# Nakamoto Consensus Continued: GHOST and Selfish Mining

Kai Mast

CS639/839

Spring 2023

# ANNOUNCEMENTS

- Submit project proposals by 3/26
  - Can be short
  - Can be earlier!

- Next week's lecture will be on Zoom

# TODAY'S AGENDA

**Part 1:** Nakamoto Consensus Continued

- A deeper look at difficulty adjustments
- The GHOST protocol

**Break**

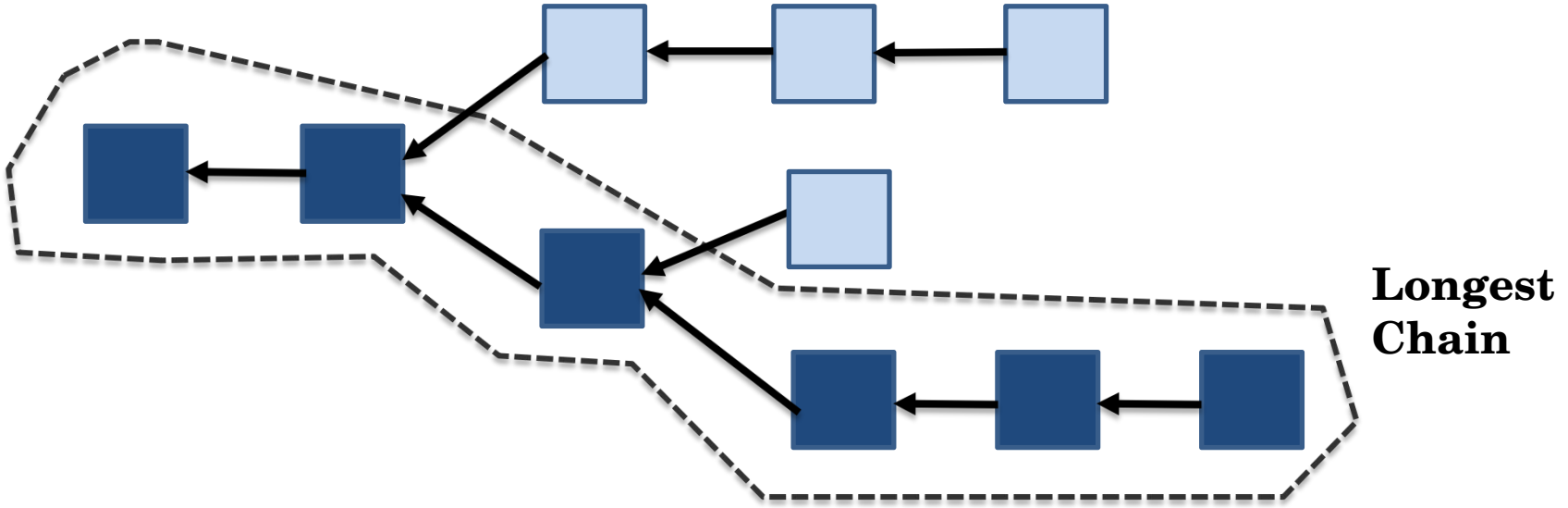
**Part 2:** Selfish Mining

- Problems with the original Bitcoin protocol

# RECAP: GOSSIP PROTOCOLS

- Each node is connected to a small set of peers (=other nodes)
- Nodes forward messages to their peers when they receive them
- Malicious nodes can delay or not forward any message
  - However, for each pair of nodes multiple paths of communications exist (usually)
  - Attackers can only efficiently withhold their own messages

# RECAP: NAKAMOTO CONSENSUS



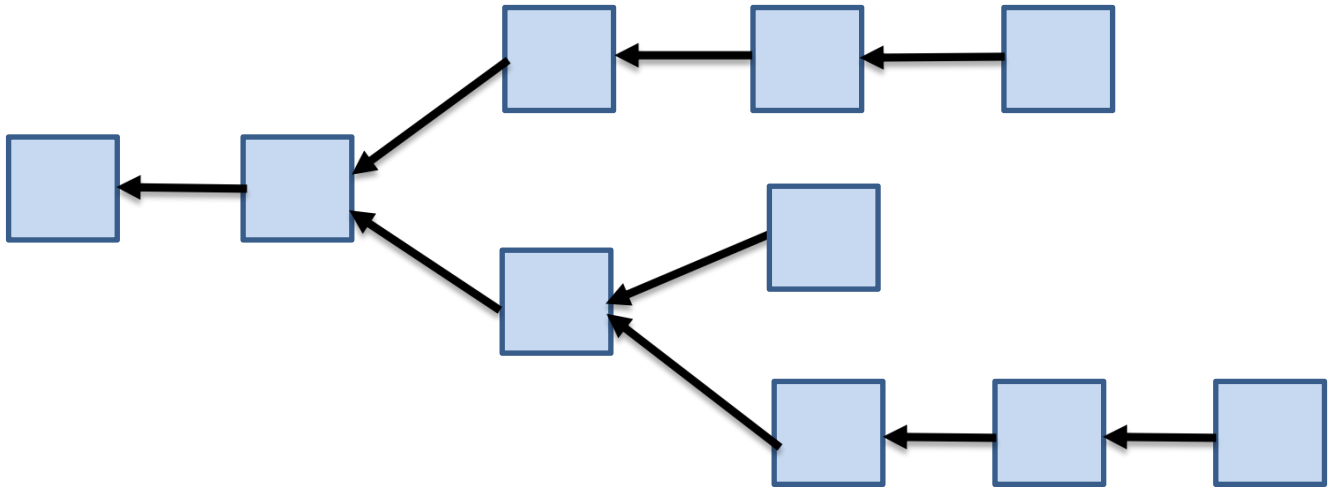

**Component 1:** (Pseudo-)random Block Generation

- No predetermined entity generates blocks, but virtually anyone can
- Multiple blocks can be created at the same time

**Component 2:** Longest Chain Rule

- Correct nodes will always extend the longest chain when creating a new block
- When there are multiple longest chains, pick one at random

# RECAP: THE BLOCKCHAIN

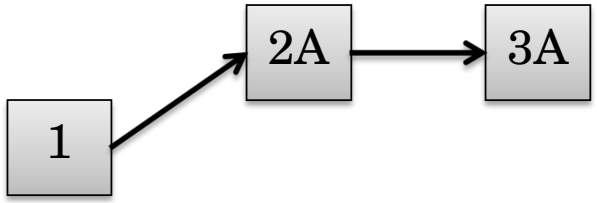


**Purpose 1:** Store transaction data and determine transaction ordering

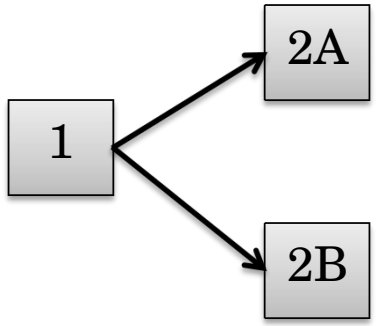
**Purpose 2:** Track agreement on which transactions are accepted

- Blocks *confirm* their ancestors

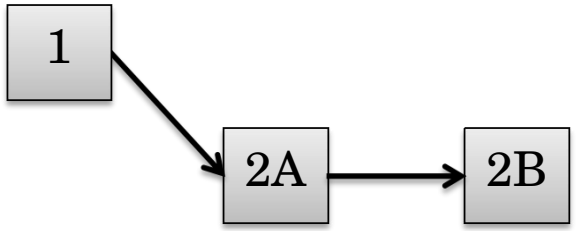
# INFORMATION ASYMMETRY




**Node 1's View**




**Node 2's View**




**Node 3's View**

- Each node sees some subset of all blocks
- In Bitcoin and Ethereum 1.0 there is no certain way of knowing which blocks have been seen by a majority of nodes

**Why?**

- Network failures and delays
- Attackers might not forward blocks

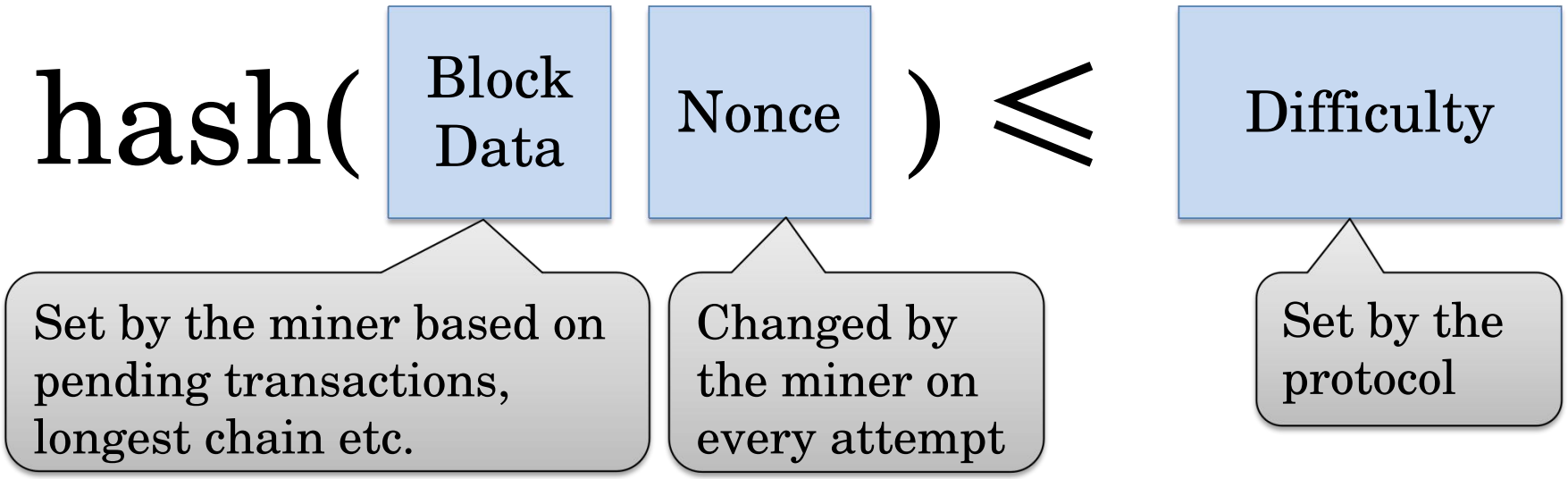
# RECAP: PROOF-OF-WORK

**Goal:** Tie likelihood of generating a block to processing power

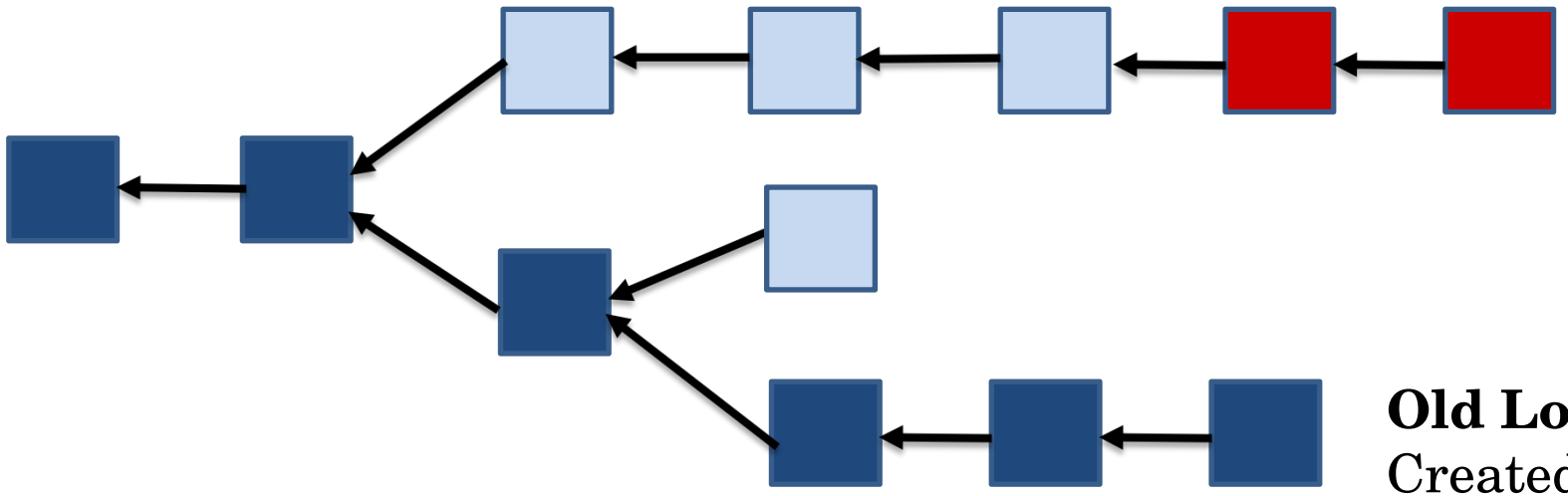
- Each node only has some finite amount of hardware

**Approach:** Create a very hard-to-solve task (the "crypto puzzle")

- Random tries are needed to find the solution
- We might need many attempts to solve it



# CHAIN SAFETY



**New Longest Chain**
Created by Attacker

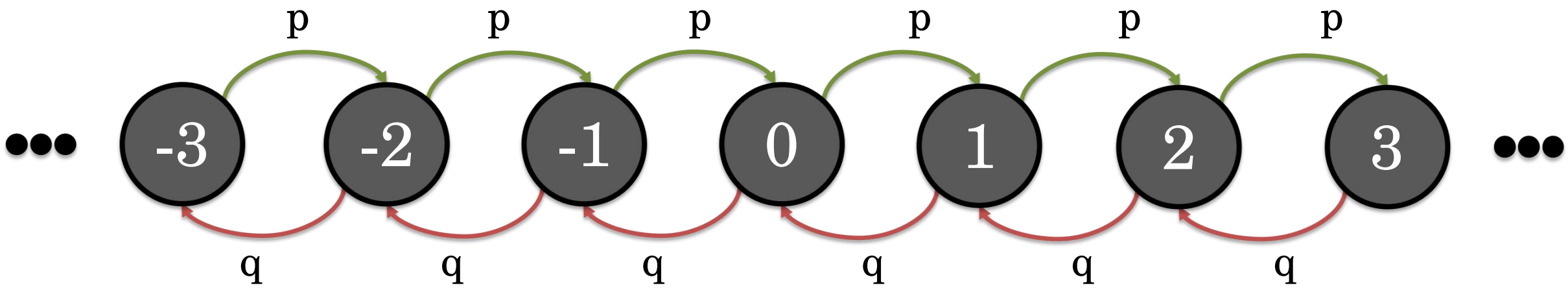
**Old Longest Chain**
Created by Honest Miners

Attackers might intentionally build a fork to *overtake the honest chain*

**Worst Case:** One strong coordinated attacker

**Assumption:** Attacker's mining power <50%

# ADVANTAGE OF HONEST CHAIN




**Honest Mining Power:** p

**Attacker's Mining Power:** q = 1 - p

Chance the miner overtakes after z blocks
**asymptotically approaches 0**

# RECAP: DIFFICULTY ADJUSTMENT

**Goal:** Ensure that a blocks are created at the same frequency

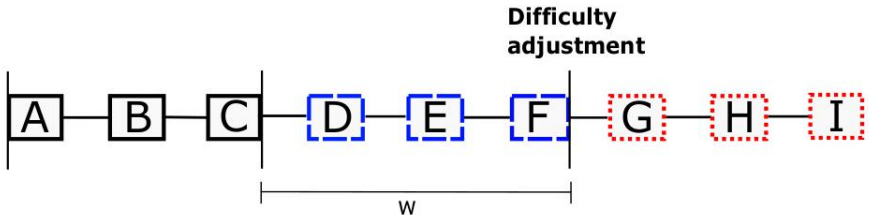
- e.g., every 10 Minutes in Bitcoin

**Challenge:** Mining power can change over time

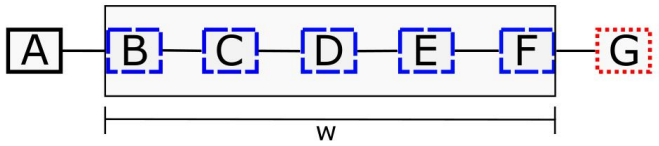
- Miners can join or leave at any time
- Miners might start (or stop) mining if it is (not) economical to so
  - Depends on electricity, cryptocurrency, and hardware prices
- Miners can switch between networks (e.g., from Bitcoin to Dogecoin)

Adjust difficulty based on the *observed frequency vs. the expected frequency*

# DIFFICULTY ADJUSTMENT MECHANISMS








**Period-Based**

- Every w blocks the difficulty will be adjusted
- For example, in Bitcoin every 2016 blocks (roughly 2 weeks) the difficulty is recalculated

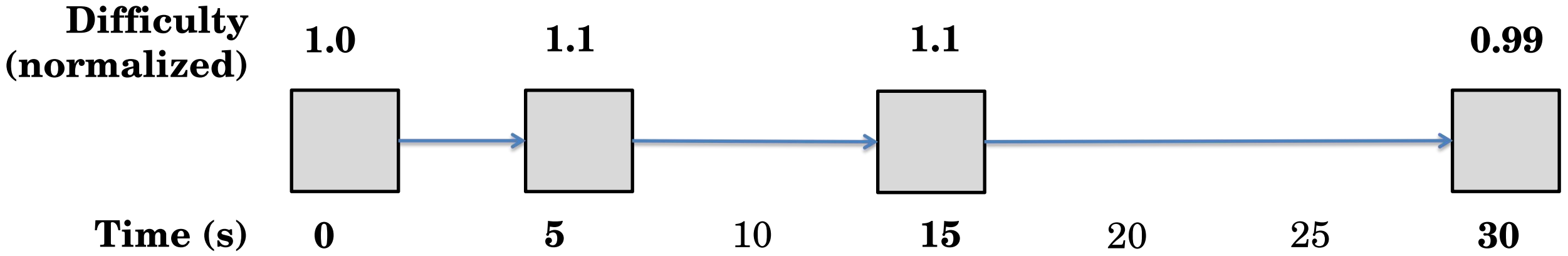
**Incrementally Extrapolated**

- Every block the difficulty will be adjusted slightly depending on how long it took to mine it
- Difficulty is only adjusted, not recalculated
- Used by Ethereum

**Sliding Window**

- Every block the difficulty will be adjusted depending on how long it took to mine the last w blocks
- Used by Monero and Bitcoin Cash

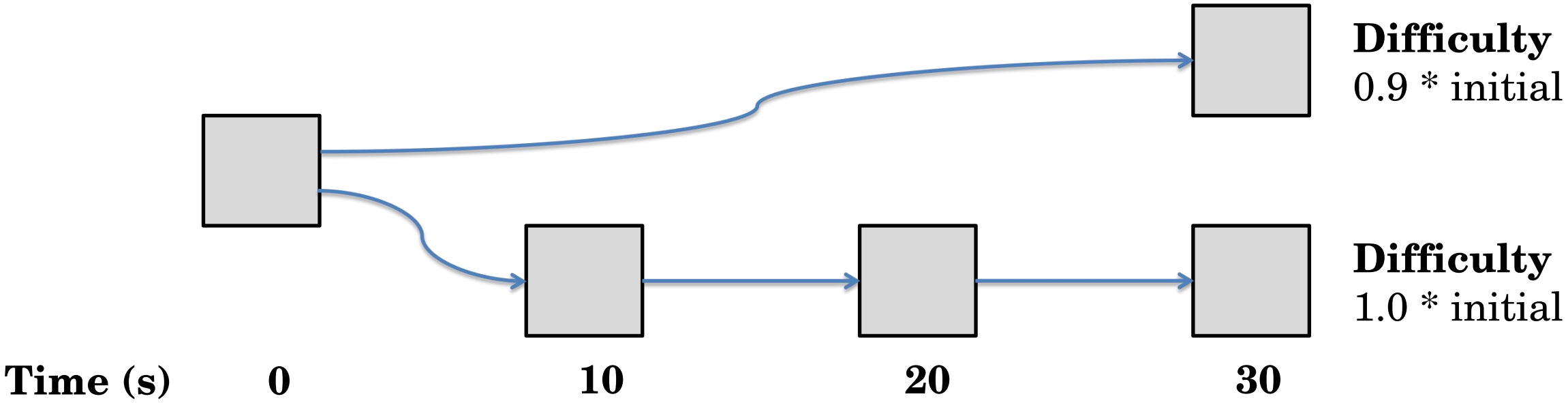
# DIFFICULTY ADJUSTMENT EXAMPLE




**Simplified Incremental Extrapolation**

```
delta_t = block.time - block.get_parent().time
expected = 12 # Ideally it should take 12 seconds
if delta_t < expected-2: # Allow for some error
    difficulty *= 1.1 # Increase difficulty
elif delta_t > expected+2:
    difficulty *= 0.9 # Decrease difficulty
```

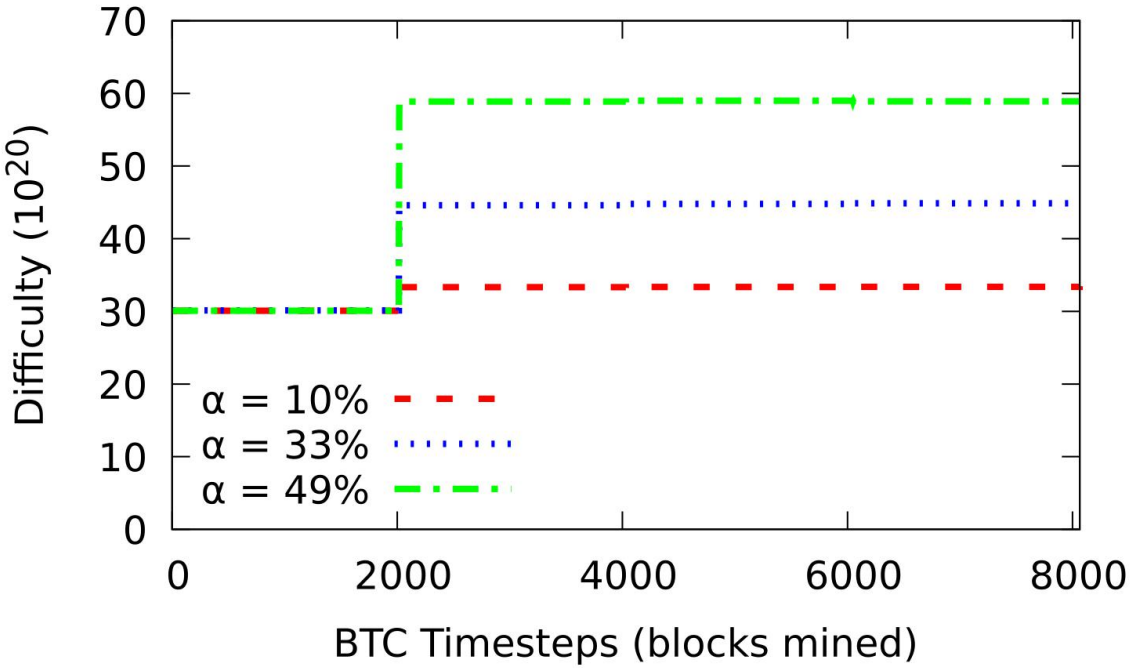
# DIFFICULTY ADJUSTMENT AND FORKS



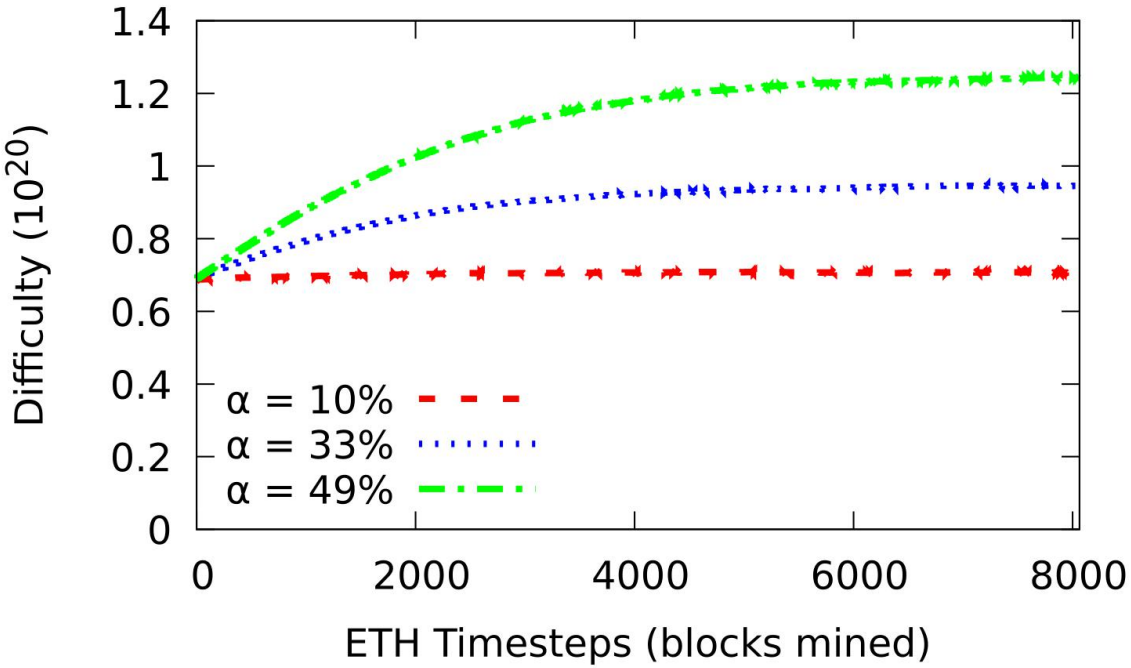

- Difficulty is adjusted based on direct ancestors, not forks
- Different, competing, forks do not need to have the same difficulty
- Difficulty change always applies to the next block

# EFFECTS OF DIFFICULTY ADJUSTMENT

A new powerful miner joins the network (at T=0). The new miner has α of the total mining power.
How long does it take to adjust difficulty?




Bitcoin adjusts difficulty every 2016 blocks




Ethereum adjusts difficulty incrementally

# INCENTIVES FOR MINERS

Majority of miners (or mining power) is “honest”

- Will follow protocol as long as it makes financial sense
- Mining is very costly

We need a mechanism to make mining profitable

- Block rewards and transaction fees are paid to the miner
- Rewards are only paid out if block is on the winning chain
  - Ensures convergence to a longest chain

Lower difficulty means, potentially, higher rewards for miners!

# DETERMINING THE WINNING CHAIN

**So far:** Longest Chain

- The chain with highest number of block wins

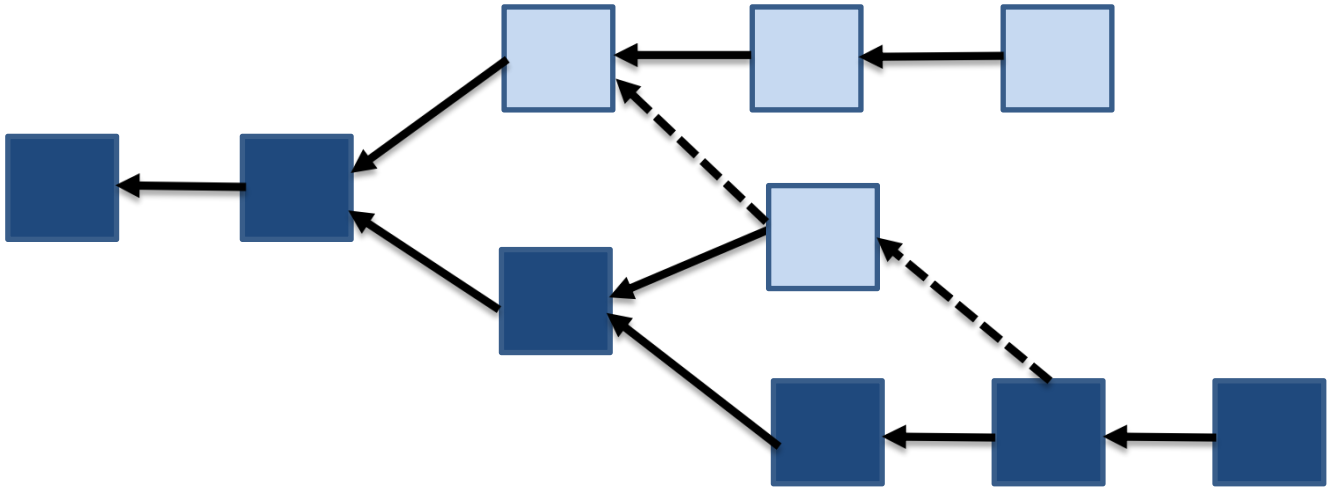
**Problem:** Does not work with difficult adjustment

- Majority of miners could work on chain with less blocks but higher difficulty

**Alternate Approach:** Heaviest Chain

- Sum up difficulty values of all blocks to determine chain weight
- Used in most PoW chains

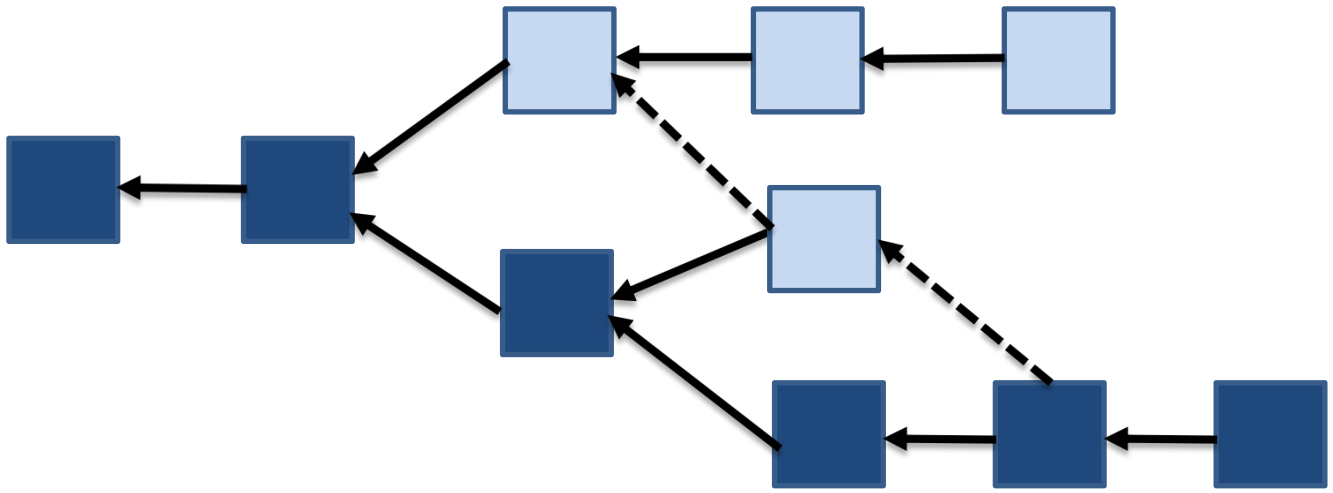
# GREEDY HEAVIEST OBSERVED SUBTREE



**Idea:** Allow frequent forks without reducing the safety of the chain

- Blocks do not only reference their direct parent, but also *uncle* blocks
- Orphan blocks' transactions are not applied to the blockchain state, but their mining power still counts towards the "heaviest" chain

# UNCLE BLOCKS

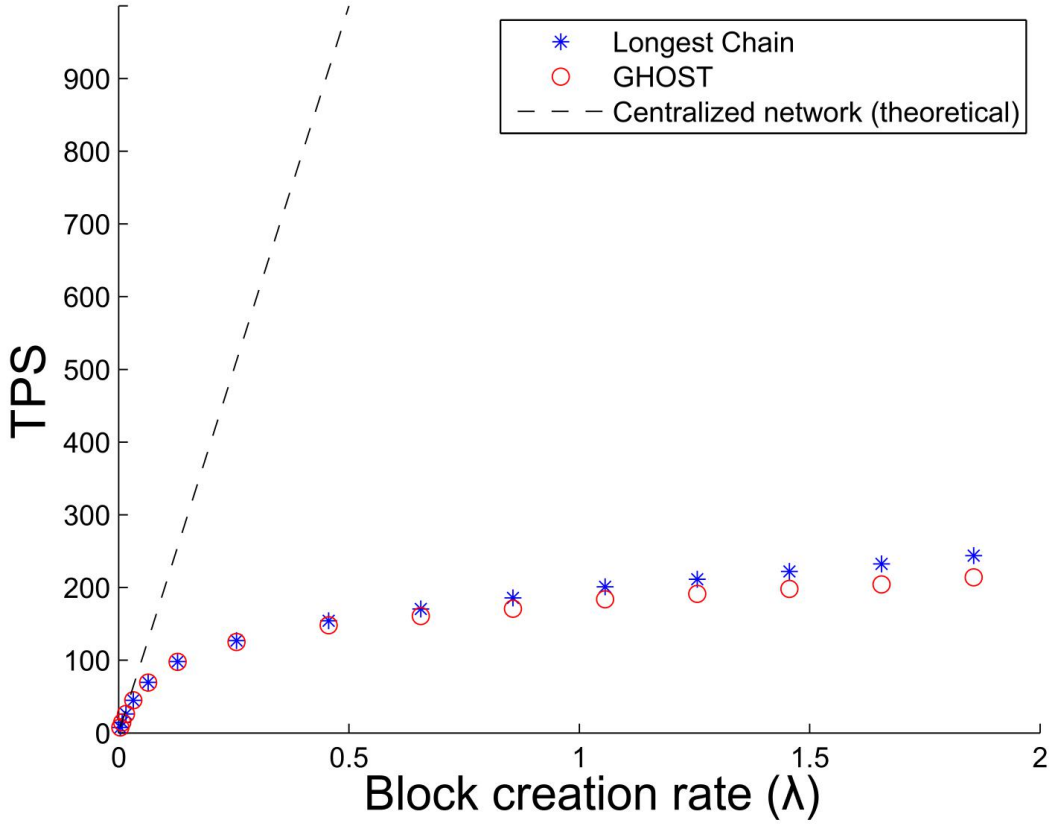


In Ethereum, an uncle is any block that is

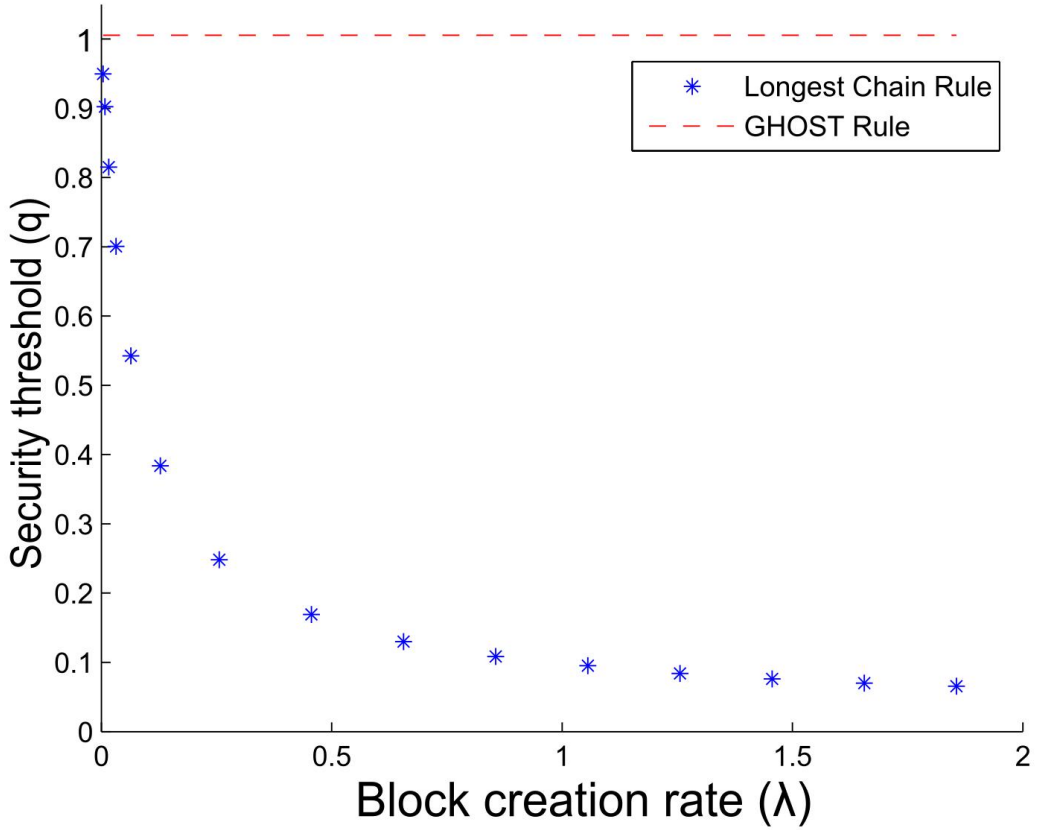
- A child of a direct ancestor
- Not an direct ancestor itself
- Not already marked as an uncle of a direct ancestor

In Ethereum, uncle blocks get some of the block reward (but not transaction fees)

# GHOST: FORMAL RESULTS




Throughput does increase with higher block frequency




Security does not degrade with higher block frequency

# BREAK

# SELFISH MINING

- Demonstrates that the economic incentives in the original Bitcoin paper are not sound
- Published in 2013 by Ittay Eyal and Emin Gün Sirer
- Later, "re-examined" by Kevin Negy, Peter Rizun, and Emin Gün Sirer

Ittay Eyal (Technion and IC3)

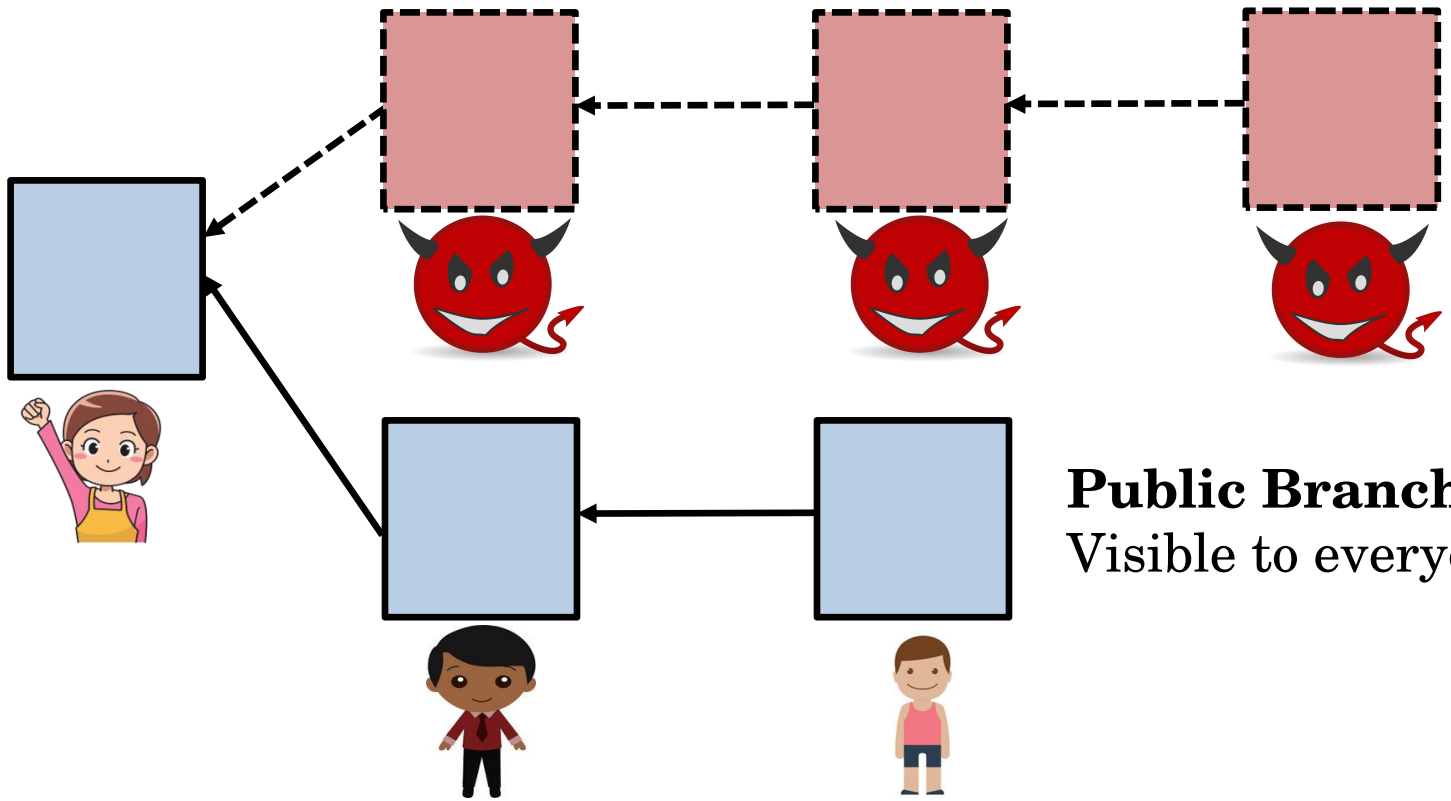
Kevin Negy (Cornell and IC3)

Emin Gün Sirer (Cornell, Ava Labs, and IC3)

Peter Rizun (Bitcoin Unlimited)

# SELFISH MINING

**Idea:** Selectively hide blocks you mine to gain an advantage



**Private Branch**
Only visible to selfish miner(s)

**Public Branch**
Visible to everyone

# SELFISH-MINE STRATEGY: INTUITION

**Goal 1:** Keep mined blocks hidden as long as possible

- Use "information asymmetry" to your advantage
- Other miners cannot mine on hidden blocks

**Goal 2:** Try to have two competing branches exist for as long as possible

- If there are multiple competing branches, honest mining power might be split between them

**Approach:** When honest miners publish a block, try to reveal one (or more) blocks from the private branch to "confuse" honest majority

# WHEN TO PUBLISH A BLOCK

**When** selfish miner finds a block
**and** private branch is now longer than public branch
**and** private branch is now exactly two blocks
**then** publish entire private branch *(selfish miner wins)*

**When** honest miner finds a block **and** public and private branch now have the *same length*
**then** publish one block *(confuse honest miners)*

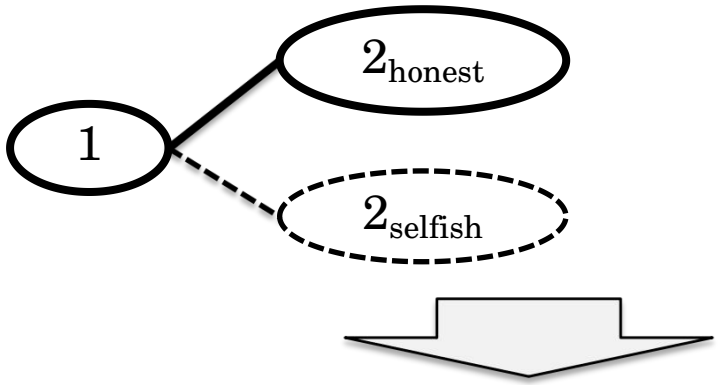
**When** honest miner finds a block **and** private branch is *exactly two blocks longer*
**then** publish entire private chain *(selfish miner wins)*

**When** honest miner finds a block **and** private branch is still *more than two blocks longer*
**then** publish one block *(confuse honest miners)*

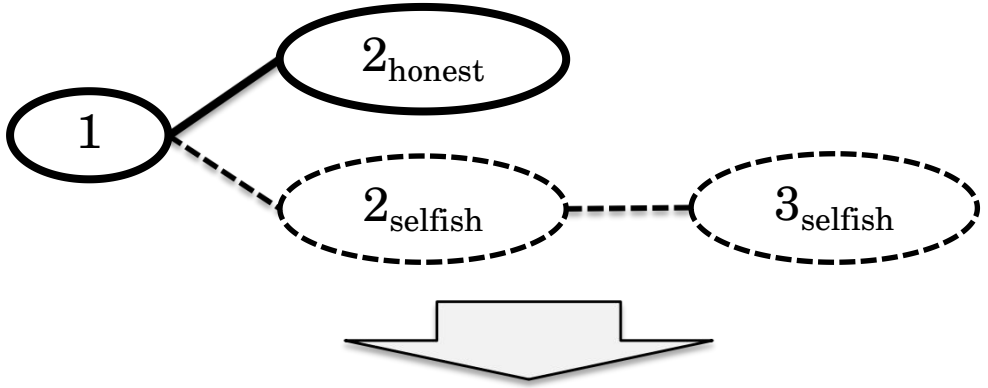
**When** honest miner finds a block **and** *public branch is longer* than private branch
**then** switch to mining on the public branch *(honest miners win)*

# SELFISH MINING: EXAMPLE 1

**1)** Private and public branch are both of length one

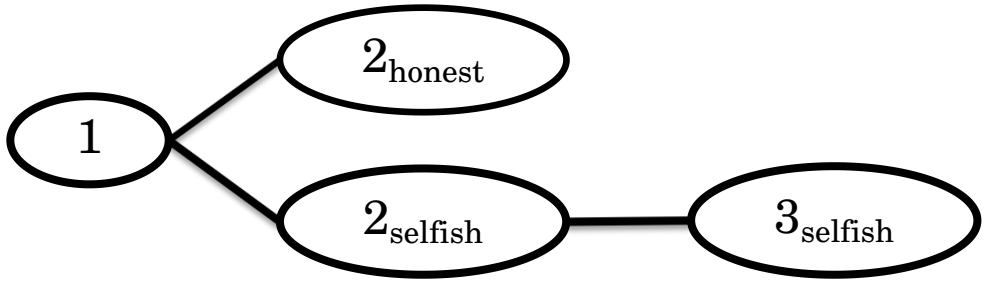



**2)** Selfish miner finds a new block




**3)** Selfish miner reveals both of its blocks and wins

($2_{honest}$ becomes orphaned)

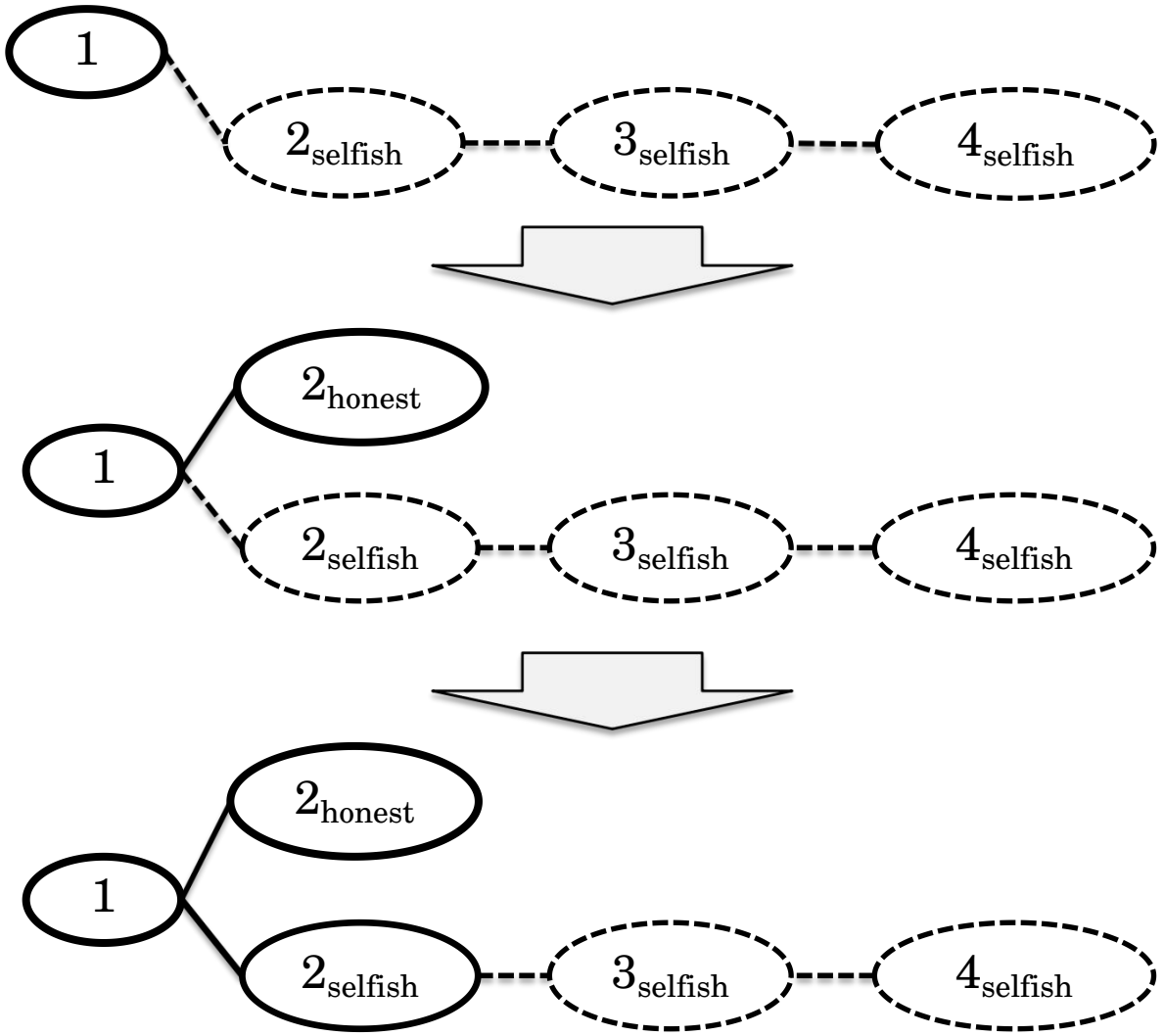


# SELFISH MINING: EXAMPLE 2

**1)** Private branch consist of three hidden blocks

**2)** Honest miners publish a block

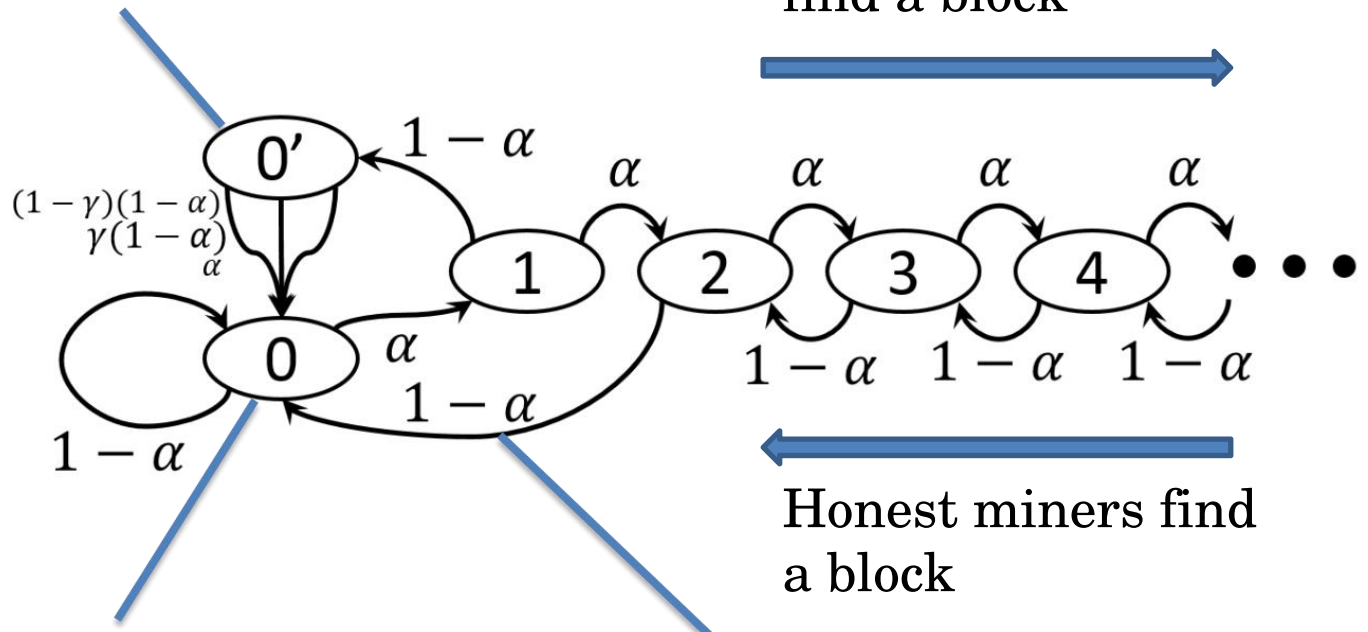
**3)** Selfish miner reveals one of their blocks so there are two competing public branches



# FORMAL ANALYSIS

There are two competing branches

Selfish miners find a block




Honest miners find a block

There is only a single (public) branch

Selfish miner reveals entire private branch and "wins"

- α is the fraction of the total mining power that belongs to the selfish miner
- γ is the fraction of honest miners that pick the selfish branch to mine on

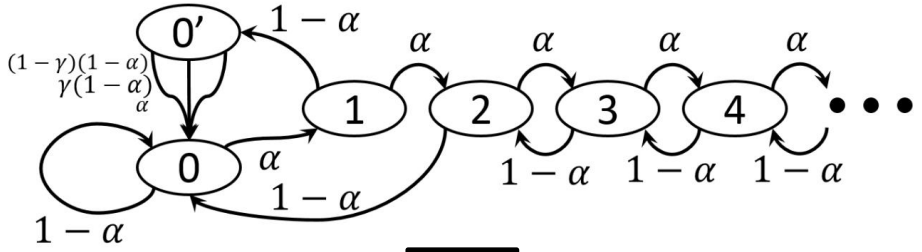
You do not need to know this for the midterm/final

# SELFISH-MINE REVENUE

Use state probabilities to calculate mining revenue

For the strategy to be successful, the fraction of revenue must be greater or equal to the pools mining powerα

Success of selfish mining strategy depends on how many honest miners choose to mine on the selfish branch

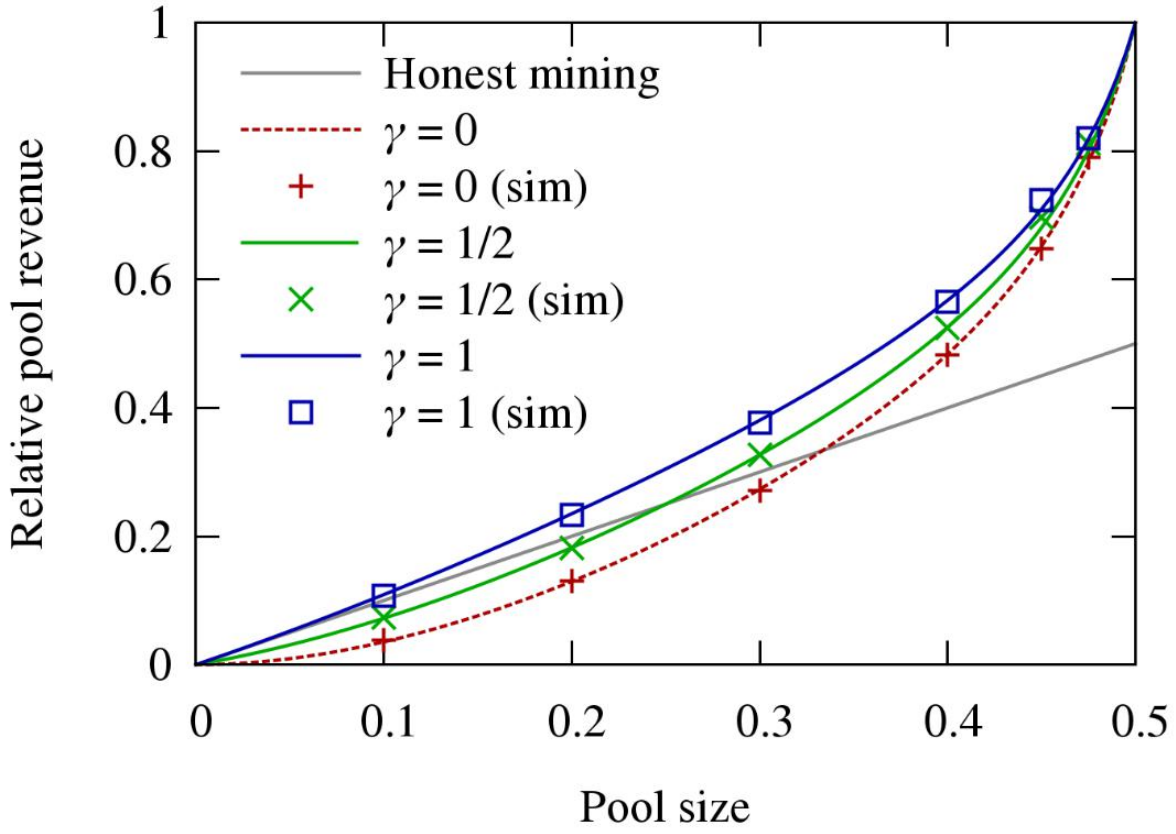



$$R_{\text{pool}} = \frac{r_{\text{pool}}}{r_{\text{pool}} + r_{\text{others}}} = \cdots = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)}$$

$$\frac{1-\gamma}{3-2\gamma} < \alpha < \frac{1}{2}$$

You do not need to know this for the midterm/final

# SELFISH-MINE REVENUE




- The paper presents both theoretical and simulated results
- γ is the fraction of honest miners that choose to mine on the selfish chain
- If all miners pick the selfish chain, selfish-mining is always profitable!

# "FIXING" THE BITCOIN PROTOCOL

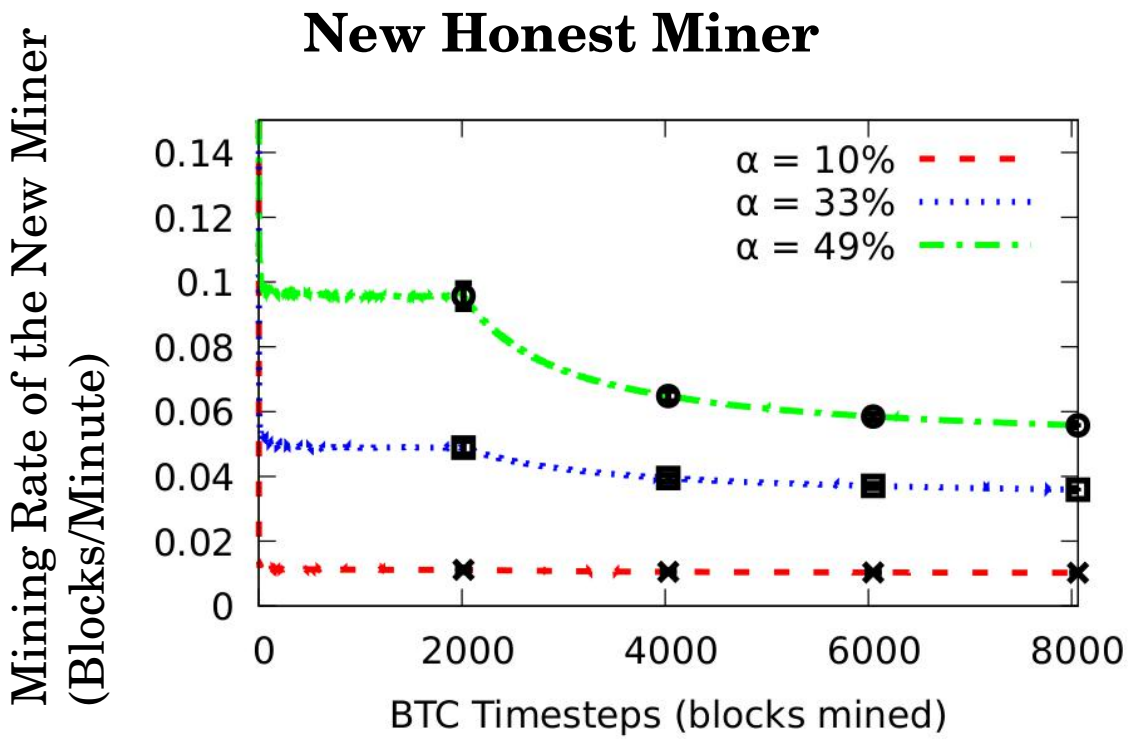
- If there are two (or more) competing chains, pick which one you mine on at random ($\gamma = 0.5$)
  - Simple modification that does not require a hard fork
- Without this modification Bitcoin is vulnerable against any fraction of selfish miners
  - With the modification the threshold is still 1/3, not 1/2

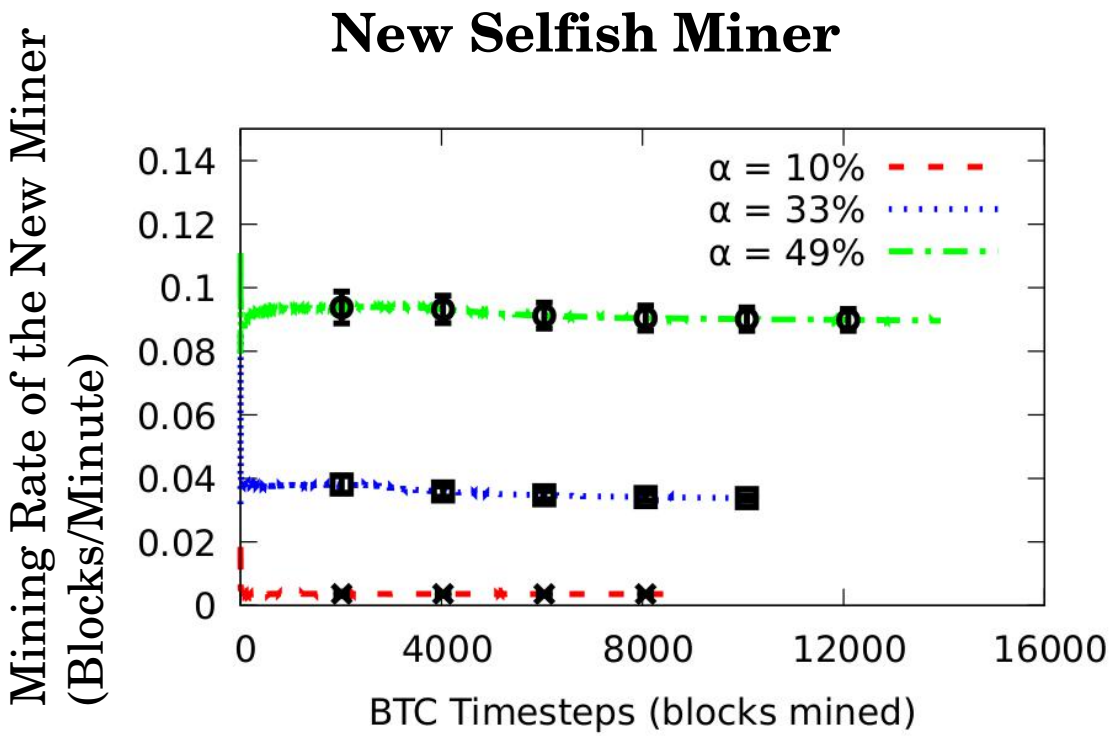
# CRITICISM OF SELFISH MINING

1. While selfish miners may win more blocks than expected, overall revenue could still be lower because of high orphan rate
   - If more blocks are orphaned, less blocks receive block rewards
   - Difficulty adjustment will resolve this somewhat

2. Original analysis does not fully take difficulty adjustment into account
   - Some critics say selfish mining must be maintained for weeks before it becomes profitable

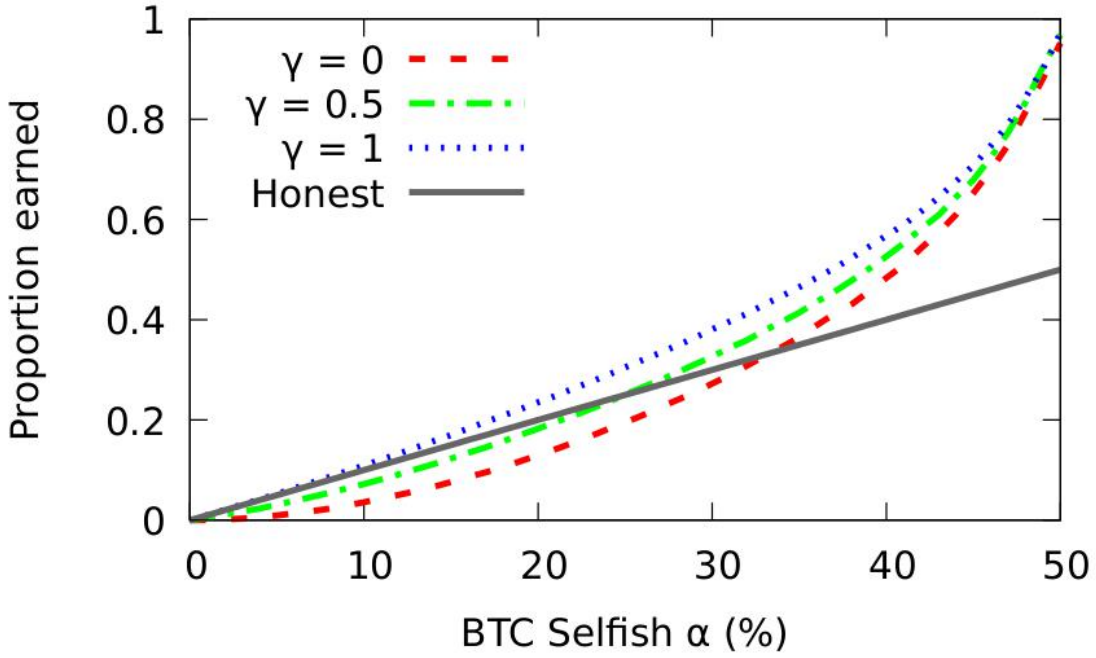
# EFFECTS OF DIFFICULTY ADJUSTMENT

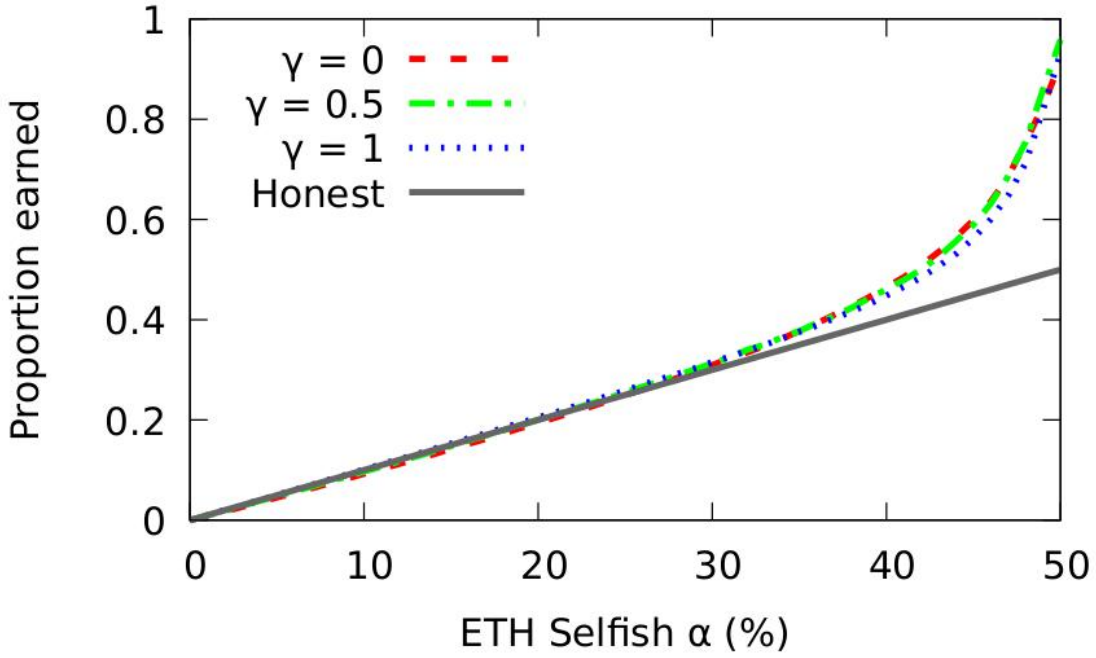



**Difficulty does not adjust (much) with selfish mining as rate of orphan blocks is higher!**

# SELFISH MINING AND GHOST

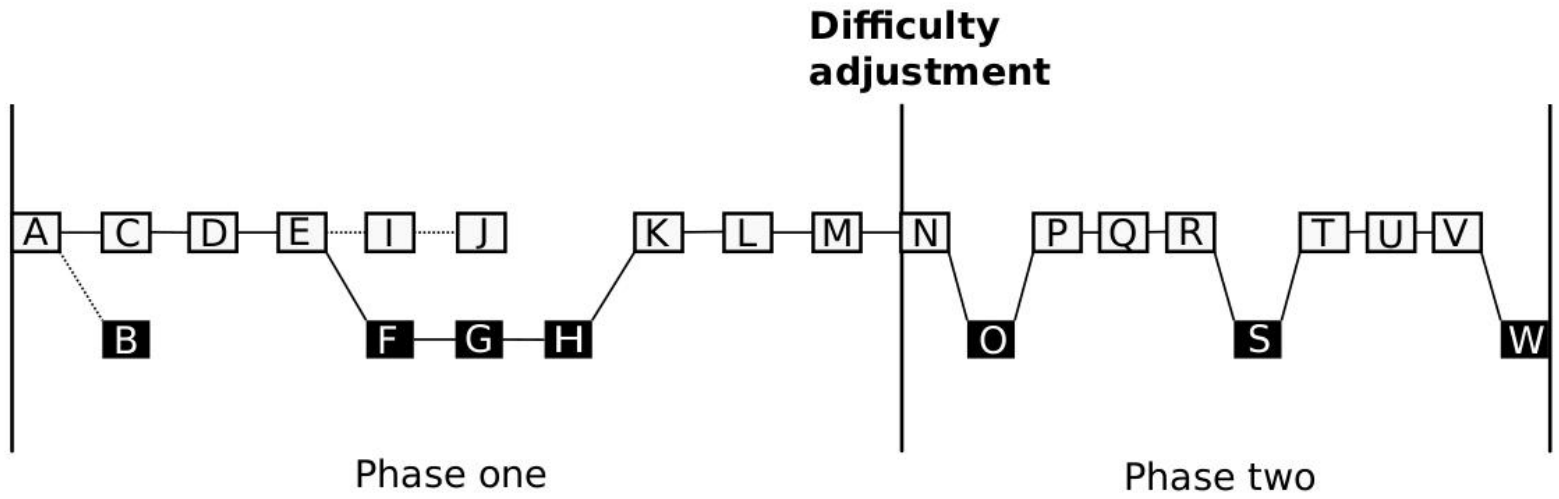



**GHOST almost nullifies the downsides of selfish mining, because even orphan blocks get some reward!**

# INTERMITTENT SELFISH-MINING (ISM)

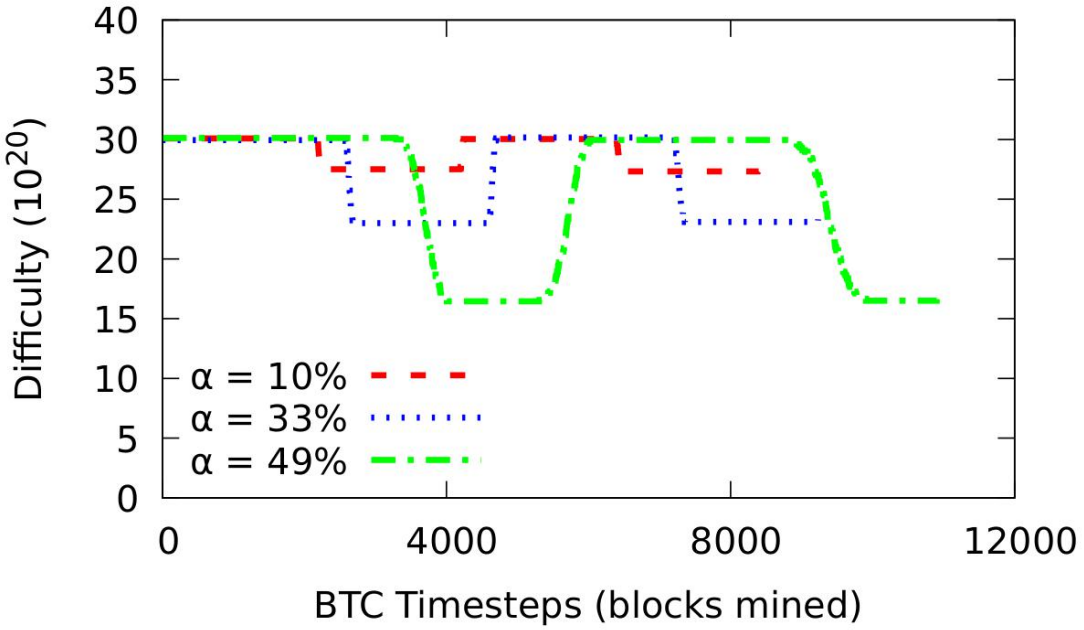
**Idea:** Leverage change in mining difficulty to increase selfish-mine revenue
**Approach:** Alternate between selfish and honest mining



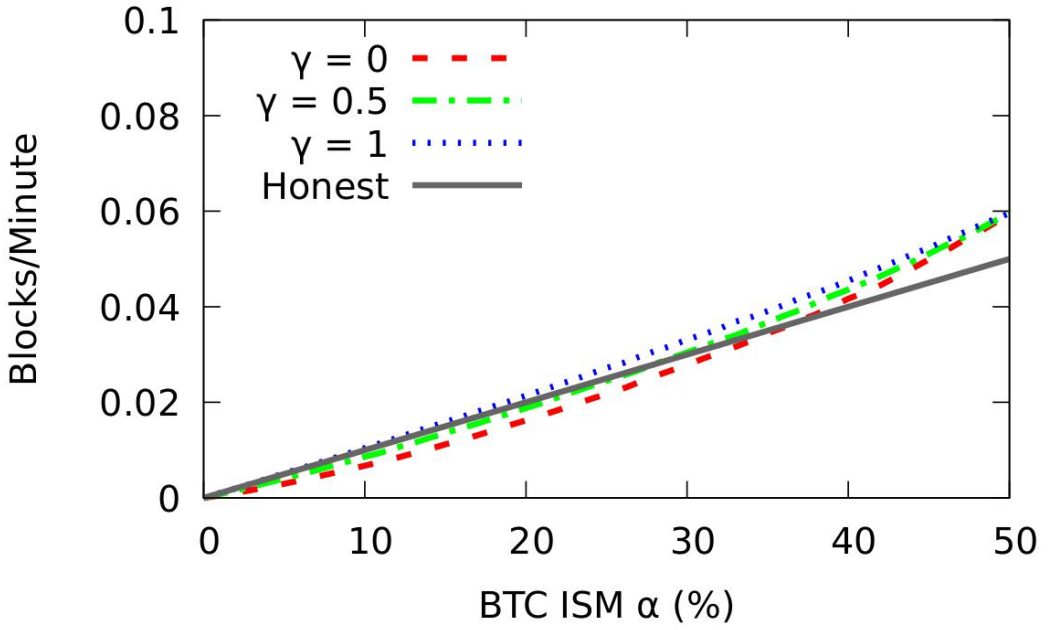

Follow selfish mining strategy
(to win more blocks than expected)

Follow honest mining strategy
(to maximize revenue)

# INTERMITTENT SELFISH-MINING IN ACTION




An intermittent selfish miner causes the difficulty to lower after selfish phases and rise after honest phases (γ = 0)




Block win rate of the intermittent selfish miner after two periods (2*2016 blocks)

- "Win rate" refers to the blocks that are mined and do not become orphans
- Revenue is lower than before but still higher than that of an honest miner

# THAT'S ALL FOR TODAY

## Conclusions

- For Blockchain systems, we always need to take economic incentives into account
- Do not blindly believe claims of a paper, especially if it is not peer-reviewed