RECAP

View of someone who wants to make a transaction

Wait a few blocks until you can say that the transaction is confirmed

WHY?

Want some assurance that this block will be on the longest chain in the long run!
PROOF OF WORK: RECAP

View of a miner

Block = #8ae1...
Prev = B₈
....
Txn #123 = ...
....
nonce

SHA256 ( ) = 0x0b39d9ca51f07fef3429ae15...
PROOF OF WORK: RECAP

View of a miner

$\text{Block} = \#8ae1...$

$\text{Prev} = B_8$

$\text{Sha}256 (\text{Txn } #123 = ... ) = 0x000000ef34244s1jd99a533g...$

'nonce'
PROOF OF WORK: RECAP

View of a miner

$\text{SHA256 (} \text{Block = #8ae1...} \text{Prev = B}_8 \text{... Txn #123 = ...} \text{nonce”) = 0x1104000gf4jd8011889mdk3c...}$
PROOF OF WORK: RECAP

View of a miner

Block = #8ae1...
Prev = B_8
....
Txn #123 = ...
....
nonce’’

SHA256 ( ) = 0x1104000gf4jd8011889mdk3c...

Include this transaction

| Txn=#871 | George | Anna | 1  | 1 |
PROOF OF WORK: RECAP

View of a miner

\[
\begin{align*}
B_1 & \quad B_2 & \quad \ldots & \quad B_8 \\
\text{Block} &= \#8ae1\ldots \\
\text{Prev} &= B_8 \\
\ldots & \\
\text{Txn} \#123 &= \ldots \\
\ldots & \\
\text{Txn} \#871 &= \ldots \\
\text{nonce}'''
\end{align*}
\]

\[
\text{SHA256} (\ldots) = 0x0000000aa38md69nb11efg48\ldots
\]
PROOF OF WORK: RECAP

View of a miner

Block = #8ae1...
Prev = B₈
....
Txn #123 = ...
....
Txn #871 = ...
nonce'’

SHA256 ( ) = 0x00000000aa38md69nb11efg48...
PROOF OF WORK: RECAP

View of a miner

Block = #8ae1...
Prev = B₉

SHA256 (Txn #123, ..., Txn #871, ...) = 0x00000000a38md69nb11efg48...

You lost the race
Update pointer to previous block
Remove transactions in B₉
PROOF OF WORK: RECAP

View of a miner

**Sha256 (**

<table>
<thead>
<tr>
<th>Block = #8ae1...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prev = B₉</td>
</tr>
<tr>
<td>....</td>
</tr>
<tr>
<td>....</td>
</tr>
<tr>
<td>Txn #871 = ...</td>
</tr>
<tr>
<td>nonce</td>
</tr>
</tbody>
</table>

**= 0x0000000aa38md69nb11efg48...**

Update pointer to previous block

You lost the race

Remove transactions in B₉

Keep trying!
RECAP OF BITCOIN

• **Transactions:** At any time, any buyer b can generate a transaction to pay d BTC to seller s.

• **Block:** A block consists of
  - A set of transactions
  - A cryptographic hash of the previous block (pointer to previous block)
  - An ID of the miner for this block
  - A nonce.

• A set of properly signed transactions is **valid** if no account ever overspent its limit.

• A block is valid if
  - It points to a valid block.
  - All transactions on the chain to B are valid.
  - SHA256(nonce|| info in block) has k leading zeros.
RECAP OF BITCOIN II

• **Mining**: the process of extending the blockchain from some block B.

• Longest Chain Protocol (for miners):
  ◦ Choose B to be the block furthest from the root, tie-breaking in favor of the first block you heard about.
  ◦ Include all valid transactions you’ve heard about.
  ◦ As soon as valid block created, announce it to the network.

• Miners are paid for creating valid blocks with freshly minted Bitcoins and with transaction fees.

• Difficulty of the puzzle is adjusted every 2016 blocks with the objective of making it so that a block takes 10 minutes to make in expectation.
KEY IDEA

• Trust the ledger that has the most "computational work" put into it.

• Ensure that fraudulent transactions/conflicting ledgers would require an infeasible amount of computation to create.
BITCOIN

• Is a mechanism.

• Question for us: are there beneficial deviations that can help a miner earn more than his fair share of rewards?
difficulty adjustment:
longest chain gets extended by block
on avg every 10 mins.

What can a miner do?
- choose any block they know
  about to mine on.
- deliberately fork, "disrupt tie-breaking"
- hide a block once they find it.
- include any transactions you want.

Double spending attacks

my mining power is $P$

$\lambda > 51\%$
Selfish Mining.

Mining game:

Set of miners $X_m$, facing a mining power that miner $m$ has:

$$\sum_{m} x_m = 1$$

At all times, each miner is aware of a directed tree graph $G_m$ from $m$ to root.

At each time $t$, a random miner is selected to mine a block. Miner $m$ is selected with probability $x_m$.

At each step, each miner can pick any node $v$ in $G_m$ and broadcast a path from $v$ to root.

Fix miner $m$:

Payoff to a miner is the number of blocks they've mined on the longest public chain at time $t$. 

$$T \rightarrow \text{blocks}$$
Assumptions: if miner $m$ is in a tie with any other block for longest chain, everyone else mines on $m$'s chain.

Miner $m$ will mine on longest chain, breaking ties in favor of his own blocks, but will only broadcast a block he has found if another block at same distance from root.

Reward fraction of his blocks on longest chain.

$Ta$ blocks in $T$ time steps.

Kills $Ta$ blocks on longest chain.

\[
\frac{T \alpha}{T(1-\alpha)} = \frac{2\alpha}{1-\alpha} > \alpha
\]

$\alpha = \frac{1}{3}$

$\frac{2}{3} = \frac{1}{2}$
his view

height

This implies rewards for m for any $\alpha > 3$.

Markov Chain has
Set of states $S$

State $u$, $q_{uv} = \text{Prob next state is } v | \text{ current state is } u$

$z q_{uv} = 1$

State $i$ means $h_m = h + i$

$0': h = h_m$ & there is fork

Let $p_i$ be long run (stationary) prob of being in state $i$

$P_0 = P_0(1-\alpha) + P_{0'} + P_2(1-\alpha)$

$P_0' = p_i(1-\alpha)$

$p_i = P_{0}\alpha$

$p_i = p_{i-1} + p_{i+1}(1-\alpha)$ for $i \geq 2$

Stationary distn: long run prob of being in state $i$

$\forall v \in S \sum_{u \in S} p_u q_{uv} (q_{uv})$ is yes

Long run prob $p_i$ of being in state $i$
Selfish payoff: \( \sum_{x \in S} \sum_{y \in S} q_{xy} S_{xy} \)

Honest payoff: \( \sum_{x \in S} \sum_{y \in S} q_{xy} H_{xy} \)

We will count a block when it is first announced & guaranteed to be on longest chain.

Selfish payoff = Honest payoff

Exp payoff of selfish miner with my power is assuming lose all ties is

\[
\frac{4 \alpha^2 (1-\alpha)^3 - \alpha^3}{1 - \alpha (1 + \alpha (2-\alpha))} > \alpha
\]

for when \( \alpha > 0.3 \)

\( \frac{\alpha}{1-\alpha} > \alpha \)
The Miner's Dilemma.

Mining pools used to reduce the variance.

A pool manager that joins Bitcoin as a single miner

use partial proofs of work to figure out how to split rewards among miners

The pool game:

\[ m = m_1 + m_2 \]
Pool 1 attacks Pool 2
with \( \frac{1}{3} \) of its total work.

Pool 1: \( \frac{1}{4} \)
Pold 2: \( \frac{3}{4} \) of blocks mining.

Pool 1 also gets \( \frac{1}{4} \times \frac{3}{4} = \frac{3}{16} \).

Pool 1 makes \( \frac{1}{3} + \frac{1}{3} + \frac{3}{16} = \frac{5}{9} \).

NE as long as both pools < 80%.

<table>
<thead>
<tr>
<th>Pool 1 doesn't attack</th>
<th>Pool 1 does attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pool 2 doesn't attack</td>
<td>((a, b))</td>
</tr>
<tr>
<td>Pool 2 does attack</td>
<td>((a - \epsilon, b + 2\epsilon))</td>
</tr>
<tr>
<td>Pool 1 doesn't attack</td>
<td>((a - 2\epsilon, b + 2\epsilon))</td>
</tr>
<tr>
<td>Pool 1 does attack</td>
<td>((a - \epsilon, b - \epsilon))</td>
</tr>
</tbody>
</table>