# Blockchain Technologies

## Decentralization and Bitcoin Mining

# CENTRALIZATION VS. DECENTRALIZATION

- It is an interconnected system where no single entity has complete authority. It is the architecture in which the workloads, both hardware, and software, are distributed among several workstations.

- Decentralization is not all or nothing; almost no system is purely decentralized or purely centralized.

- While the Bitcoin protocol is decentralized, services like Bitcoin exchanges, and wallet software, may be centralized or decentralized to varying degrees.

# DECENTRALIZATION IS NOT ALL-OR-NOTHING

E-mail:

decentralized protocol, but dominated by centralized webmail services

- The mechanism by which Bitcoin achieves decentralization is not purely technical
- It is a combination of technical methods and clever incentive engineering

# DISTRIBUTED CONSENSUS

- **Distributed consensus protocol:**
  - There are $n$ nodes that each have an input value.
  - Some of these nodes are faulty or malicious.
  - A distributed consensus protocol has the following two properties:
    1) It must terminate with all honest nodes in agreement on the value
    2) The value must have been generated by an honest node

# BITCOIN IS A PEER-TO-PEER SYSTEM

- When Alice wants to pay Bob:
  she broadcasts the transaction to all Bitcoin nodes

| signed by Alice |
| --- |
| Pay to $pk_{Bob}$ : H( ) |


Note: Bob’s computer may not be in the picture or online!

problem: out

- The peer-to-peer nodes need consensus on:
  - which transactions were broadcast
  - order in which these transactions were broadcast

# GLOBAL BITCOIN NODES

**REACHABLE BITCOIN NODES**
Updated: Tue Apr 18 05:41:31 2023 CEST

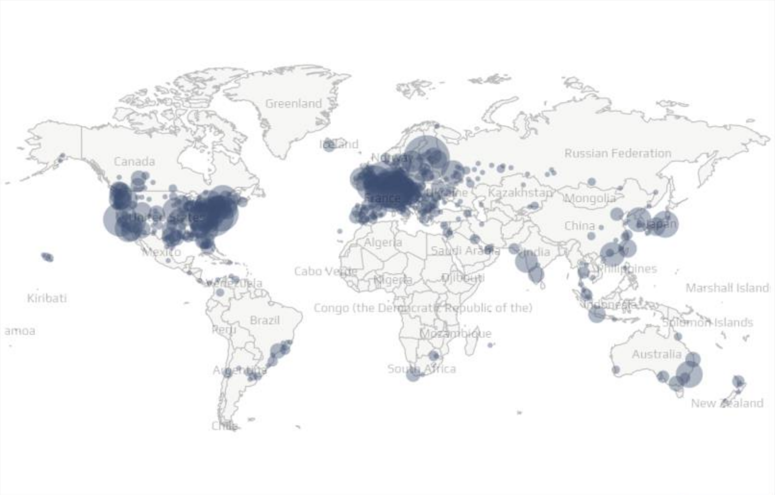
17561 NODES

CHARTS

IPv4: +5.0% / IPv6: +2.6% / .onion: +18.0%

Top 10 countries with their respective number of reachable nodes are as follows.

| RANK | COUNTRY | NODES |
|---|---|---|
| 1 | n/a | 10534 (59.99%) |
| 2 | United States | 1809 (10.30%) |
| 3 | Germany | 1380 (7.86%) |
| 4 | France | 494 (2.81%) |
| 5 | Netherlands | 369 (2.10%) |
| 6 | Canada | 331 (1.88%) |
| 7 | Finland | 273 (1.55%) |
| 8 | United Kingdom | 237 (1.35%) |
| 9 | Russian Federation | 179 (1.02%) |

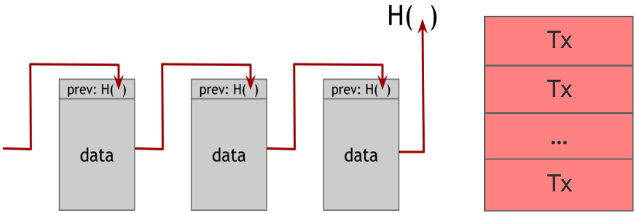



Map shows concentration of reachable Bitcoin nodes found in countries around the world.

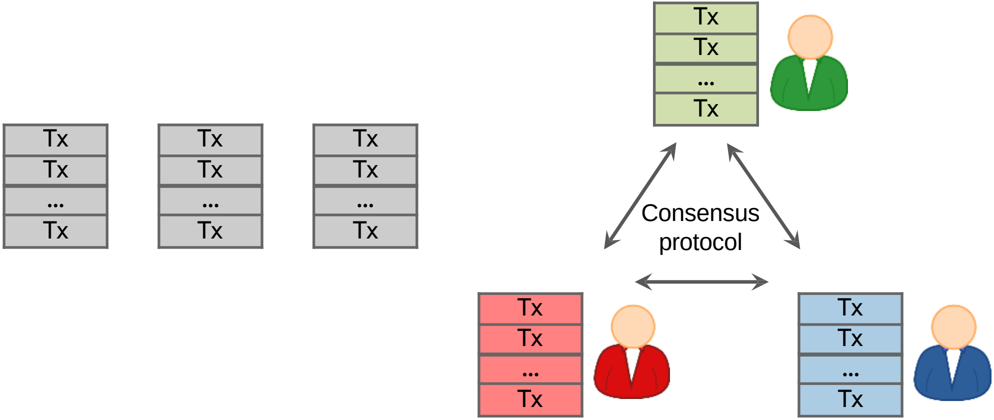
LIVE MAP

# DISTRIBUTED CONSENSUS IN BITCOIN NETWORK

- At any given time:
  - All nodes have a ledger consisting of sequence of blocks of transactions they've reached consensus on.
  - Each node has a set of outstanding transactions it's heard about but have not yet been included on the block chain.
  - So each node might have a slightly different version of the outstanding transaction pool.

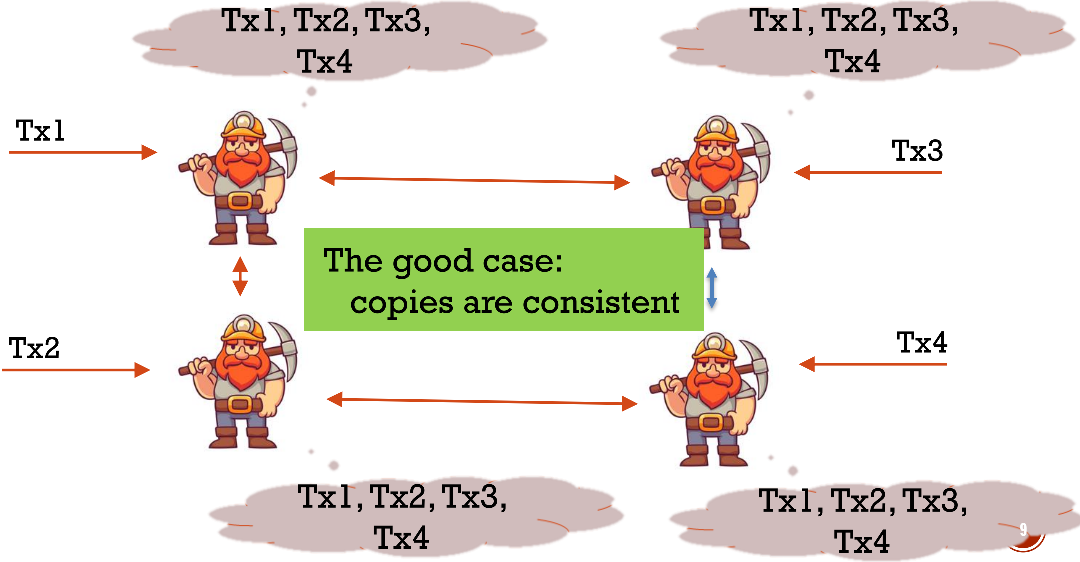


# HOW CONSENSUS COULD WORK IN BITCOIN?

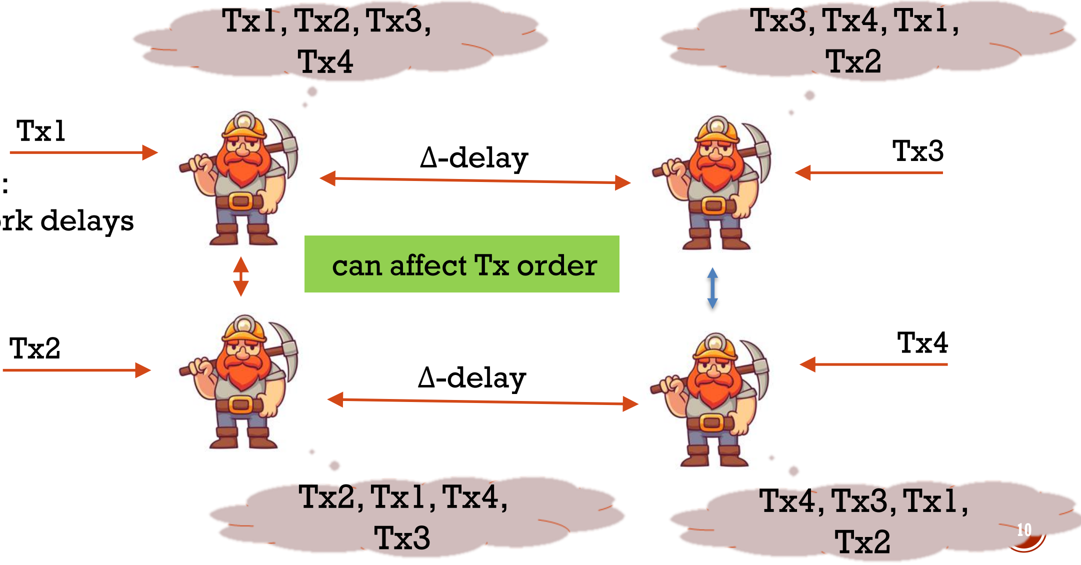



OK to select any valid block, even if proposed by only one node

# WHY IS CONSENSUS A HARD PROBLEM?



# WHY IS CONSENSUS A HARD PROBLEM?



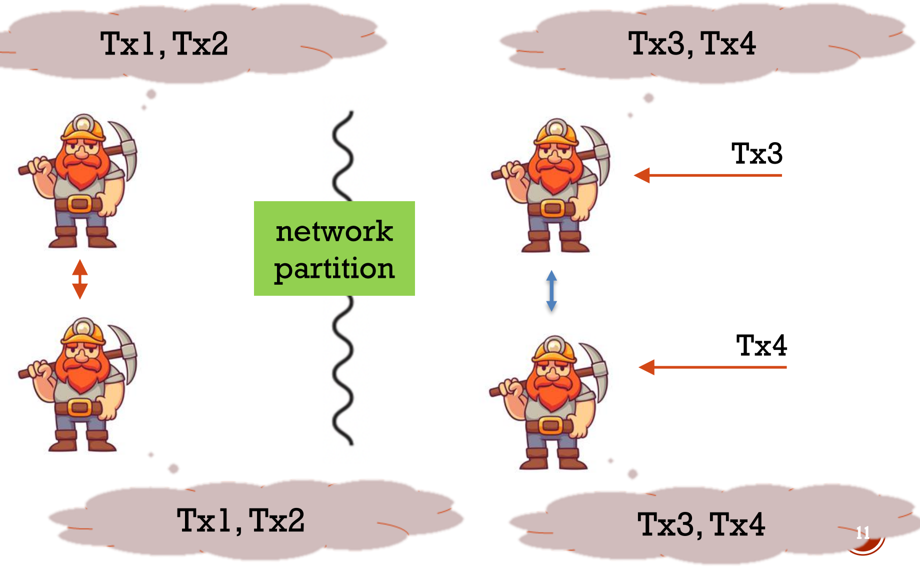

Problems:

- Network delays

# WHY IS CONSENSUS A HARD PROBLEM?

Problems:

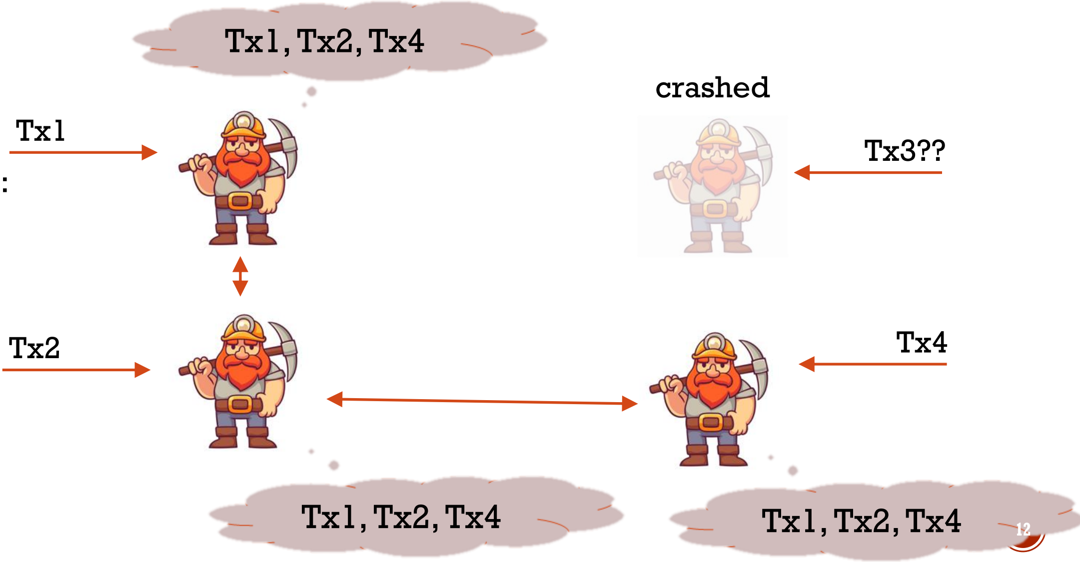
- Network delays
- Network partition



# WHY IS CONSENSUS A HARD PROBLEM?

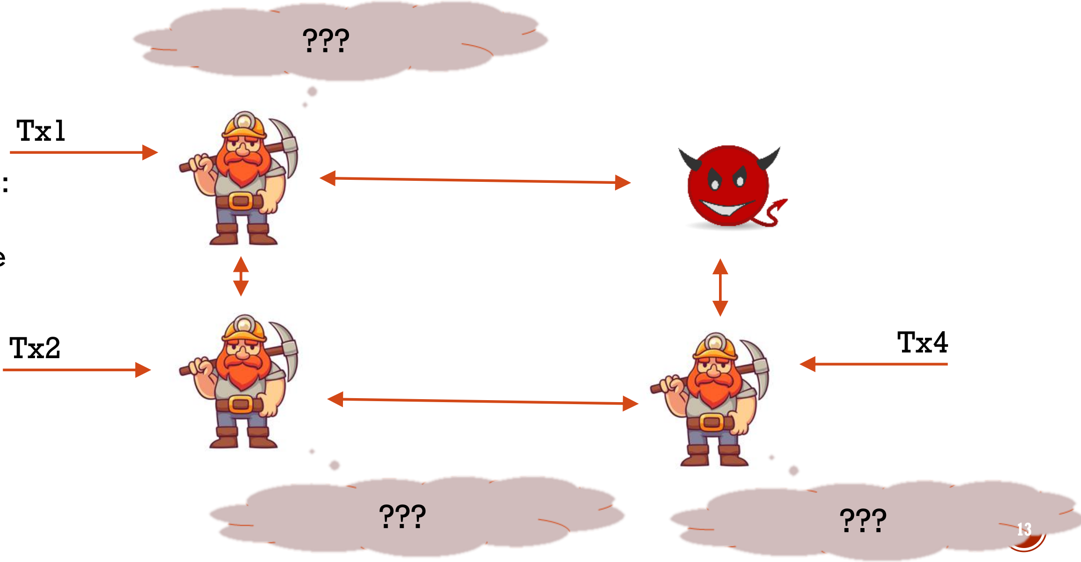
Problems:

- crash



# WHY IS CONSENSUS A HARD PROBLEM?

Problems:
- crash
- malice



# WHY CONSENSUS IS HARD?

- Nodes may crash
- Nodes may be malicious
- Network is imperfect
  - Not all pairs of nodes connected
  - Faults in network
  - Latency
    - There is no notion of global time
      - Not all nodes can agree to a common ordering of events simply based on observing timestamps.
      - So the consensus protocol cannot contain instructions of the form, “The node that sent the first message in step 1 must do X in step 2.”
      - This simply will not work because not all nodes will agree on which message was sent first in the step 1 of the protocol.

# IMPOSSIBILITY RESULTS

- The lack of global time heavily constrains the set of algorithms that can be used in the consensus protocols.
- ***Byzantine Generals Problem***:
  - Byzantine army is separated into divisions, each commanded by a general.
  - The generals communicate by messenger in order to devise a joint plan
  - Some generals are traitors and try to prevent achieving a unified plan.
  - Goal: all of the loyal generals to arrive at the same plan without the traitorous generals being able to cause them to adopt a bad plan.
  - This is impossible if one-third or more of the generals are traitors.
- Fischer-Lynch-Paterson impossibility result:
  - Under some conditions, which include the nodes acting in a deterministic manner, they proved that consensus is impossible with even a single faulty process.

# BYZANTINE GENERALS PROBLEM

- Introduced by Lamport et al. in 1982.
- Problem statement:
  - There are $n$ generals (where $n$ is fixed), one of which is the commander.
  - Some generals are loyal, and some of them can be traitors (including the commander).
  - The commander sends out an order that is either attack or retreat to each general.
  - If the commander is loyal, it sends the same order to all generals.
  - All generals take an action after some time.

Retreat

Attack

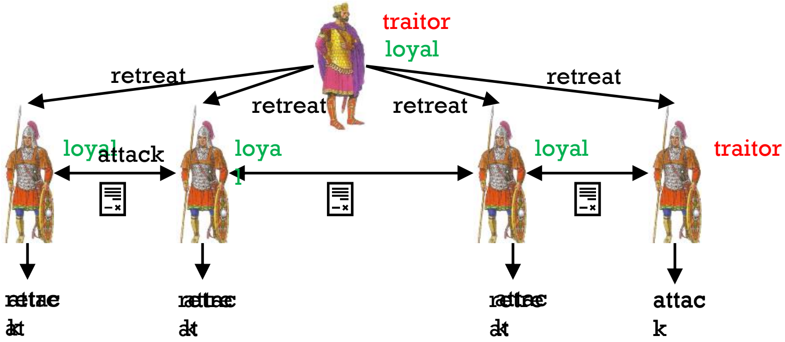
Attack

Retreat

Retreat

# BYZANTINE GENERALS PROBLEM

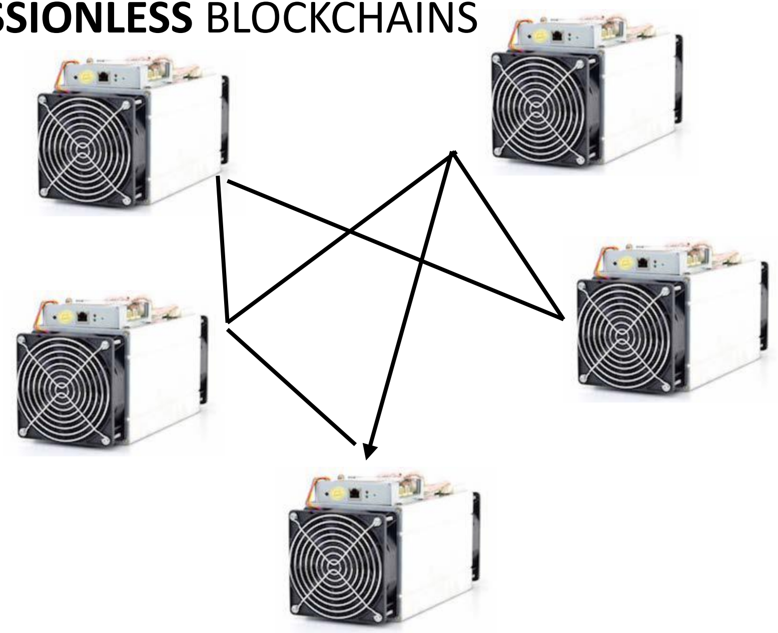
- Goal:
  - All loyal generals must take the **same** action.
  - If the commander is loyal, then all loyal generals must take the action **suggested by the commander.**



# FROM GENERALS TO NODES

- Solution to the Byzantine Generals Problem is a *consensus protocol*.
- When modelling consensus protocols:
  - Generals → Nodes
  - Commander → Leader
  - Loyal → Honest, Traitor → Adversary
    - What can the adversarial nodes do?

# **PERMISSIONLESS** BLOCKCHAINS

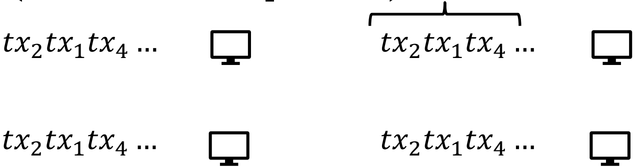


# STATE MACHINE REPLICATION (SMR)

A Centralized Bank


Blockchain (State Machine Replication)

**Log (Ledger):** an ever-growing, linearly-ordered *sequence* of transactions.

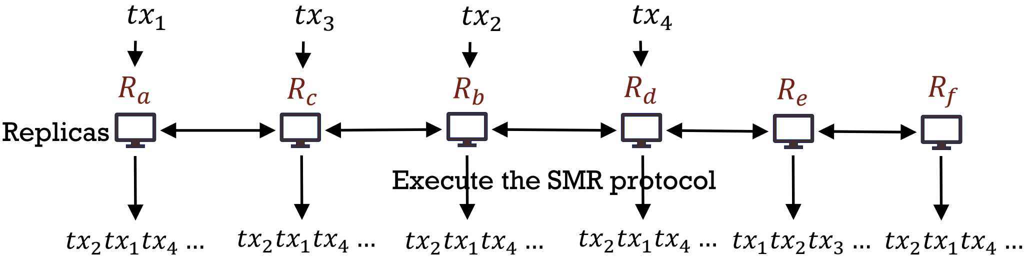


# STATE MACHINE REPLICATION (SMR)

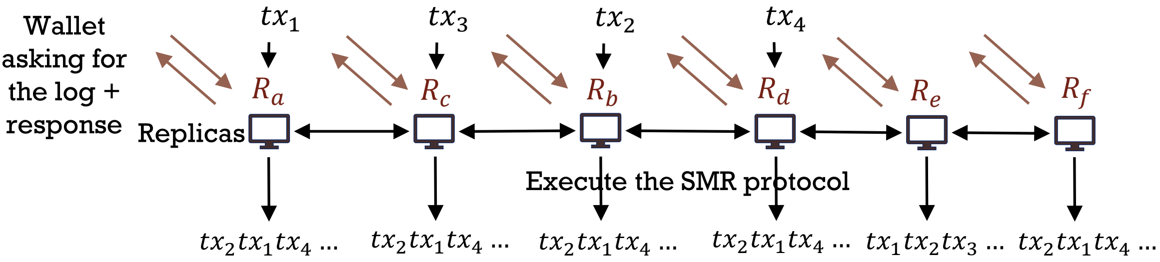
**Two parties of SMR:**

- *Replicas* receive transactions, execute the SMR protocol and determine the log.
- *Clients* are the learners: They communicate with the replicas to learn the log.

**Goal of SMR** is to ensure that the clients learn the *same* log.



# STATE MACHINE REPLICATION (SMR)




$C_1$ $LOG_t^1 = tx_2 tx_1 tx_4 \ldots$

$C_2$ $LOG_t^2 = tx_2 tx_1 tx_4 \ldots$

$C_3$ $LOG_t^3 = tx_2 tx_1 tx_4 \ldots$

$C_4$ $LOG_t^4 = tx_2 tx_1 tx_4 \ldots$

Clients (Wallets)

Clients (Wallets)

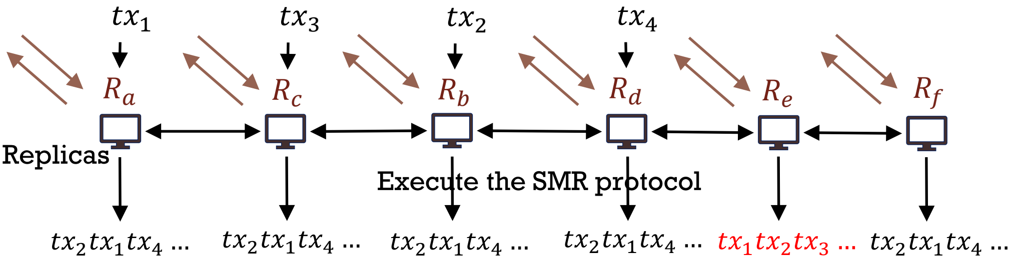
Wallets are an example of a client.

Wallets ask the replicas what the correct log is.

Wallets **do not** execute the SMR protocol and **do not** talk to each other.

# STATE MACHINE REPLICATION (SMR)

Wallet asking for the log + response




$C_1$ $LOG_t^1 = tx_2 tx_1 tx_4 \ldots$

$C_2$ $LOG_t^2 = tx_2 tx_1 tx_4 \ldots$

**How does a wallet learn the correct log from the replicas?**

- It asks the replicas what the correct log is.
- Wallet then accepts the answer given by majority of the replicas as its log.

**Wallet learns the correct log if over half of the replicas are honest!**

Clients (Wallets)

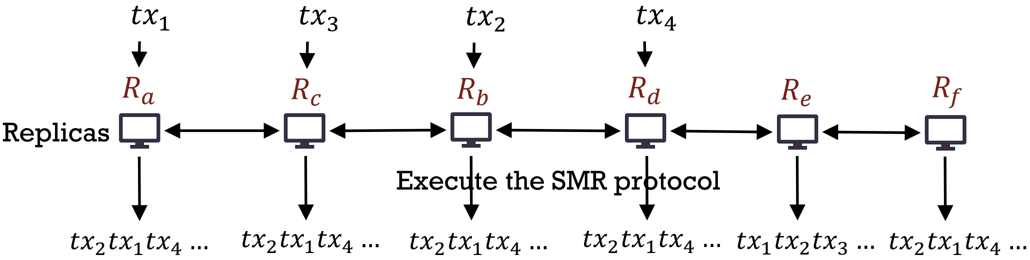
$C_3$ $LOG_t^3 = tx_2 tx_1 tx_4 \ldots$

Clients (Wallets)

$C_4$ $LOG_t^4 = tx_2 tx_1 tx_4 \ldots$

# STATE MACHINE REPLICATION (SMR)

Going forward, we will focus primarily on the replicas and the execution of the SMR by the replicas.



# SMR VS. BYZANTINE GENERALS

- **Single shot vs. Multi-shot**
  - **Byzantine Generals Problem** is single shot consensus. Each node outputs a single value.
  - **State Machine Replication** is multi-shot. Each client *continuously* outputs a log, which is a sequence of transactions (values).
- **Who are the learners?**
  - In **Byzantine Generals Problem**, the nodes executing the protocol are the same as the nodes that output decision values.
  - In **State Machine Replication**, protocol is executed by the replicas, whereas the goal is for the clients to learn the log. Replicas must ensure that the clients learn the same log.

# SECURITY FOR SMR: DEFINITIONS

**Concatenation ($A||B$):**

- Suppose we have sequences $A = tx_1 tx_2$ and $\mathrm{B} = tx_3 tx_4$. What is $A||B$?

$$A||B = tx_1 tx_2 tx_3 tx_4$$

**Prefix relation ($A \preccurlyeq B$):** Sequence $A$ is said to be a prefix of sequence $B$, if there exists a sequence $C$ (that is potentially empty) such that $B = A||C$.

Suppose we have $A = tx_1 tx_2 tx_3 tx_4$, $B = tx_1 tx_2 tx_3$ and $D = tx_1 tx_2 tx_4$.

- Is $B$ a prefix of $A$?
  - Yes
- Is $D$ a prefix of $A$?
  - No

# Security for SMR: Definitions

Two sequences $A$ and $B$ are consistent if either $A \preccurlyeq B$ is true or $B \preccurlyeq A$ is true or both statements are true.

Are these two logs consistent: $LOG^{Alice} = tx_1 tx_2 tx_3 tx_4, LOG^{Bob} = tx_1 tx_2 tx_3$?

- Yes!

What about $LOG^{Alice} = tx_1 tx_2 tx_3, LOG^{Bob} = tx_1 tx_2 tx_3 tx_4$?

- Yes!

What about $LOG^{Alice} = tx_1 tx_2, LOG^{Bob} = tx_1 tx_3$?

- No!

# SECURITY FOR SMR

Let $LOG_t^i$ denote the log outputted by a client $i$ at time $t$.

Then, a **secure** SMR protocol satisfies the following guarantees:

**Safety (Consistency):**

- For any two clients $i$ and $j$, and times $t$ and $s$: either $LOG_t^i \preccurlyeq LOG_s^j$ is true or $LOG_s^j \preccurlyeq LOG_t^i$ is true or both (Logs are consistent).

how it that possible?

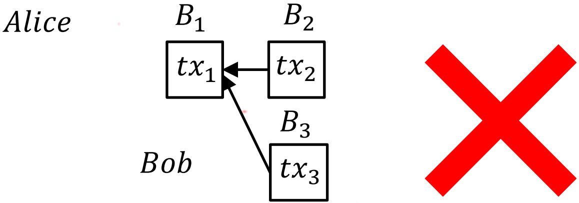
No double spend

**Liveness:**

- If a transaction $tx$ is input to an honest replica at some time $t$, then for all clients $i$, and times $s \geq t + T_{conf}$: $tx \in LOG_s^i$.

No censorship

# SECURITY FOR SMR




Safety violation!!

# BLOCKCHAIN PROTOCOLS

Transactions are often batched into blocks to enhance throughput.

Let $\boldsymbol{ch}_t^i$ denote the chain *accepted* by a client $i$ at time $t$.

**Safety (Consistency):**

- For any two clients $i$ and $j$, and times $t$ and $s$: either $\boldsymbol{ch}_t^i \preccurlyeq \boldsymbol{ch}_s^j$ is true or $\boldsymbol{ch}_s^j \preccurlyeq \boldsymbol{ch}_t^i$ is true or both (Chains are consistent).
- **Liveness:** If a transaction $tx$ is input to an honest replica at some time $t$, then for all clients $i$, and times $s \geq t + T_{conf}$: $tx \in \boldsymbol{ch}_s^i$.

# UNDERSTANDING IMPOSSIBILITY RESULTS

- These results say more about the model than about the problem.
- The models were developed to study systems like distributed databases.
- Bitcoin consensus works better in practice than in theory.
- Theory is still catching up.
- BUT theory is important, can help predict unforeseen attacks.

# THINGS BITCOIN DOES DIFFERENTLY

- Introduces incentives
  - Possible because it's a currency!
    - has a natural mechanism to incentivize participants to act honestly
    - Bitcoin doesn't quite solve the distributed consensus problem in a general sense, but it solves it in the specific context of a currency system.
- Embraces randomness
  - Does away with the notion of a specific end-point
  - Consensus happens over long time scales — about 1 hour

# Why identity?

## WITH IDENTITY THE DESIGN OF DISTRIBUTED CONSENSUS PROTOCOL WOULD BE EASIER

**Pragmatic: some protocols need node IDs**

identities would allow us to put in the protocol instructions of the form, "Now the node with the lowest numerical ID should take some step." Without identities, the set of possible instructions is more constrained.

**Security: assume less than 50% malicious**

If nodes were identified and it weren't trivial to create new node identities, then we could make assumptions about the number of nodes that are malicious, and we could derive security properties based on those numbers.

# CONSENSUS WITHOUT IDENTITIES

- Bitcoin nodes do not have persistent, long-term identities:
- Reason 1. In a P2P system, there is no central authority to assign identities to participants and verify that they're not creating new nodes at will.
  - ***Sybil Attack***
    - Sybils are just copies of nodes that a malicious adversary can create to look like there are a lot of different participants, when in fact all those pseudo-participants are really controlled by the same adversary.
- Reason 2. Pseudonymity is inherently a goal of Bitcoin.
  - Nobody is forced to reveal their real-life identity, like their name or IP address, in order to participate.
    - This is a central feature of Bitcoin's design

# SYBIL ATTACK

How to select the nodes that participate in consensus?

**Two variants:**

- *Permissioned:* There is a *fixed* set of nodes (previous lecture).
- *Permissionless*: Anyone satisfying certain criteria can participate.

Can we accept any node that has a signing key to participate in consensus?

Sybil
Attack!

# SYBIL RESISTANCE

Consensus protocols with Sybil resistance are typically based on a bounded (scarce) resource:

| | Resource dedicated to the protocol | Some Example Blockchains |
|---|---|---|
| **Proof-of-Work** | Total computational power | Bitcoin, PoW Ethereum… |
| **Proof-of-Stake** | Total number of coins | Algorand, Cardano, Cosmos, PoS Ethereum… |
| **Proof-of-Space/Time** | Total storage across time | Chia, Filecoin… |

How does Proof-of-Work prevent Sybil attacks?

If more than 0.5, it's not sybil attack, it's 51% attack.

We assume that the adversary controls a small fraction of the scarce resource!

# IMPLICIT CONSENSUS

- To compensate lack of identities: select a random node through a process like lottery
- In each round, random node is picked
- This node proposes the next block in the chain
- Other nodes implicitly accept/reject this block
  - by either extending it (accept)
  - or ignoring it and extending chain from earlier block (reject)
- Every block contains hash of the block it extends

# CONSENSUS ALGORITHM (SIMPLIFIED)

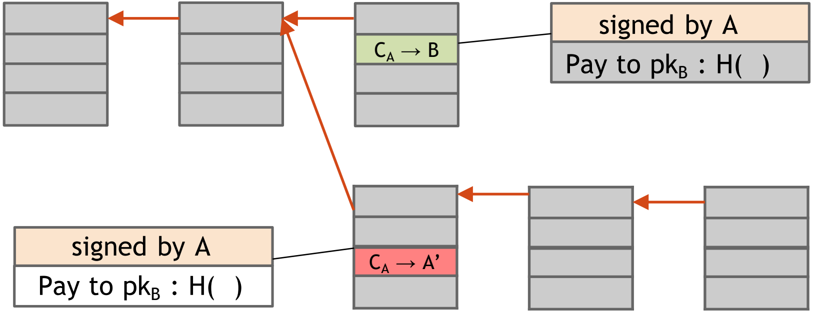
1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. In each round a random node gets to broadcast its block.
4. Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures).
5. Nodes express their acceptance of the block by including its hash in the next block they create.

# WHAT CAN A MALICIOUS NODE DO?

- **Stealing Bitcoins:**
  - Stealing another user's coins would require to forge the owner's signature.
- **Denial-of-Service:**
  - Alice wants to prevent Bob's transactions from being included in block chain.
  - Alice may prevent for one or more rounds.
  - Eventually, honest node will be picked, who will include Bob's transaction in proposed block.
- **Double-Spend Attack:**
  - Alice purchases service from Bob and pays in coins.
  - Alice creates transaction and broadcasts it to the network.
  - Later, Alice attempts to pay same coin to one of her accounts.

# DOUBLE-SPENDING ATTACK

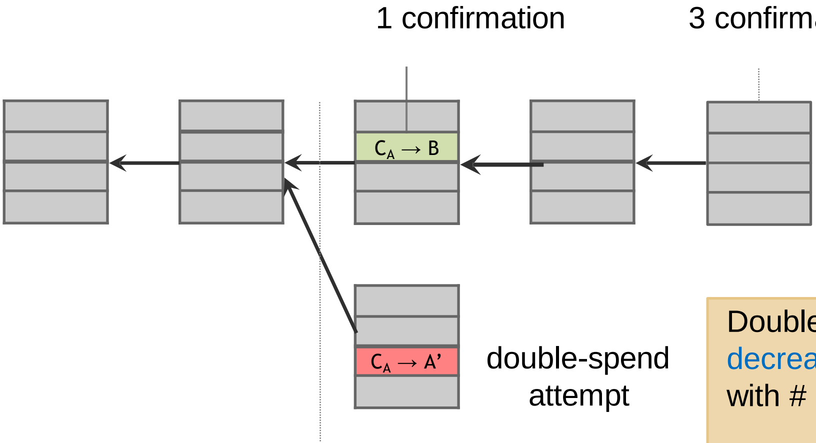
Double spent is possible in case of fork.




block DAG

Honest nodes will extend the longest valid branch

# FROM BOB THE MERCHANT'S POINT OF VIEW




Hear about $C_A \rightarrow B$ transaction
1 confirmations

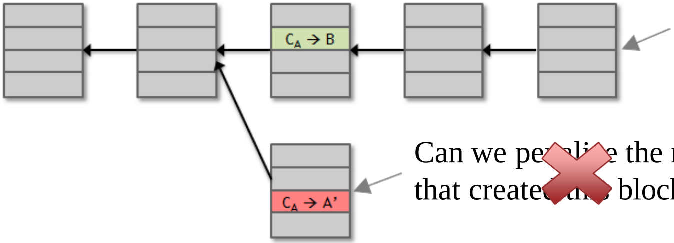
Double-spend probability decreases exponentially with # of confirmations

Most common heuristic:
6 confirmations

Not good for real time transactions.

# ASSUMPTION OF HONESTY IS PROBLEMATIC

- Can we give nodes incentives for behaving honestly?




Can we reward nodes that created these blocks?

Can we penalize the node that created this block?

- Everything so far is just a distributed consensus protocol.
- But now we utilize the fact that the currency has value.

# INCENTIVE 1: BLOCK REWARD

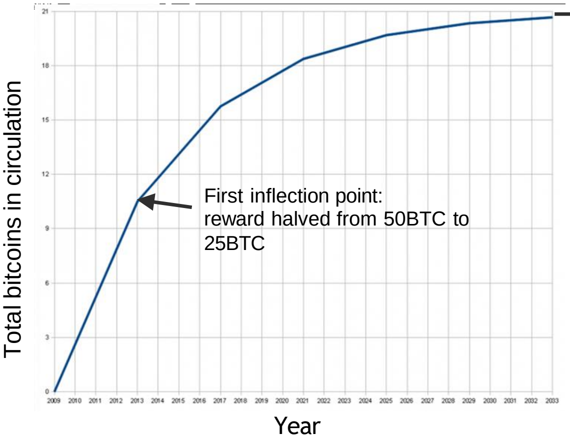
- Creator of block gets to
  - include special coin-creation transaction in the block
  - choose recipient address of this transaction

- Value is fixed: currently 6.25 BTC, halves every 4 years.

- Block creator gets to collect the reward only if the block ends up on long-term consensus branch!

- Note: This is the only way to create new Bitcoins!

# THERE'S A FINITE SUPPLY OF BITCOINS




Total supply: 21 million

- Block reward is how new Bitcoins are created
- Runs out in 2140. No new Bitcoins unless rules change

# INCENTIVE 2: TRANSACTION FEES

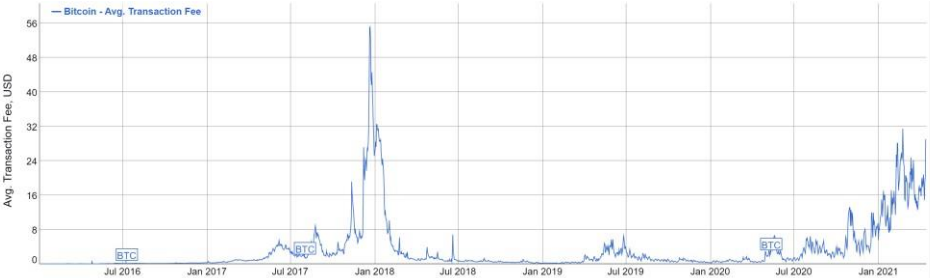
- Creator of transaction can choose to make output value less than input value.

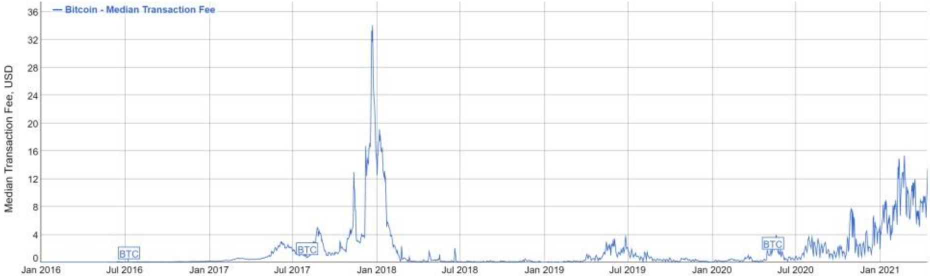
- Remainder is a transaction fee and goes to block creator.

- Purely voluntary, like a tip.

Choose the transactions with hig

# AVERAGE TRANSACTION FEE



# RANDOM NODE: PROOF OF WORK

- To approximate selecting a random node:
  - select nodes in proportion to a resource
  - that no one can monopolize (we hope)

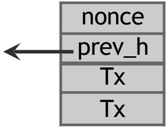
- In proportion to computing power: proof-of-work

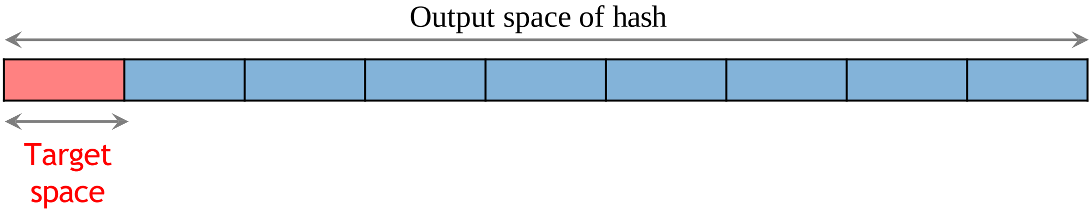
- Equivalent view of proof-of-work
  - Select nodes in proportion to computing power
  - Let nodes compete for right to create block
  - Make it moderately hard to create new identities

# HASH PUZZLES

To create block, find nonce s.t.

H(nonce ‖ prev_hash ‖ merkle_root) < Hash Target




If hash function is secure:
only way to succeed is to try enough nonces until you get lucky

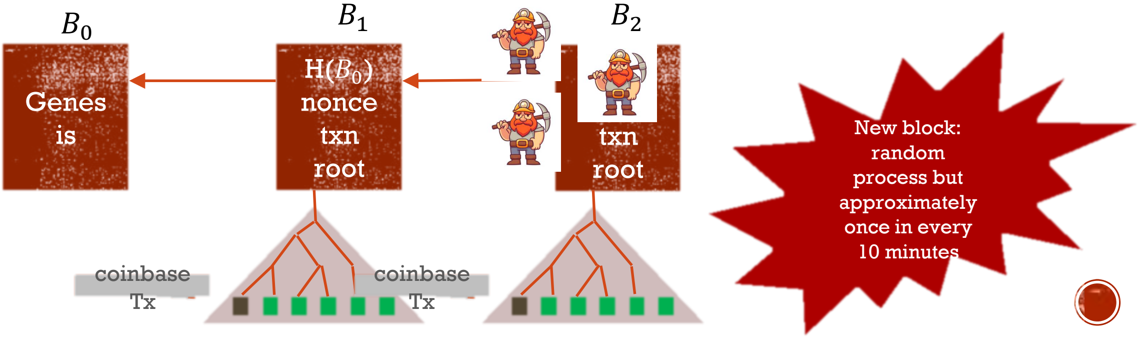
# BITCOIN: MINING

To mine a new block, a miner must find *nonce* such that

$$H\left(h_{prev}, txn\ root, nonce\right) < \text{Target} = \frac{2^{256}}{D}$$

**Difficulty:** How many nonces on average miners try until finding a block?

Each miner tries different nonces until one of them finds a nonce that satisfies the above equation.



# PROPERTIES OF HASH PUZZLES

Property 1: Must be (moderately) difficult to compute

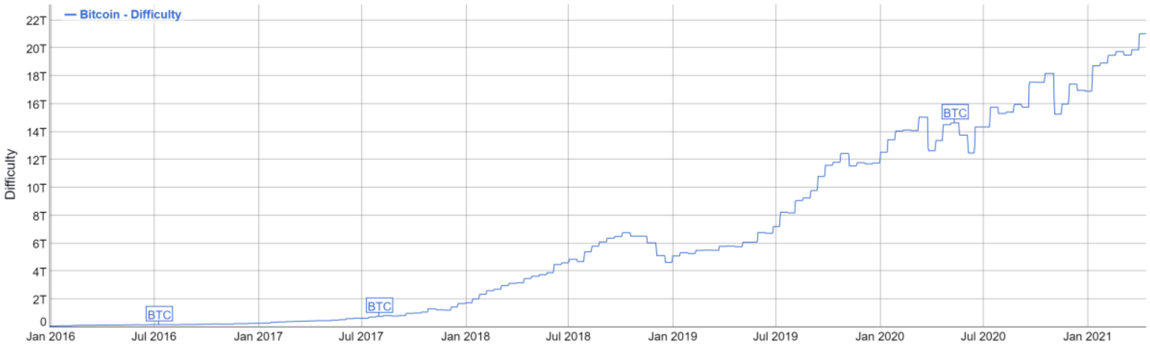
Property 2: The Cost must be "parameterizable"

Property 3: Must be trivial to verify

# 1. DIFFICULT TO COMPUTE

ex:mentioned: 2 ^ 56ac

➢ It takes about $2^{32} \times$ Difficulty hashes to find a block.




➢ Only some nodes bother to compete: **Miners**

# 2. Parametrizable Cost

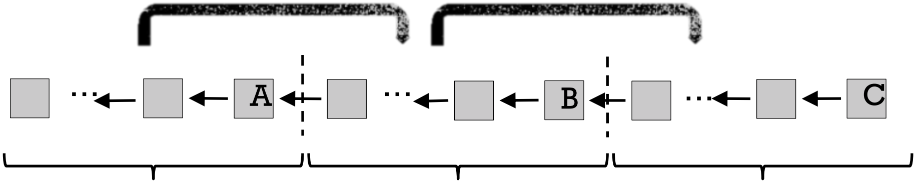
- Nodes automatically re-calculate the target every 2016 blocks (about every two weeks).
- Goal: average time between blocks = 10 minutes
- Adjust difficulty to meet 10-minute goal.
  - Current difficulty is around $2^{44} \rightarrow 2^{76}$ hash/block
  - Maximum difficulty is $2^{224}$

target space

bitcoin difficulty

# BITCOIN: DIFFICULTY ADJUSTMENT

New target: $T_2 = T_1 \dfrac{t_1}{2016 \times 10\ mins}$ New target: $T_3 = T_2 \dfrac{t_2}{2016 \times 10\ mins}$




| | | |
|---|---|---|
| 2016 blocks<br>Time it took to mine:<br>$t_1$(min)<br>Target: $T_1$ | 2016 blocks<br>Time it took to mine:<br>$t_2$(min)<br>Target: $T_2$ | 2016 blocks<br>Time it took to mine:<br>$t_3$(min)<br>Target: $T_3$ |

New target is not allowed to be more than 4x old target.
New target is not allowed to be less than ¼ x old target.

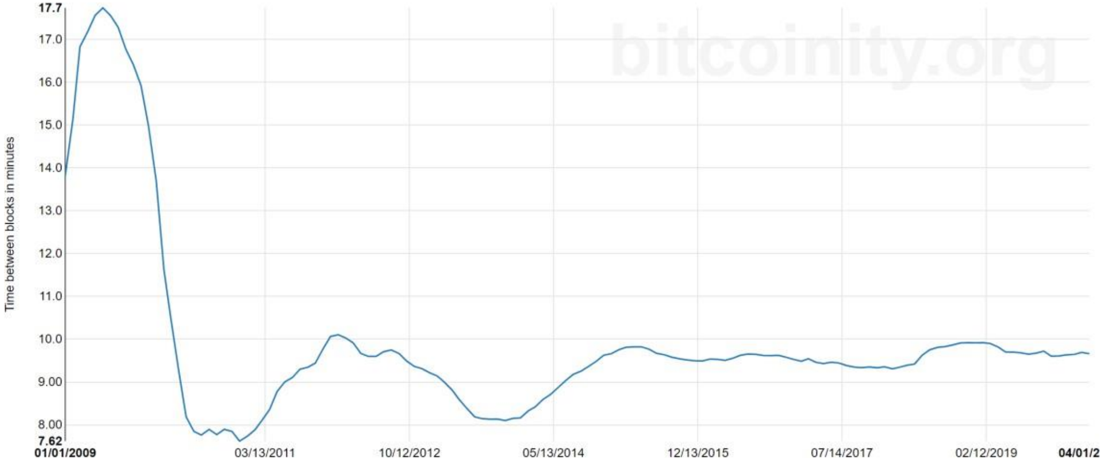
# SOLVING HASH PUZZLES IS PROBABILISTIC

Prob (Alice wins next block) =
fraction of global hash power she controls

For individual miner:

$$\text{mean time to find block} = \frac{\text{10 minutes}}{\text{fraction of hash power}}$$

# AVERAGE TIME TO MINE A BLOCK



# 3. TRIVIAL TO VERIFY

- Nonce is published as part of block.

- Other miners simply verify that

$$H(nonce \parallel prev\_hash \parallel merkle\_root) < target$$

- This is an important property because, once again, it allows us to get rid of centralization.
  - We don't need any centralized authority verifying that miners are doing their job correctly.
  - Any node or any miner can instantly verify that a block found by another miner satisfies this proof-of-work property

# BLOCKS VERIFYING

A block is valid if condition is true

```
IF (SHA256(SHA256(HDR)) < max(DIFFICULTY)/ DIFFICULTY)
    return;
```

two hashes

# BLOCK HEADER

An 80-byte block header contains:

- 4 bytes: version
- 32 bytes: previous block hash
- 32 bytes: merkle tree of transactions
- 4 bytes: timestamp
- 4 bytes: difficulty target
- 4 bytes: nonce

ex: change 10 min interval to 20 minutes

# SECURITY

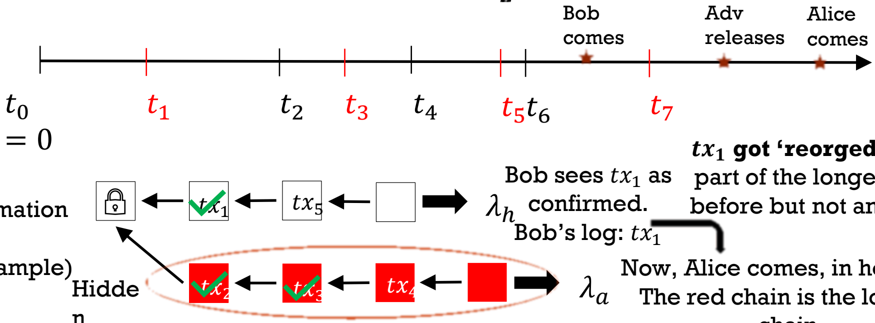
Can we show that Bitcoin is secure under synchrony against a Byzantine adversary?

What would be the best possible resilience?

$$\beta < 1/2?$$

**Fraction of the mining power controlled by the adversary.**

# NAKAMOTO'S PRIVATE ATTACK: $\beta \geq 1/2$




k deep confirmation rule
(k=3 in our example)

Bob sees $tx_1$ as confirmed.
Bob's log: $tx_1$

**$tx_1$ got 'reorged':** It was part of the longest chain before but not anymore!!

Now, Alice comes, in her view: The red chain is the longest chain.
$tx_1$ is not confirmed!
Alice's log: $tx_2 tx_3$

Private attack succeeds!

Private attack (mostly) succeeds if $\lambda_a \geq \lambda_h$, i.e., if $\beta \geq 1 - \beta$, i.e., if $\beta \geq \frac{1}{2}$.

Private attack (mostly) fails if $\lambda_a < \lambda_h$, i.e., if $\beta < 1 - \beta$, i.e., if $\beta < \frac{1}{2}$.

checkpointing in ETH to prevent this attack (Not 100%)

Can another attack succeed?

A Peer-to-Peer Electronic Cash System (2008)

# WHAT CAN A "51% ATTACKER" DO?

- **Key security assumption:** Attacks infeasible if majority of miners weighted by hash power follow the protocol.
- What would happen if consensus failed and there was in fact an attacker who controls 51 percent or more of the mining power?
  - Steal coins from existing address? ✗
  - Suppress some transactions?
    - From the block chain ✓
    - From the P2P network ✗ ~~Can't affect network topology.~~
  - Change the block reward? ✗ After a while, honest nodes re
  - Destroy confidence in Bitcoin? ✓✓

# BITCOIN GOLD 51% ATTACK

- Bitcoin Gold (BTG) is a hard fork of Bitcoin.

- The stated purpose of the hard fork is to change the proof of work algorithm so that ASICs which are used to mine Bitcoin cannot be used to mine the Bitcoin Gold blockchain in the hopes that enabling mining on commonly available graphics cards will democratize and decentralize the mining and distribution of the cryptocurrency.

- In May 2018, Bitcoin Gold was hit by a 51% hashing attack by an unknown actor. During the attack, 388,000 BTG (worth approximately US$18 million) was double-spent.

- Bitcoin Gold suffered from 51% attacks again in January 2020.

# OTHER 51% ATTACKS

digital currency initiative | mit media lab

about research education events communications github

## 51% attacks

- btg counterattack (jan/feb 2020)
- bitcoin gold (btg) 51% attack (jan 2020)
- vertcoin (vtc) 51% attack (dec 2019)
- expanse (exp) 51% attack (jul 2019)
- litecoin cash (lcc) 51% attack (jul 2019)

# CPU MINING

A block is valid if condition is true

```
IF (SHA256(SHA256(HDR)) < max(DIFFICULTY)/ DIFFICULTY)
    return;
```

two hashes

Throughput on a high-end PC = 2 **GHz ≈ $2^{32}$** Hash/s

**500,000+ years** to find a block today!

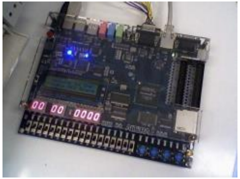
# GPU MINING

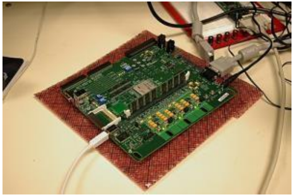


- GPUs designed for high-performance graphics
    - high parallelism
    - high throughput
- First used for Bitcoin in October 2010

# GPU MINING RIG

# FPGA MINING






- **F**ield **P**rogrammable **G**ate **A**rea
- First used for Bitcoin in June 2011
- Implemented in Verilog

# FPGA MINING

# ASIC MINING

BITMAIN

IN STOCK

BITMAIN ANTMINER S19 PRO - 110TH/S

**SKU:** ANTMINER S19 PRO

$4600.00

IN STOCK

**268 SOLD / LIMIT 5 PER CUSTOMER**

**QUANTITY**

\- 1 +

**Pre-Order Terms:** This is a pre-order. 28nm ASIC bitcoin mining hardware products are shipped according to placement in the order queue, and delivery may take 3 months or more after order. All sales are final.

THE LEOPARD

DETAILS :

- **2,5 TH/s**
- Dimensions:
  15" x 13.3" x 13.7"
  (38cm x 34cm x 35cm)
- 28nm ASIC technology
- Silent Cooling
- In-built WiFi Connection (without Antenna)
- Less than 750 watt (0.3 per GH)
- 1 Year Guarantee
- $ 5.800

COMES WITH :

1. Power Supply
2. Free Remote Power Outlet & Smartphone App
3. Free User Guide
4. Free Personal Assistance for Setup

SHIPPING :

- Worldwide, Express
- Included in the price
- Available:
  100 Units: Shipping April (Week 3)

# ASIC MINING

- Special purpose
  - less than 10x performance improvement expected
- Designed to be run constantly for life
- Require significant expertise, long lead-times
- Perhaps the fastest chip development ever!

# PROFESSIONAL MINING CENTERS

Needs:

- cheap power
- good network
- cool climate

BitFury mining center, Republic of Georgia

# EVOLUTION OF MINING

CPU

GPU

FPGA

ASIC

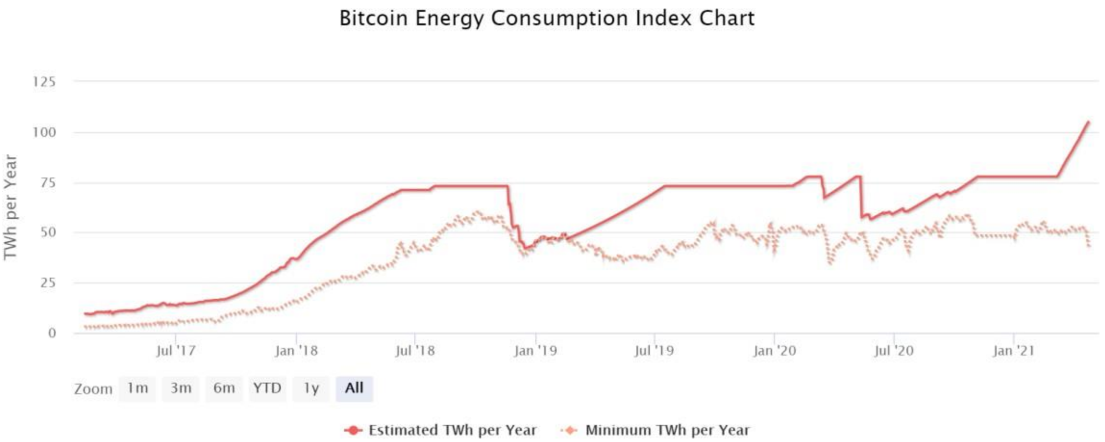
gold pan

sluice box

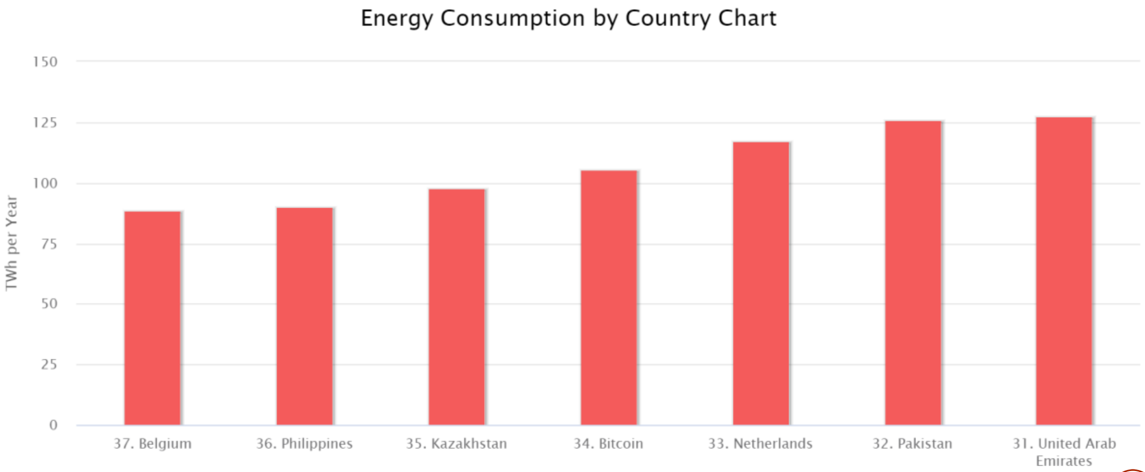
placer mining

pit mining

# BITCOIN ENERGY CONSUMPTION



# BITCOIN ENERGY CONSUMPTION



# MINING ECONOMICS

| If mining reward (block reward + Tx fees) | > | mining cost (hardware + operational cost) | → | Profit |
|---|---|---|---|---|

- Operational Costs: electricity, cooling, …
- Complications:
  - fixed vs. variable costs
  - reward depends on global hash rate
  - cost in USD vs. reward in Bitcoins → Exchange rate varies fast
  - being an honest miner is not provably optimal!

- Actually analyzing whether it makes sense to mine is a complicated game theory problem.

# MINING PROFITABILITY

## Currency

BTC | ETH | ETC | XMR | ZEC | DASH | LTC

Calculated for
1 BTC = $ 49,552.19

**Hashing Power**
110 TH/s

**Power consumption (w)**
3250

**Cost per KWh ($)**
0.01

**Pool Fee (%)**
1

**PROFIT PER MONTH**
**$ 850.32**

| | | | |
|---|---|---|---|
| Day | Profit per day $ 28.34 Pool Fee $ 0.2942 | Mined/day Ƀ 0.0005937 | Power cost/Day $ 0.7800 |
| Week | Profit per week $ 198.41 Pool Fee $ 2.06 | Mined/week Ƀ 0.004156 | Power cost/Week $ 5.46 |
| Month | Profit per month $ 850.32 Pool Fee $ 8.83 | Mined/month Ƀ 0.01781 | Power cost/Month $ 23.40 |
| Year | Profit per year $ 10.35 k Pool Fee $ 107.38 | Mined/year Ƀ 0.2167 | Power cost/Year $ 284.70 |

# MINING UNCERTAINTY

- Being a small miner
- Example: Antminer S19 pro
- Cost: ~ USD 4,600
- Hash power: 110 TH/s

  Fraction of total hash rate = $110/145{,}000{,}000 \approx 7.6 \times 10^{-7}$

  Expected time to find a block: ~25 years!

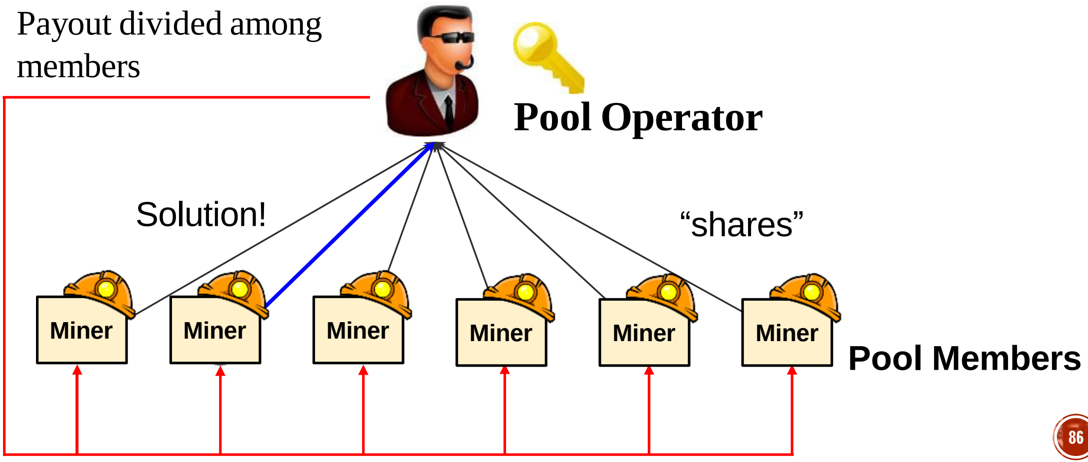
How to compute this?

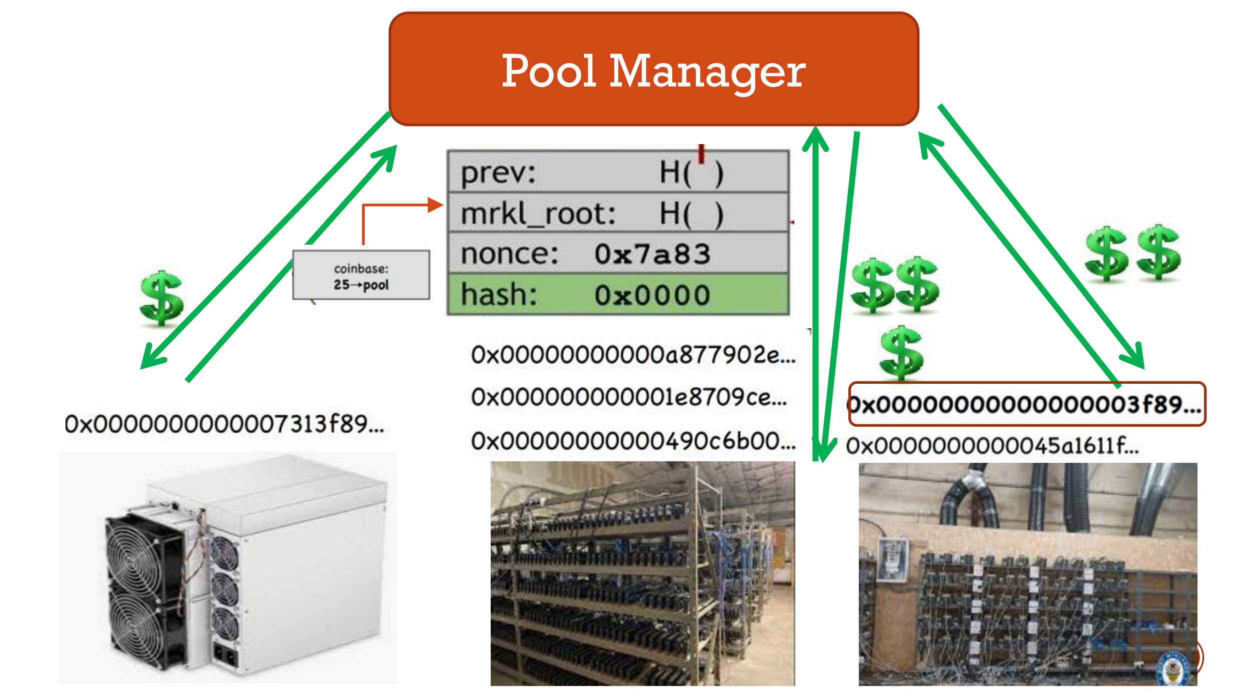
# MINING POOLS

- Goal: pool participants all attempt to mine a block with the same coinbase recipient
  - send money to key owned by pool manager
- Distribute revenues to members based on how much work they have performed
  - minus a cut for pool manager

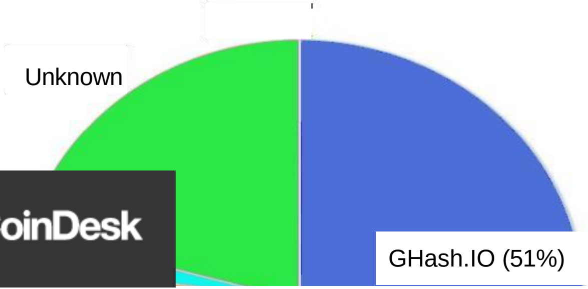
# MINING SHARES

- Idea: Prove work with near-valid blocks (shares)




Pool Manager
prev: H( )
mrkl_root: H( )
nonce: 0x7a83
hash: 0x0000
coinbase:
25→pool
0x00000000000a877902e...
0x000000000001e8709ce...
0x0000000000490c6b00...
0x000000000007313f89...
0x00000000000000003f89...
0x000000000045a1611f...

# MINING POOLS




**June 12, 2014**

**GHash.IO large mining pool crisis**

CoinDesk

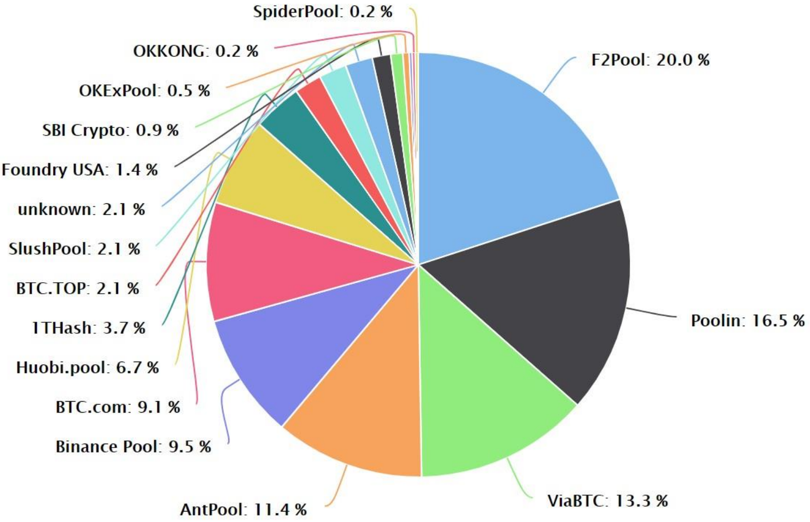
MINING • NEWS

## GHash Commits to 40% Hashrate Cap at Bitcoin Mining Summit

To increase the credit, so increase the price

Stan Higgins | Published on July 16, 2014 at 18:40 GMT

# MINING POOLS



# MINING POOLS

| | Pool | Hashrate Share | Hashrate | Blocks Mined | Empty Blocks Count | Empty Blocks Percentage | Avg. Block Size (Bytes) | Avg. Tx Fees Per Block (BTC) | Tx Fees % of Block Reward |
|---|---|---|---|---|---|---|---|---|---|
| 0 | NETWORK | 100.00 % | 144.77 EH/s | 430 | 3 | 0.70 % | 1,315,035 | 1.55783701 | 24.93 % |
| 1 | F2Pool | 20.00 % | 28.95 EH/s | 86 | 0 | 0.00 % | 1,322,179 | 1.52945747 | 24.47 % |
| 2 | Poolin | 16.51 % | 23.90 EH/s | 71 | 1 | 1.41 % | 1,300,951 | 1.58942627 | 25.43 % |
| 3 | ViaBTC | 13.26 % | 19.19 EH/s | 57 | 1 | 1.75 % | 1,299,086 | 1.50795192 | 24.13 % |
| 4 | AntPool | 11.40 % | 16.50 EH/s | 49 | 0 | 0.00 % | 1,319,934 | 1.59100892 | 25.46 % |
| 5 | Binance Pool | 9.53 % | 13.80 EH/s | 41 | 0 | 0.00 % | 1,320,313 | 1.46012748 | 23.36 % |
| 6 | BTC.com | 9.07 % | 13.13 EH/s | 39 | 0 | 0.00 % | 1,319,946 | 1.66646046 | 26.66 % |
| 7 | Huobi.pool | 6.74 % | 9.76 EH/s | 29 | 0 | 0.00 % | 1,317,926 | 1.62328039 | 25.97 % |
| 8 | 1THash | 3.72 % | 5.39 EH/s | 16 | 0 | 0.00 % | 1,360,034 | 1.62348409 | 25.98 % |
| 9 | BTC.TOP | 2.09 % | 3.03 EH/s | 9 | 1 | 11.11 % | 1,117,161 | 1.41602713 | 22.66 % |
| 10 | SlushPool | 2.09 % | 3.03 EH/s | 9 | 0 | 0.00 % | 1,308,928 | 1.38203378 | 22.11 % |