How Bitcoin achieves Decentralization

- Centralization vs. Decentralization
- Distributed Consensus
- Consensus without Identity, using a Block Chain
- Incentives and Proof of Work
- Putting it all together

How Bitcoin achieves Decentralization

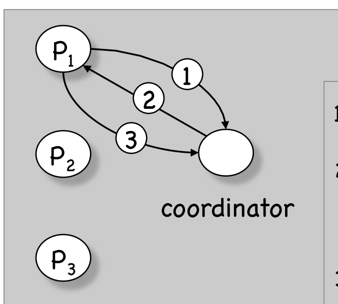
- Centralization vs. Decentralization
- Distributed Consensus
- Consensus without Identity, using a Block Chain
- Incentives and Proof of Work
- Putting it all together

Simple Example: Mutual Exclusion (*)

Recall: Mutual exclusion in shared-memory systems:

```
bool lock; /* init to FALSE */

while (TRUE) {

  while (TestAndSet(lock)) no_op;

  critical section;

  lock = FALSE;

  remainder section;

}
```

Distributed Mutual Exclusion (D.M.E.): Centralized Approach (*)

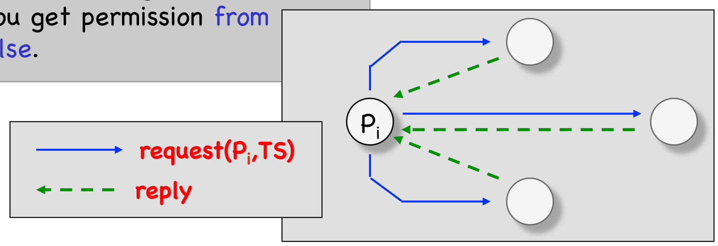


1. Send **request** message to coordinator to enter critical section (C.S.)
2. If C.S. is free, the coordinator sends a **reply** message. Otherwise it queues request and delays sending *reply* message until C.S. becomes free.
3. When leaving C.S., send a **release** message to inform coordinator.

Characteristics:
- ensures mutual exclusion
- service is fair
- small number of messages required
- fully dependent on coordinator

D.M.E.: Fully Distributed Approach (*)

Basic idea: Before entering C.S., ask and wait until you get permission from everybody else.

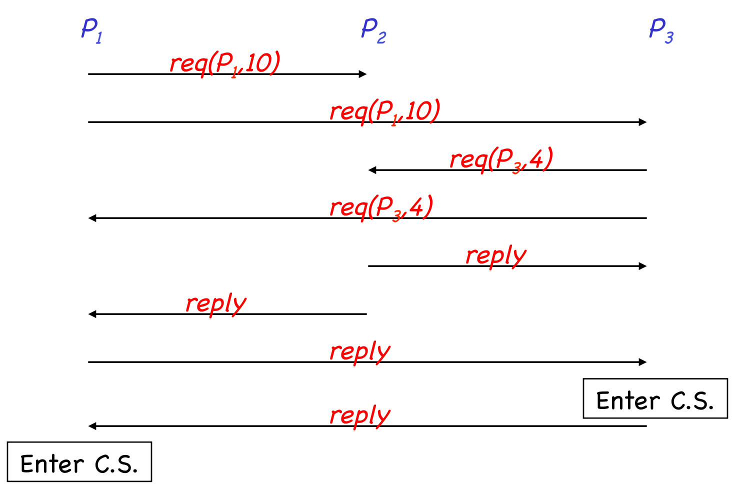


Upon receipt of a message **request(P_j, TS_j)** at node P_i:

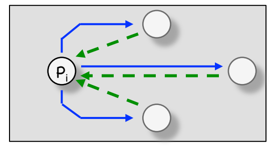
1. if P_i does not want to enter C.S., immediately send a **reply** to P_j.
2. if P_i is in C.S., defer **reply** to P_j.
3. if P_i is trying to enter C.S., compare TS_i with TS_j. If $TS_i > TS_j$ (i.e. "P_j asked first"), send **reply** to P_j; otherwise defer **reply**.

Fully Distributed Approach: Example (*)

Scenario: P_1 and P_3 want to enter C.S.



D.M.E. Fully Distributed Approach (*)



The Good:

- ensures mutual exclusion
- deadlock free
- starvation free
- number of messages per critical section: *2(n-1)*

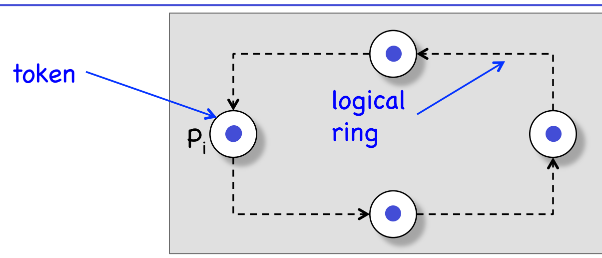
The Bad:

- The processes need to know identity of all other processes involved ("join" & "leave" protocols needed)

The Ugly:

- One failed process brings the whole scheme down!

D.M.E.: Token-Passing Approach (*)



- Token is passed from process to process (in logical ring)
- Only process owning a token can enter C.S.
- After leaving the C.S., token is forwarded

Characteristics:

- mutual exclusion guaranteed
- no starvation
- number of messages per C.S. varies

Problems:

- Process failure (new logical ring must be constructed)
- Loss of token (new token must be generated)

Just for Fun: Recovering Lost Tokens (**)

Solution: use **two** tokens!

- When one token reaches P_i, the other token has been **lost** if
  the token has not met the other token since last visit
  and
  P_i has not been visited by other token since last visit.

Algorithm:

- uses two tokens, called "ping" and "pong"

```
int nping = 1; /*invariant: nping+npong = 0 */
int npong = -1;
```

- each process keeps track of value of last token it has seen.

```
int m = 0; /* value of last token seen by Pi */
```

"Ping-Pong" Algorithm (**)

upon arrival of ("ping", nping)

```
if (m == nping) {
  /* "pong" is lost!
      generate new one. */
  nping = nping + 1;
  pong = - nping;
}
else {
  m = nping;
}
```

when tokens meet

```
nping = nping + 1;
npong = npong - 1;
```

upon arrival of ("pong", npong)

```
if (m == npong) {
  /* "ping" is lost!
      generate new one. */
  npong = npong - 1;
  ping = - npong;
}
else {
  m = npong;
}
```

How Bitcoin achieves Decentralization

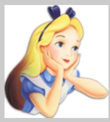
- Centralization vs. Decentralization
- Distributed Consensus
- Consensus without Identity, using a Block Chain
- Incentives and Proof of Work
- Putting it all together

Distributed Consensus

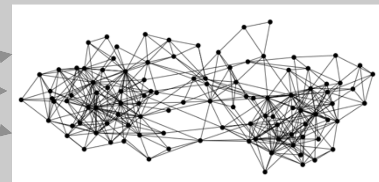
Distributed Consensus: Given *n* nodes that each have an input value. Some of these nodes are malicious. A distributed consensus protocol has the following two properties:

1. It must terminate with all honest nodes in agreement on the value.
2. The value must have been generated by an honest node.

Distributed Consensus in a Cryptocurrency



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



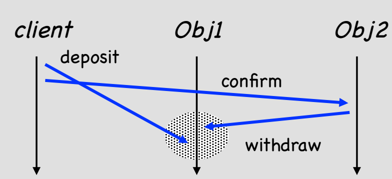
Alice broadcasts transaction to entire currency network!

The peer-to-peer nodes need consensus on:

- which transaction were broadcast
- order in which these transactions were broadcast

Consensus on Order?! (*)
(But, we don't have a global time!?)

What can go wrong if we don't agree on order (in general, not in Bitcoin):



Solution: **Timestamps**

Q: What is a Timestamp?

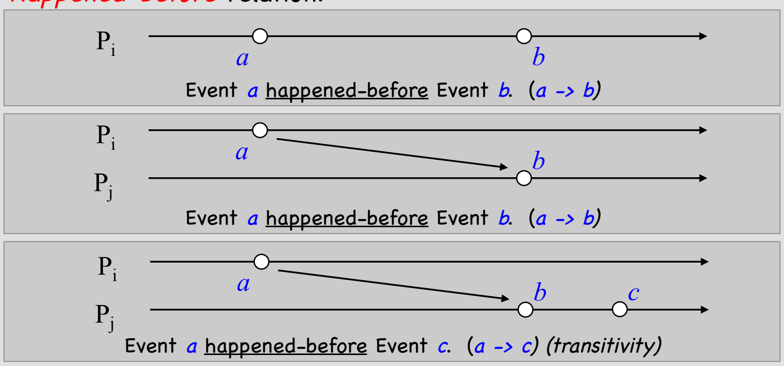
A1: A random number

A2: maybe a bit more than that . . .

Happened-Before Ordering of Events (*) (Lamport 1978)

- Absence of central time means: no notion of *happened-when* (no total ordering of events)
- But can generate a *happened-before* notion (partial ordering of events)

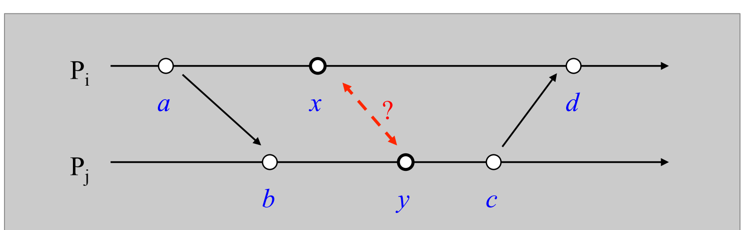
- *Happened-Before* relation:



Happened-Before Ordering (2) (*)

Q: What when no *happened-before* relation exists between two events?

A: The two events are **concurrent**.

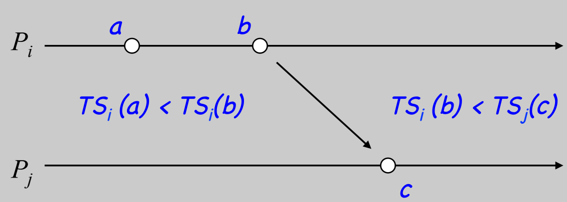


Events *x* and *y* are **concurrent**.

Happened-Before compliant Timestamps (*)

Clock Condition

if a -> b then TS(a) < TS(b)

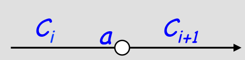


Happened-Before compliant Clocks (*)

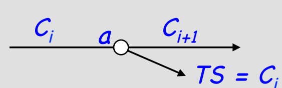
Timestamps are generated by local clocks.

Feel free to initialize local clock to some random number.

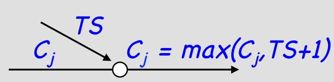
Rule 1: increment C_i after every local event.



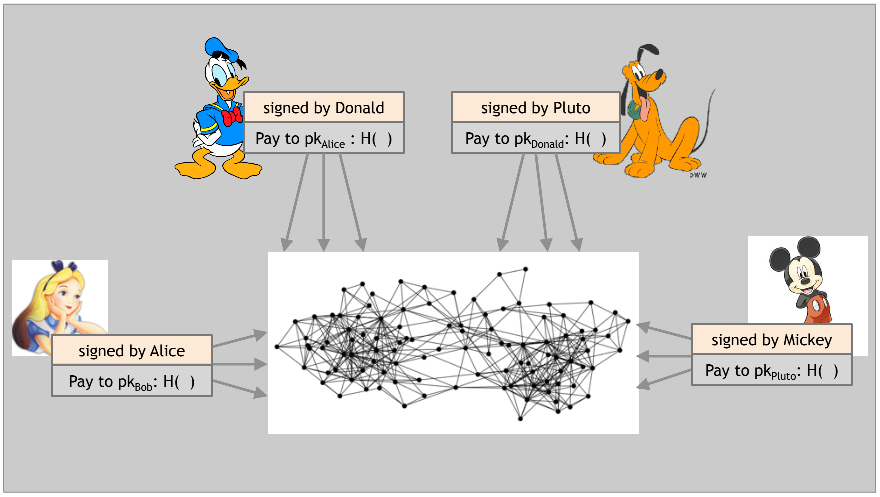
Rule 2: timestamp outgoing messages with current local clock C_i.



Rule 3: Upon receiving message with timestamp *TS*, update local clock C_j to be $C_j = max (C_j, TS+1)$



Tie back to Cryptocurrencies

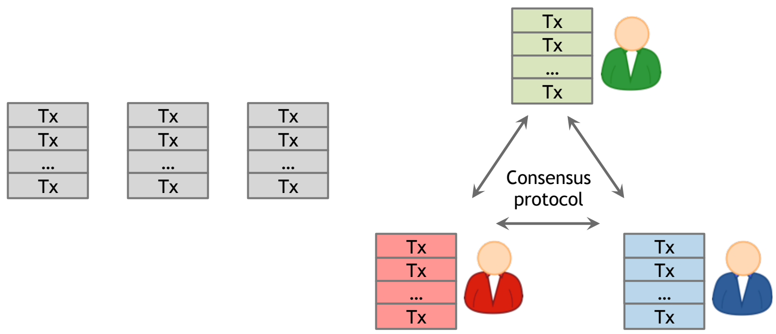


How Consensus could work in Bitcoin

At any given time:

- All nodes have a sequence of blocks of transactions they have reached consensus on
- Each node has a set of outstanding transactions it has heard about

How Consensus could work in Bitcoin



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

Consensus is hard!

Nodes may crash

Nodes may be malicious (Byzantine behaviour)

Network is imperfect

- Not all pairs of nodes connected
- Faults in network
- Latency; no global time

Bitcoin Consensus: Theory & Practice

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 only because it's a currency!

Embraces randomness

- Does away with the notion of a specific end-point
- Consensus happens over long time scales — about 1 hour

How Bitcoin achieves Decentralization

- Centralization vs. Decentralization
- Distributed Consensus
- Consensus without Identity, using a Block Chain
- Incentives and Proof of Work
- Putting it all together

Consensus without Identities

Why **identity**?

- Pragmatic: some protocols need node IDs
- Security: assume less than 50% malicious

Why don't Bitcoin nodes have identities?

- Identities are hard in P2P systems - Sybil attacks
- Pseudonymity is a goal of Bitcoin

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

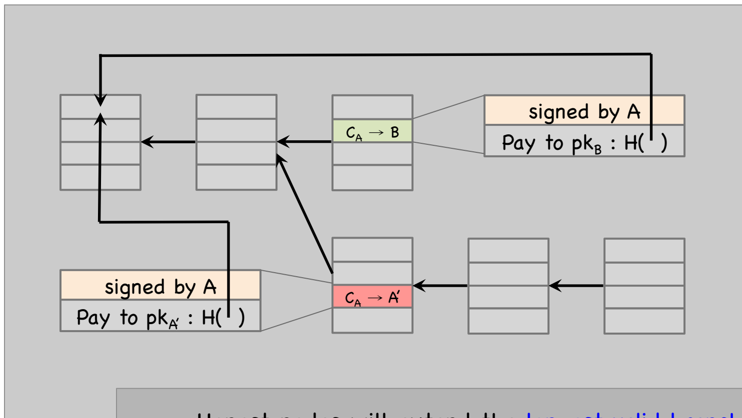
Denial-of-Service:

- 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:

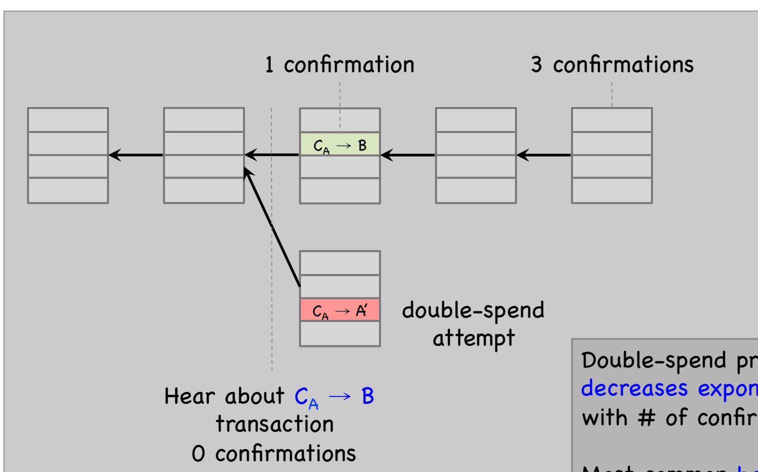
- Alice purchases service from Bob and pays in coins.
- Alice creates transaction and broadcasts it to the network.
- Later, Alice attempts to pay same coin to one of her accounts.

Double-Spend Attack



Honest nodes will extend the longest valid branch

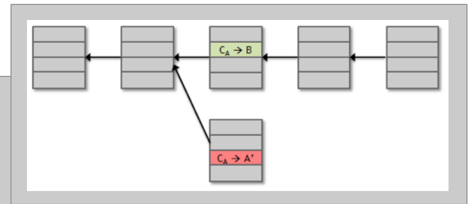
From Merchant Bob's Perspective



Double-spend probability decreases exponentially with # of confirmations.

Most common heuristic: 6 confirmations

Recap



Protection against invalid transactions is cryptographic, but enforced by consensus

Protection against double-spending is purely by consensus

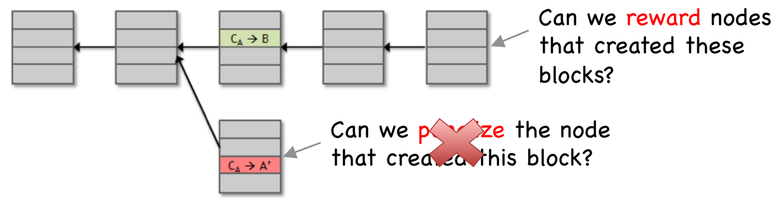
You are never 100% sure a transaction is in consensus branch. Guarantee is probabilistic.

How Bitcoin achieves Decentralization

- Centralization vs. Decentralization
- Distributed Consensus
- Consensus without Identity, using a Block Chain
- Incentives and Proof of Work
- Putting it all together

Assumption of Honesty is problematic

Q: Can we give nodes **incentives** for behaving honestly?



Everything so far is just a distributed consensus protocol.

But now we utilize the fact that the currency has value.

Two Types of Incentives

Incentive Type 1: **Block Reward**

Incentive Type 2: **Transaction Fees**

Incentive 1: Block Reward

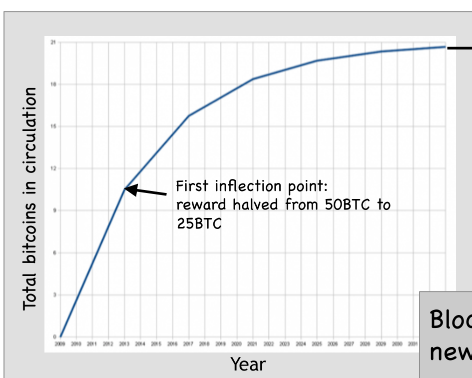
Creator of block gets to
1. include special coin-creation transaction in the block
2. choose recipient address of this transaction (typically creator)

Value is fixed: currently 25 BTC, halves every 4 years

The Catch:
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 is a finite Supply of Bitcoins



Total supply: 21 million

Block reward is how new bitcoins are created.

Runs out in 2040. 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.

Transaction fees become increasingly important, as block rewards start running out.

It is a bit unclear how this all will work out. Ongoing research!

Three Remaining Problems

1. How to pick a random node?
2. How to avoid a free-for-all due to rewards?
3. How to prevent Sybil attacks?

Selecting a Random Node: Proof of Work

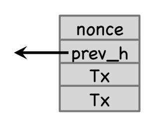
To approximate selecting a random node:

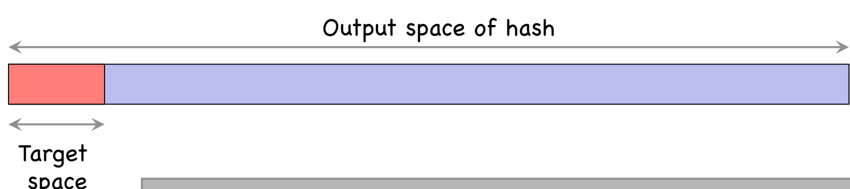
Select nodes in proportion to a resource that no one can monopolize (we hope)

- In proportion to computing power: **proof-of-work**
- In proportion to ownership: **proof-of-stake**

Proof-of-Work: Hash Puzzles

To create block, find **nonce** such that
H(nonce ‖ prev_hash ‖ tx ‖ … ‖ tx)
is very small.





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

The 3 necessary Properties of Proof-of-Work

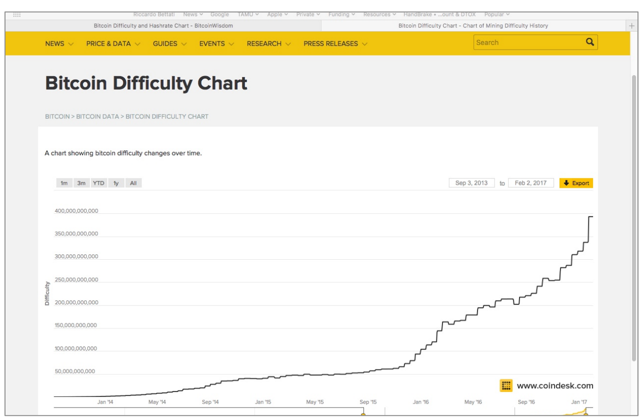
Property 1: Must be (moderately) difficult to compute

Property 2: The Cost must be "parameterizable"

Property 3: Must be trivial to verify

Property 1: Difficult to compute

It takes about *2^32 * Difficulty* to find a block.



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

Property 2: Parameterizable Cost

Nodes automatically re-calculate the target every 2016 blocks (about every two weeks).

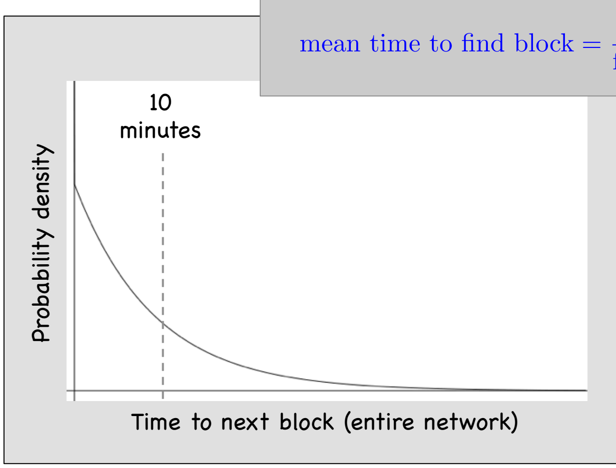
Goal: average time between blocks = 10 minutes

Adjust difficulty to meet 10-minute goal.

When will I get my Bitcoins?

For individual miner:

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



Property 3: Trivial to Verify

Nonce is published as part of block.

Other miners simply verify that

$H(nonce \parallel prev\_hash \parallel tx \parallel \ldots \parallel tx) < target$

How Bitcoin achieves Decentralization

- Centralization vs. Decentralization
- Distributed Consensus
- Consensus without Identity, using a Block Chain
- Incentives and Proof of Work
- Putting it all together

Economics of Mining

If
mining reward > mining cost
then miner makes a **profit**
where
mining reward = block reward + tx fees
mining cost = hardware cost + operating costs (electricity, cooling, etc.)

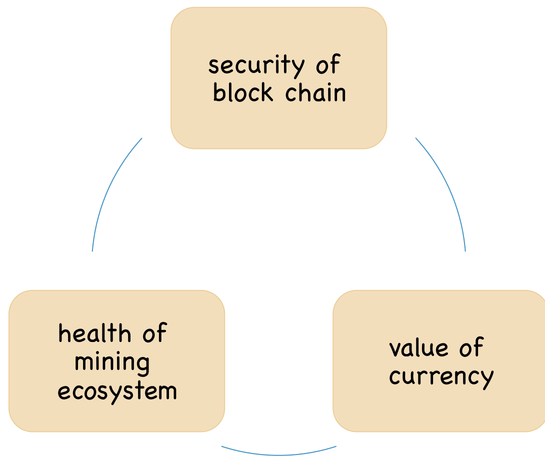
Complications:

- fixed vs. variable costs
- reward depends on global hash rate
- Cost in US$ vs. reward in Bitcoins
- Being an honest miner is not provably optimal!

We need Three Types of Consensus

1. Consensus on **Value**
2. Consensus on **State**
3. Consensus on **Rules**

Bootstrapping a Cryptocurrency



What about the "51% Attacker" Scenario?!

Steal coins from existing address? ✗

Suppress some transactions?
- From the block chain ✓
- From the P2P network ✗

Change the block reward? ✗

Destroy confidence in Bitcoin? ✓✓