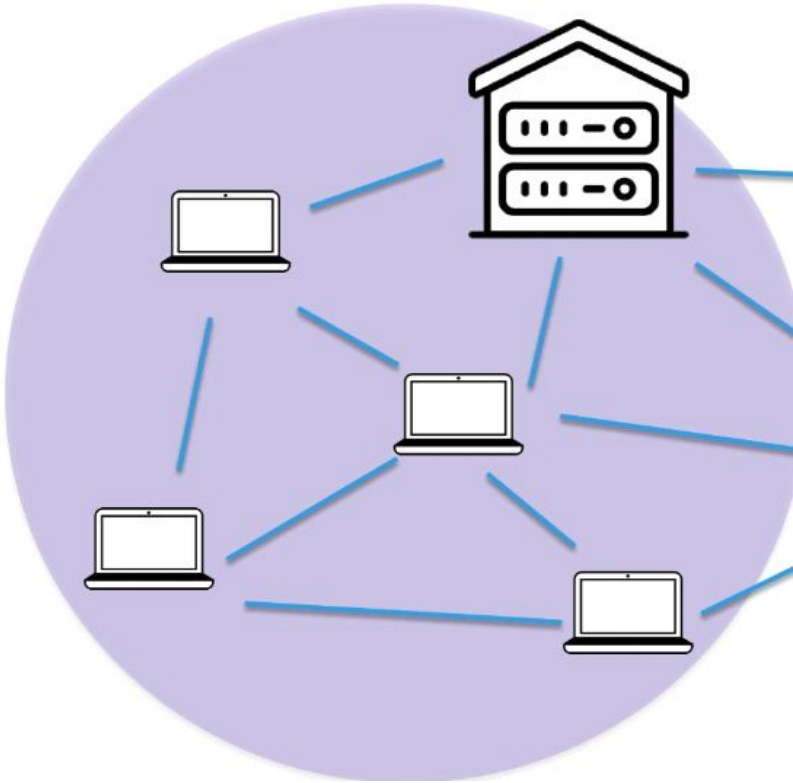
Difficulty is adjusted every 2016 blocks (2 weeks)



Mining pool

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





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

Let's do some estimations:

Difficulty:
Total possible 64-digit hexadecimal numbers: 16 x 16 x …. x 16 = $16^{64} \approx 10^{77}$
Total valid hashes (with 18 leading zeros): 16 x 16 x … x 16 = $16^{64-18} \approx 2 \times 10^{55}$
Probability that a Randomly picked hash is valid: $2 \times 10^{55} / 10^{77} = 2 \times 10^{-22}$ = 0.00000000000000000002%

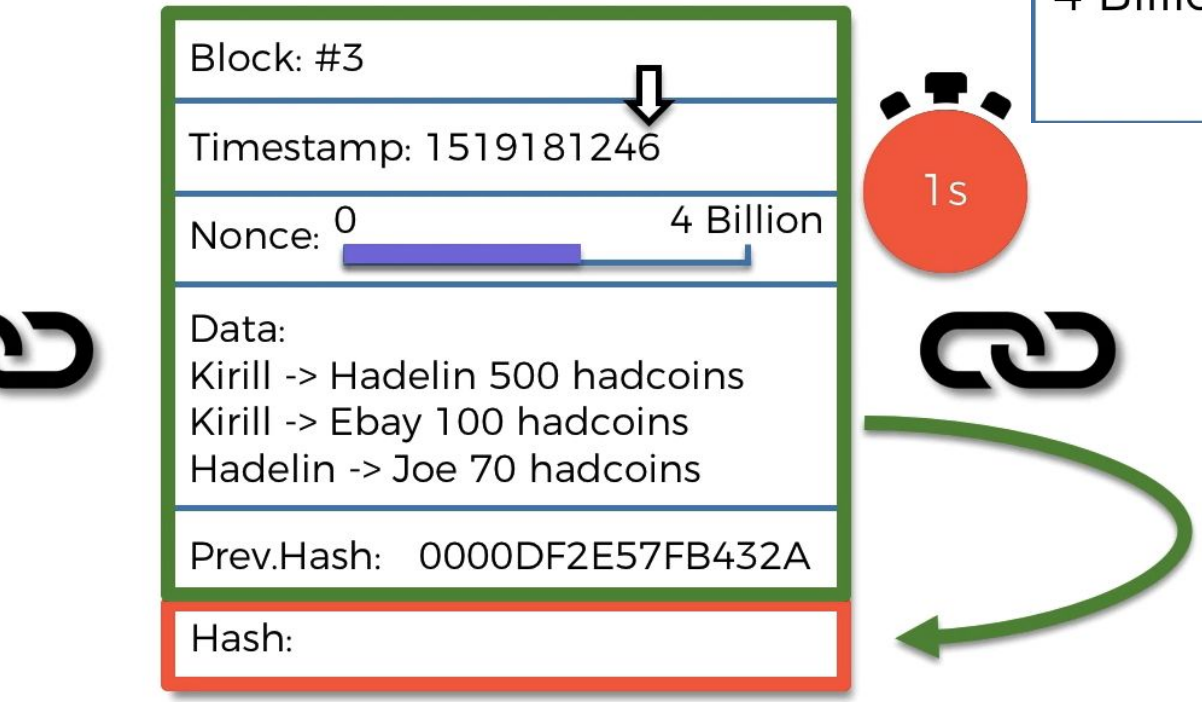
Nonce:
The Nonce is a 32-bit number, the Max Nonce = 2^{32} = 4,294,967,296 = 4×10^{9}
Assuming no collisions, this means 4×10^{9} different hashes
Probability that ONE of them will be valid: $4 \times 10^{9} \times 2 \times 10^{-22} = 8 \times 10^{-13} \approx 10^{-12}$ = 0.0000000001%

Conclusion: One Nonce Range is not enough

32 bits Nonce : around 4 billions trials

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

4 Billion / 100 Million = 40 seconds



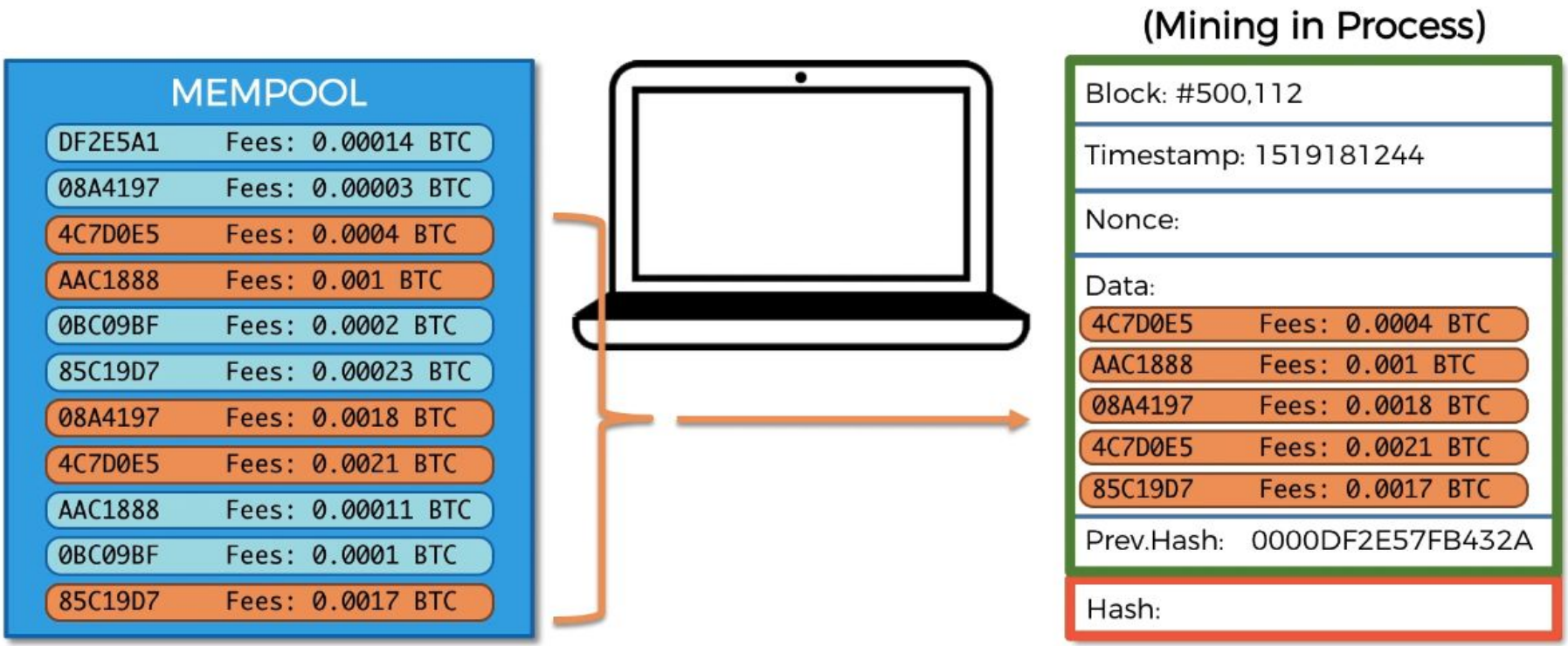
- good for a single miner
- What for a mining pool ?

Blockchain.com explorer

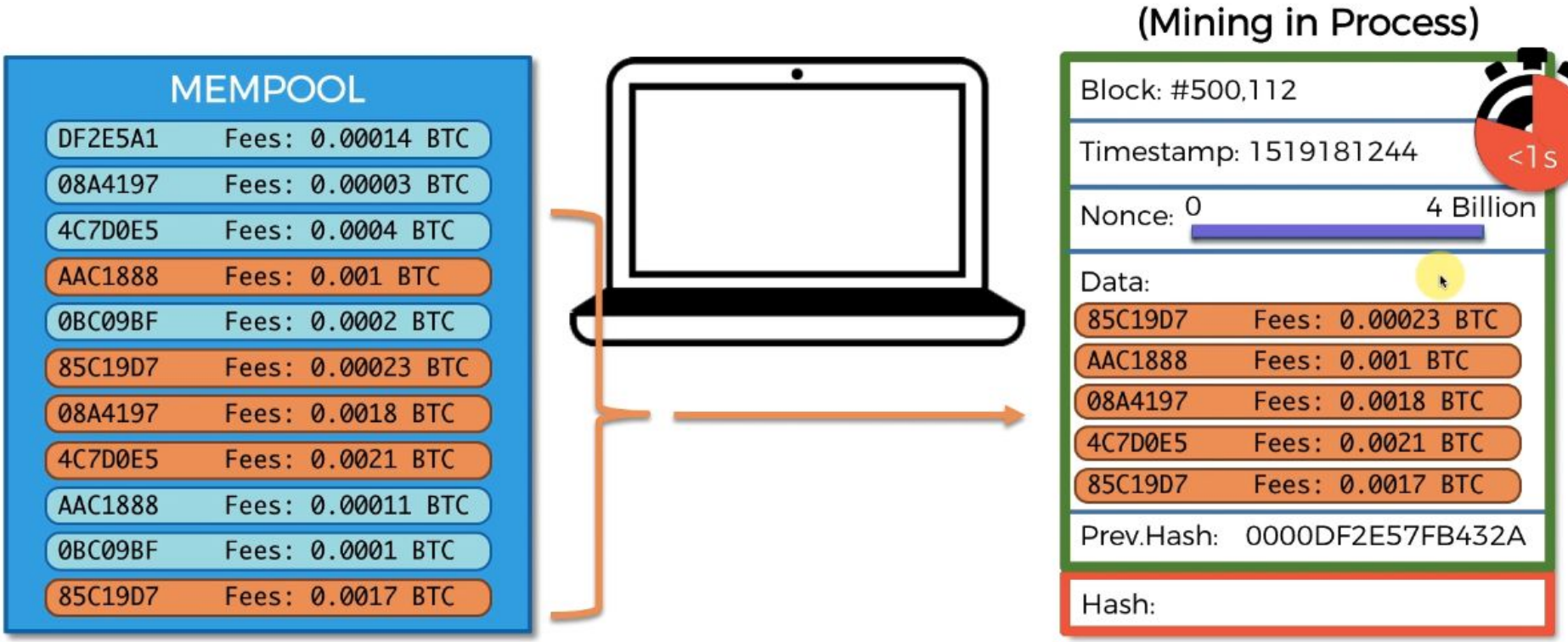
https://www.blockchain.com/explorer/charts/hash-rate

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

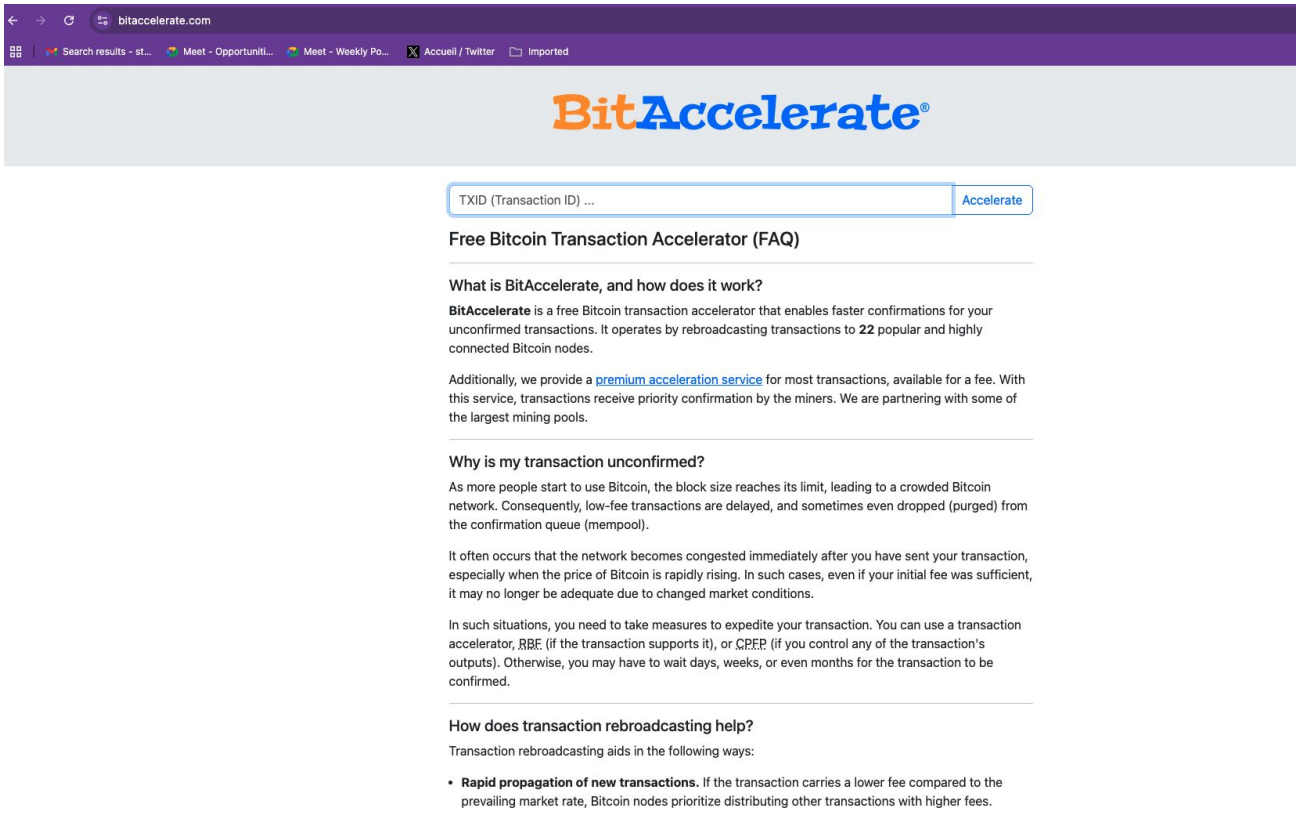
Picking transactions



Reshuffling transactions to use most of hashing power

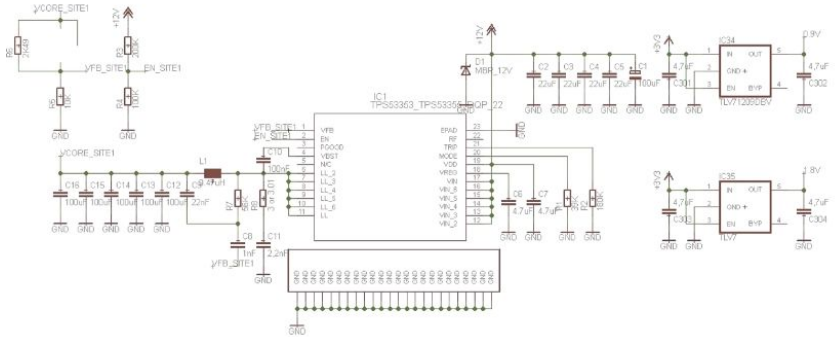


Accelerate your transaction

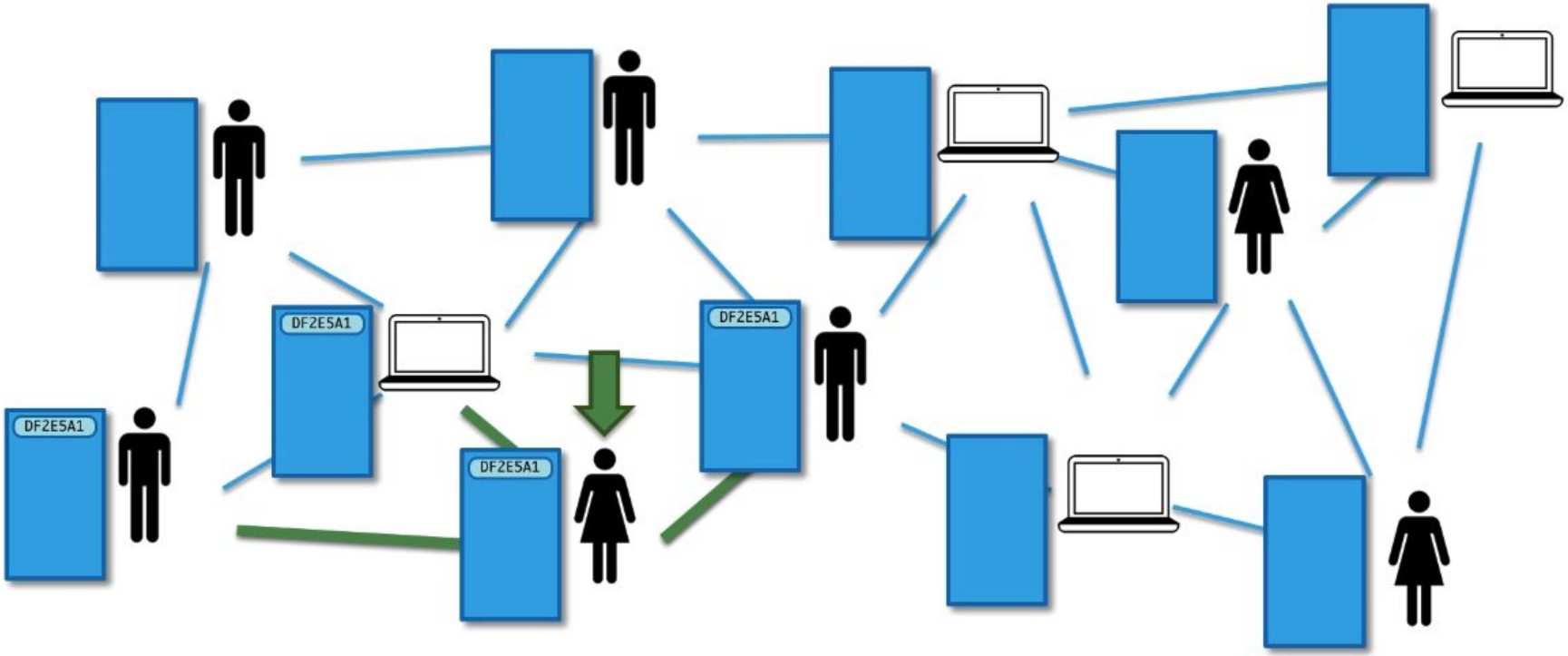


CPU versus GPU versus ASICS

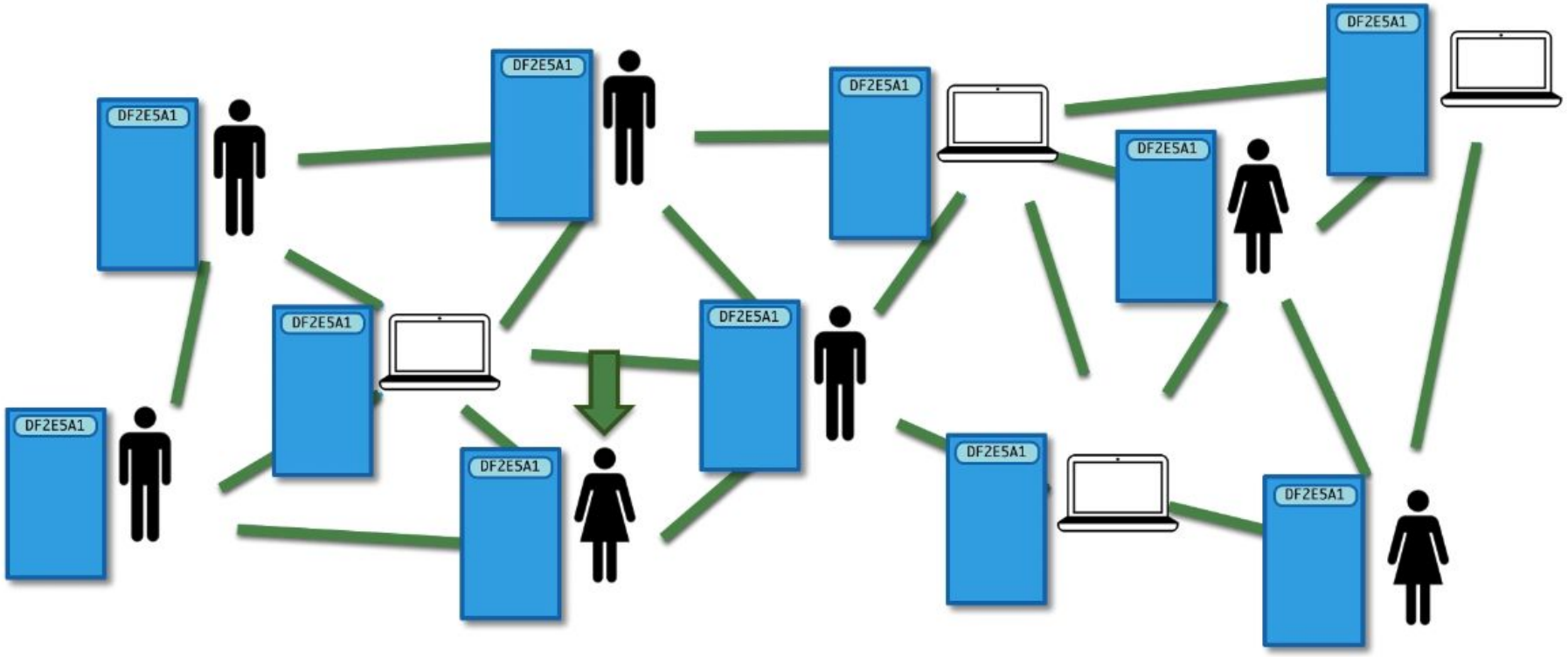
CPU = Central Processing Unit	General	< 10 MH/s
GPU = Graphics Processing Unit	Specialized	< 1 GH/s
ASIC = Application-Specific Integrated Circuit	Totally Specialized	> 1,000 GH/s

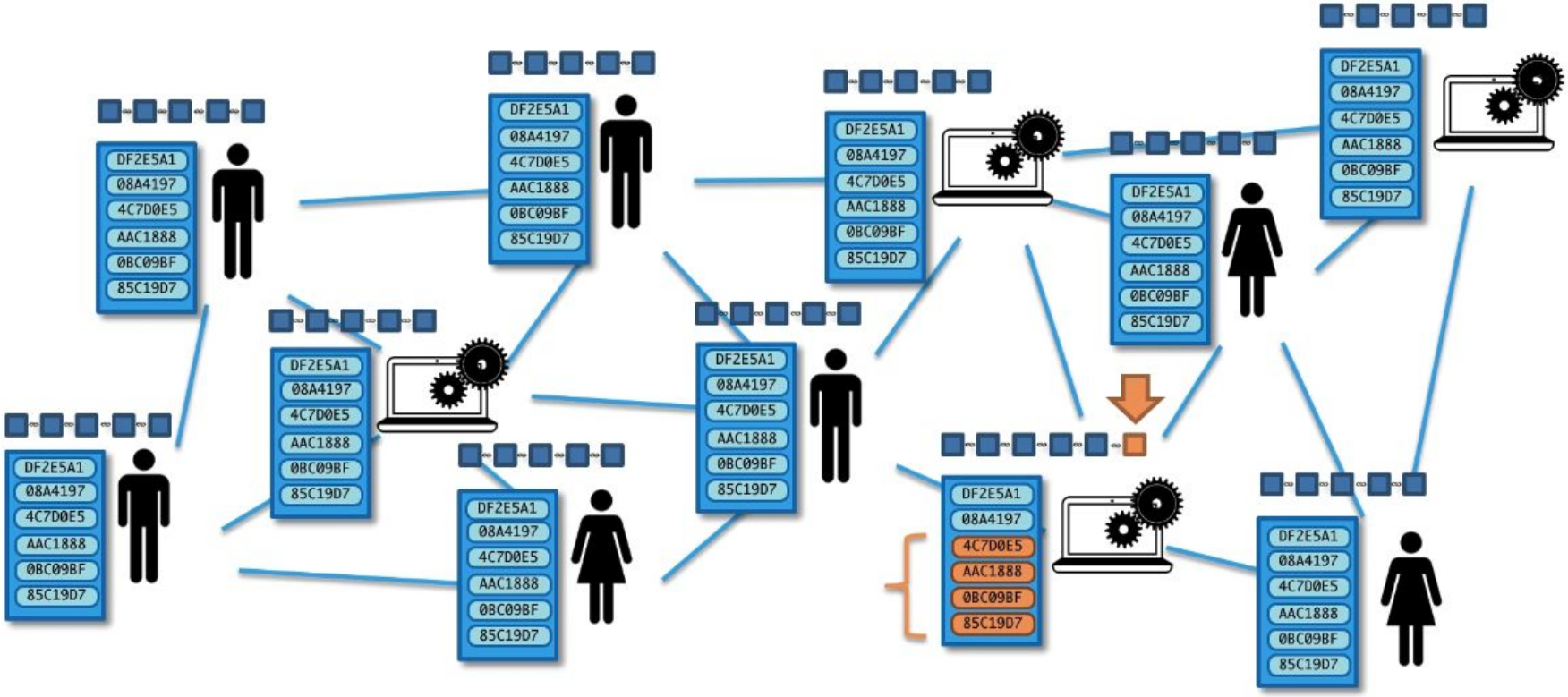


How do MemPools work ?



How do MemPools work ?

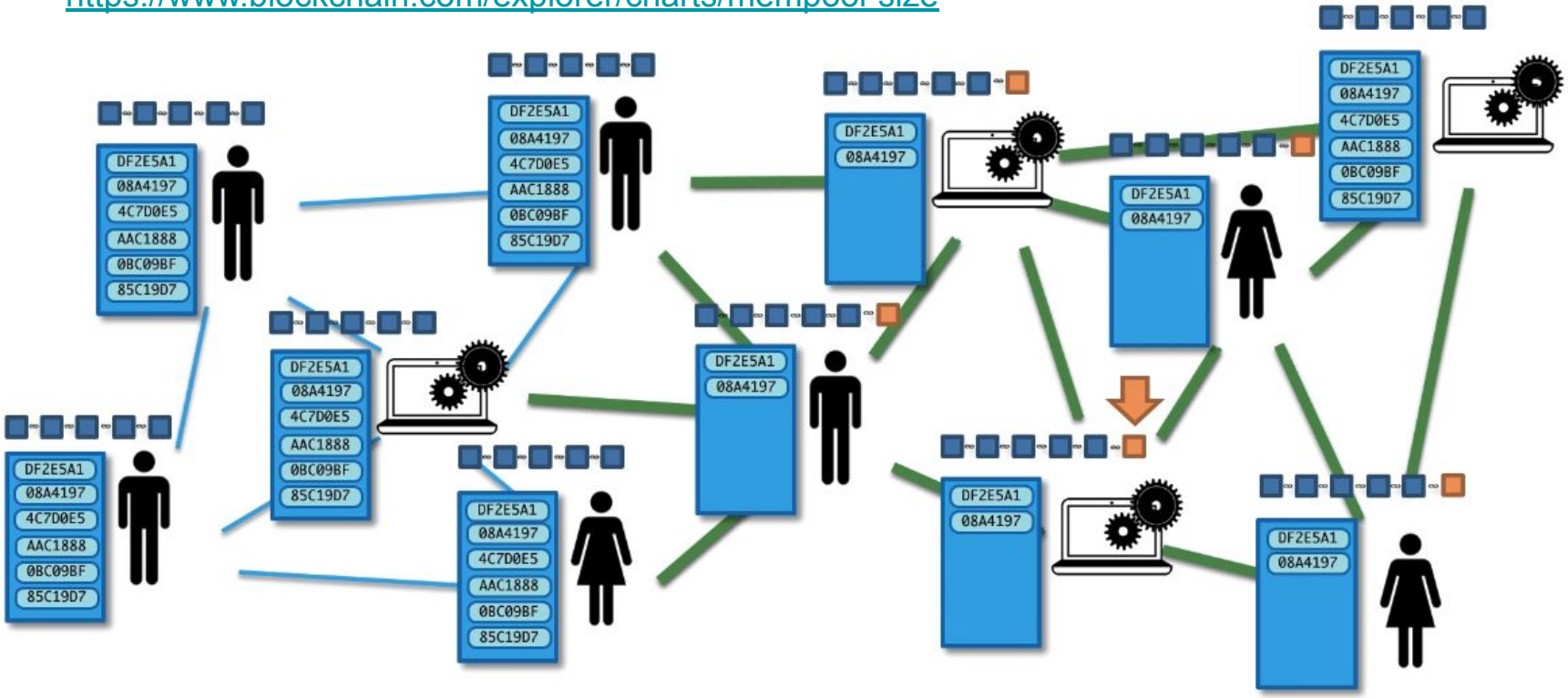



DF2E5A1
08A4197
4C7D0E5
AAC1888
0BC09BF
85C19D7

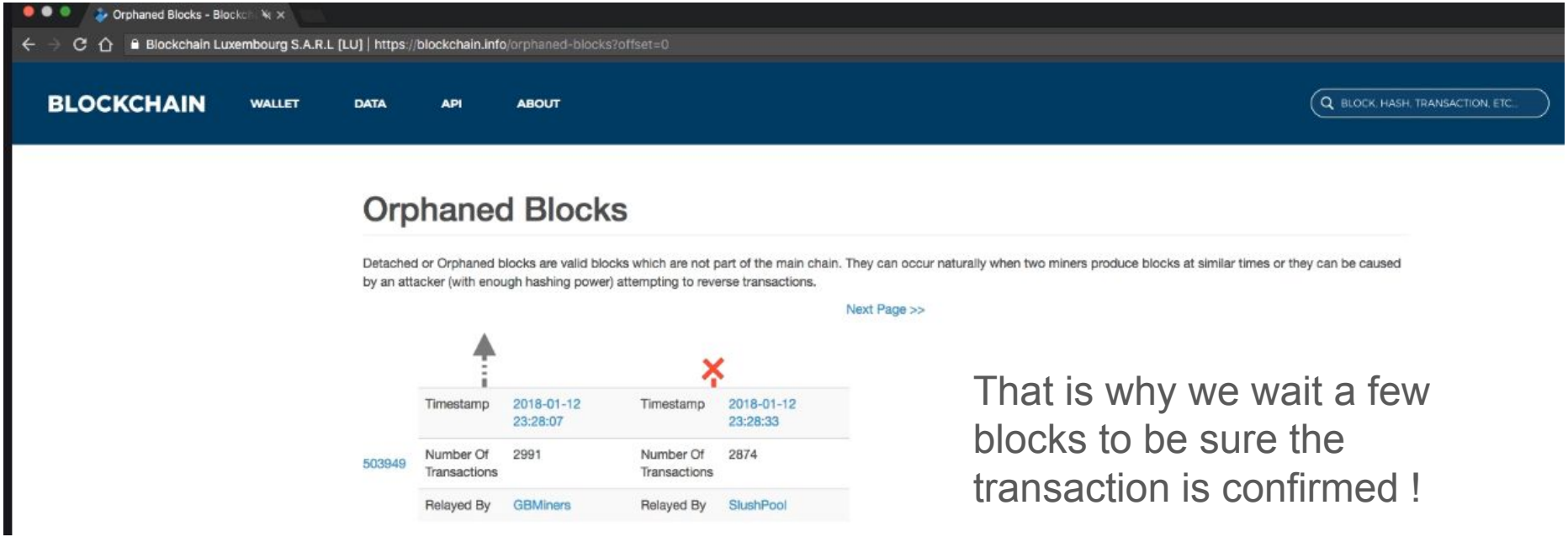
https://blog.kaiko.com/an-in-depth-guide-into-how-the-mempool-works-c758b781c608

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

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

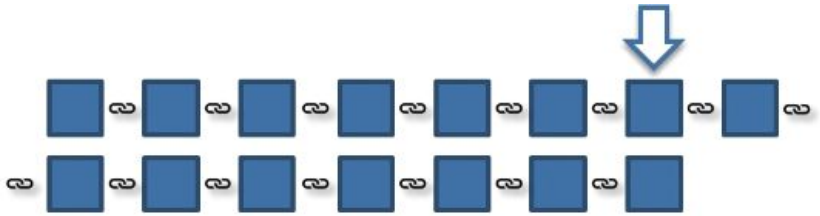


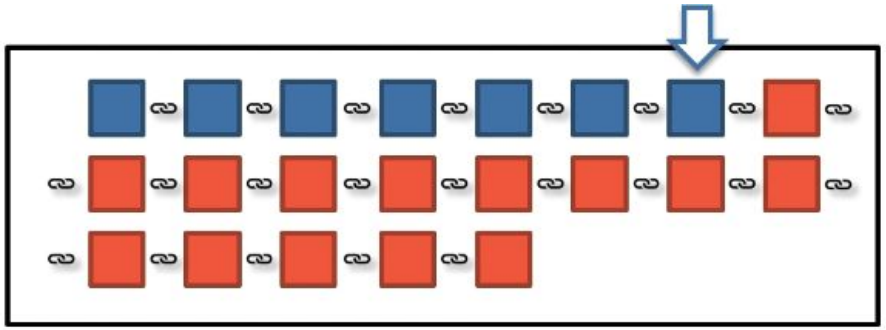
Orphaned block: part of the experience
transactions are rereleased into the mempool

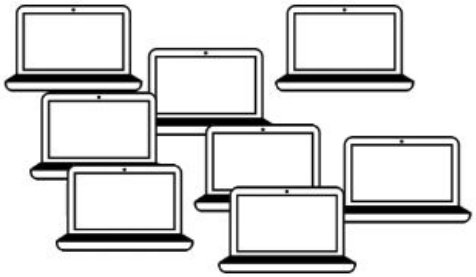


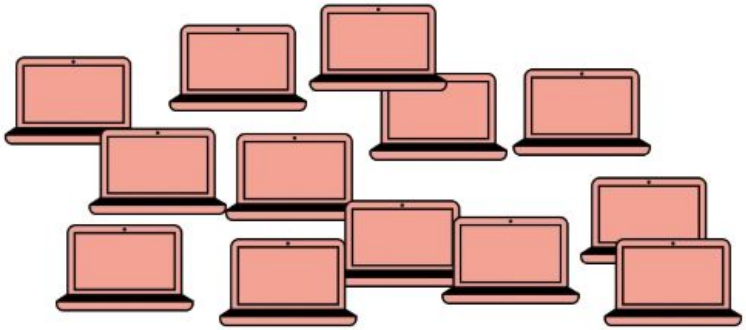
That is why we wait a few blocks to be sure the transaction is confirmed !

The 51% attack

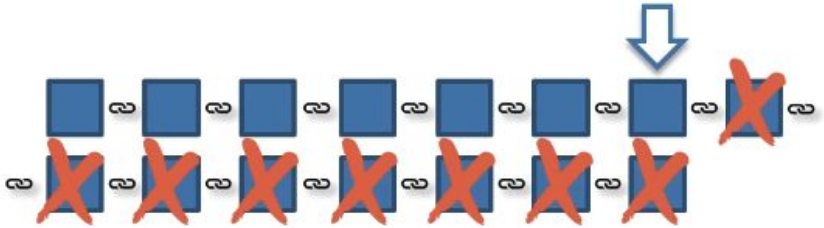




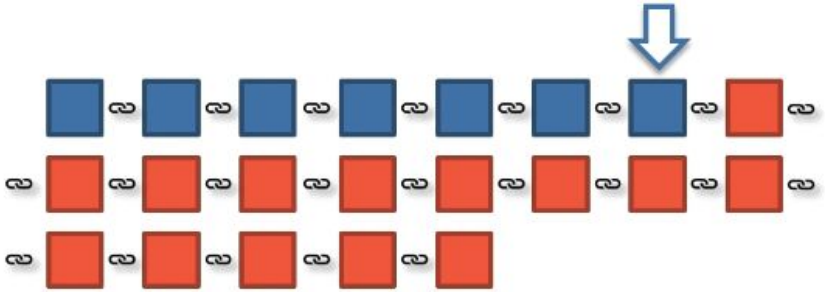


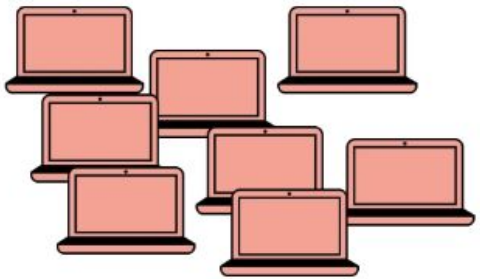


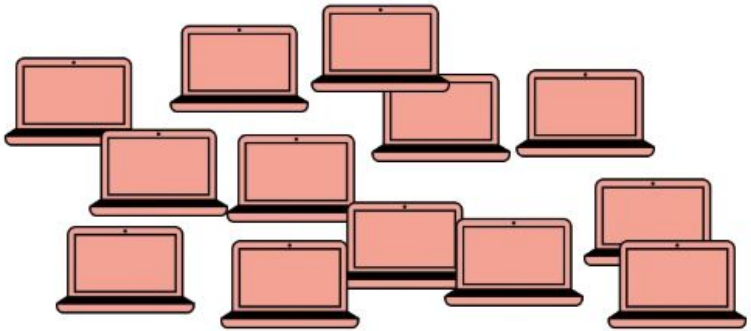
The 51% attack



Double spent occurring!







Deriving the current target

Difficulty = current target / max target

Curr target = `0000000000000000005d97dc0000000000000000000000000000000000000000`

Max target = `00000000FFFF0000000000000000000000000000000000000000000000000000`

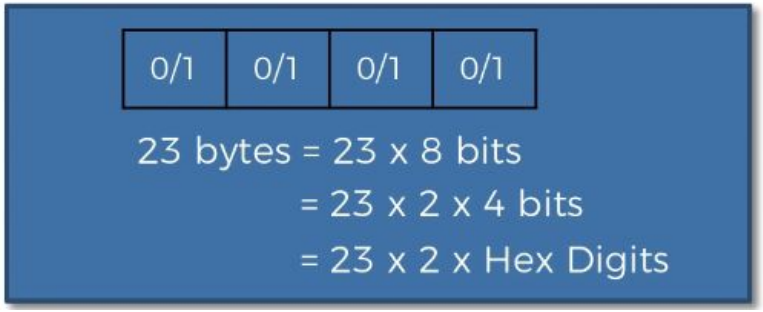
Where is the current target stored?

Bits -> Hex -> Derive target

Bits: 392009692

Bits in Hex: 175D97DC

16*1+7
= 23

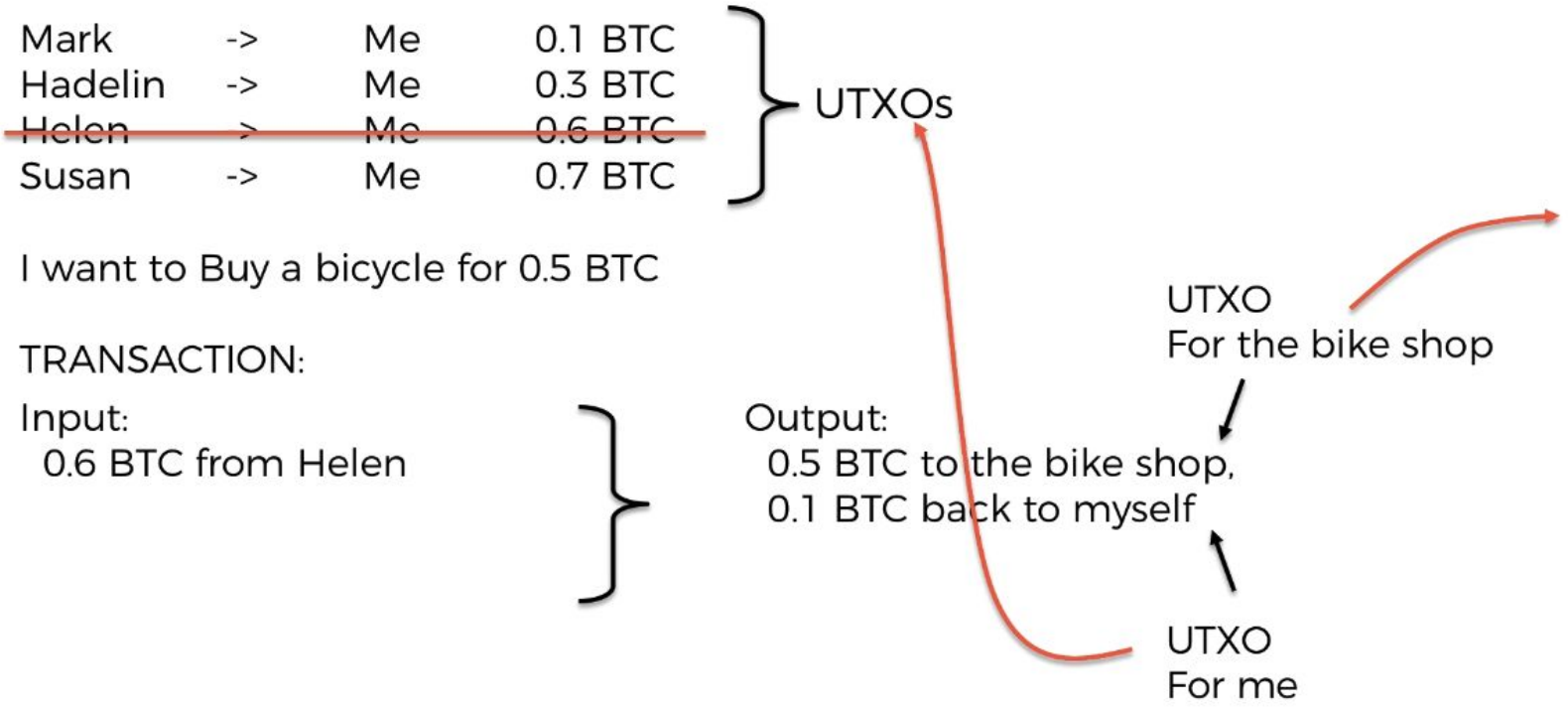


23x2 = 46 Hex Digits

`0000000000000000005D97DC0000000000000000000000000000000000000000`

Add missing zeros
(64-46 = 18)

Transactions and UTX0s



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:

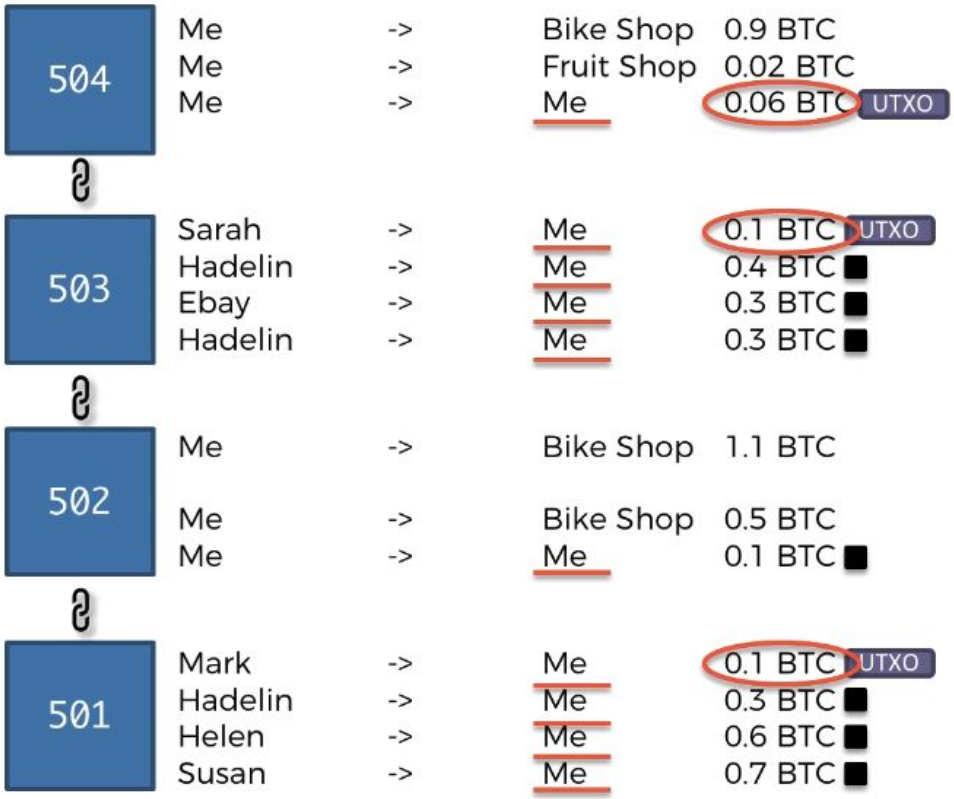
Input:
0.4 BTC from Hadelin,
0.3 BTC from Ebay
0.3 BTC from Hadelin

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

Fees: 0.02 BTC ← UTXO for the miner

UTXO for the bike shop
UTXO for the bike shop
UTXO for me

How wallets work ?

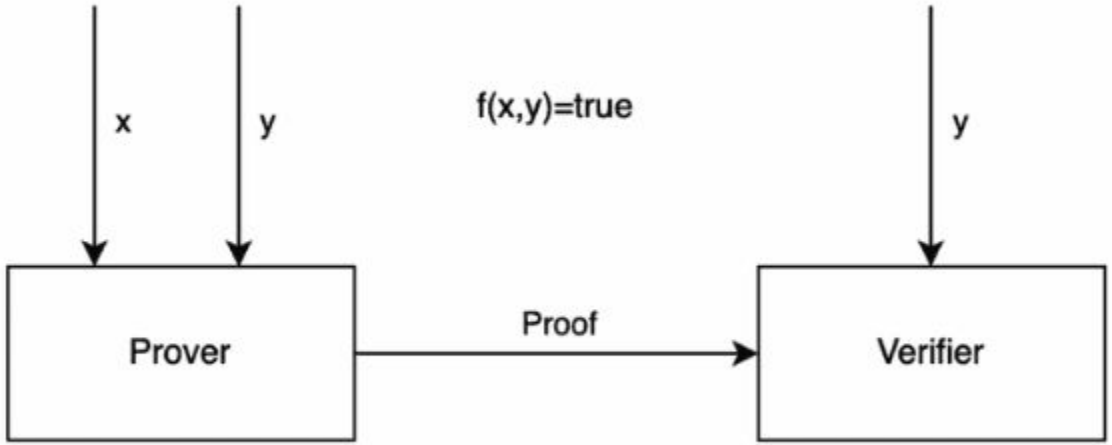


Private & Public Keys

Privacy

Ownership

ZK SNARK explained



https://github.com/Magnum35puc/VanityAddressGenerator

Samouraïe wallet

Cash mixer : Tornado cash

Defi