Decentralized Systems

Bitcoin Security

Bitcoin Security

- Bitcoin has the ambition of being **secure**, **decentralized** and **trustless**
  - i.e., the system is secure even if everybody is selfish and ill-intentioned
- To what extent it really is?

Mining Pools: Just a Few More Details

Mining Pools Are Centralized

- A **coordinator** prepares **block templates**
  - They choose the included transactions
  - The template pays block rewards to the coordinator
- Pools have an **internal difficulty** that is much lower than the Bitcoin one
  - When you solve a block with the internal difficulty, you get a "**share**" that allows you to share profits
  - When you solve a **real block**, the profits will be distributed among all those that have a share (and a fee for the coordinator)
- The coordinator needs to be **trusted**

51% Attack

51% Attack

- Bitcoin is considered secure **only if more than half of the hashing power is composed by "honest" nodes**
  - "Honest": nodes behaving according to the protocol
- Idea here: if there is a malicious fork, the "honest" one will finally win because they will build more blocks
- What if, however, attackers get to **more than that**?
- Source: How A Mining Monopoly Can Attack Bitcoin, blogpost by Eyal and Gün Sirer

Double Spending

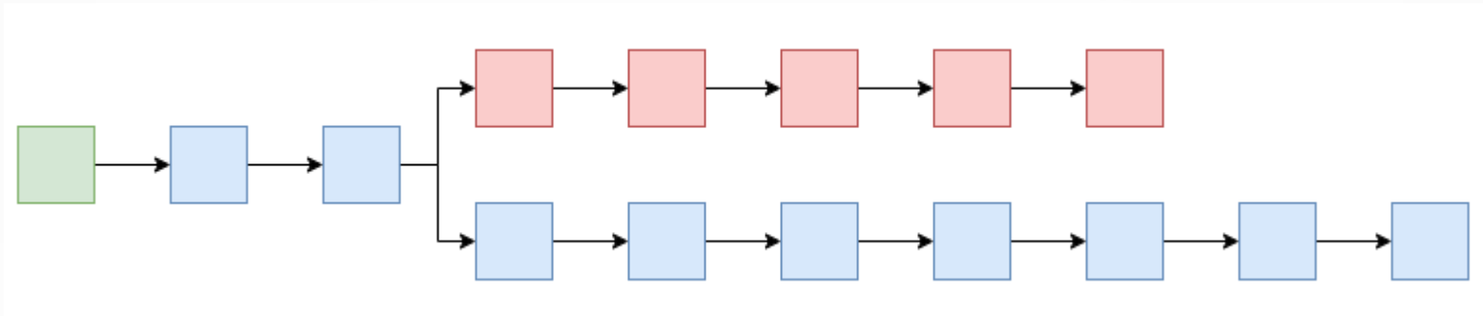


image from chowles.com

- A 51% attacker can always create a fork and eventually get a longer chain than the previous one
- They can spend all their bitcoins in the upper chain and then work on a fork where they’re not spent

Transaction Censorship

- A 51% attacker can also “censor” transactions they don’t like—simply not work on top of blocks that contain them
  - Those “orphan” blocks will eventually be in a shorter fork, and hence be censored
- This can take many forms
  - Total denial of service
  - Extortion (“pay me a huge fee otherwise you’ll never get to use your bitcoins”)
  - Targeting selected miners, transactions and addresses

Loss of Decentralized Narrative

- A 51% miner may not launch any attack
  - This will likely make BTC value fall, hence they might lose
  - However, many people choose cryptocurrencies **because they don’t want to trust anybody**
  - Does it work when **you have to trust somebody**?

Are 51% Attacks Feasible? (1)

Bitcoin Gold Blockchain Hit by 51% Attack Leading to $70K Double Spend

The Bitcoin Gold blockchain suffered a second 51% attack in two years, leading to $70,000 worth of BTG being double spent.

10572 Total views 229 Total shares Listen to article 2:44

Source: CoinTelegraph (Jan 2020)

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

BIZ & IT —

Bitcoin security guarantee shattered by anonymous miner with 51% network power

In a first, one player got a monopoly of Bitcoin's total computational power.

DAN GOODIN - 6/15/2014, 9:15 PM

Source: Ars Technica

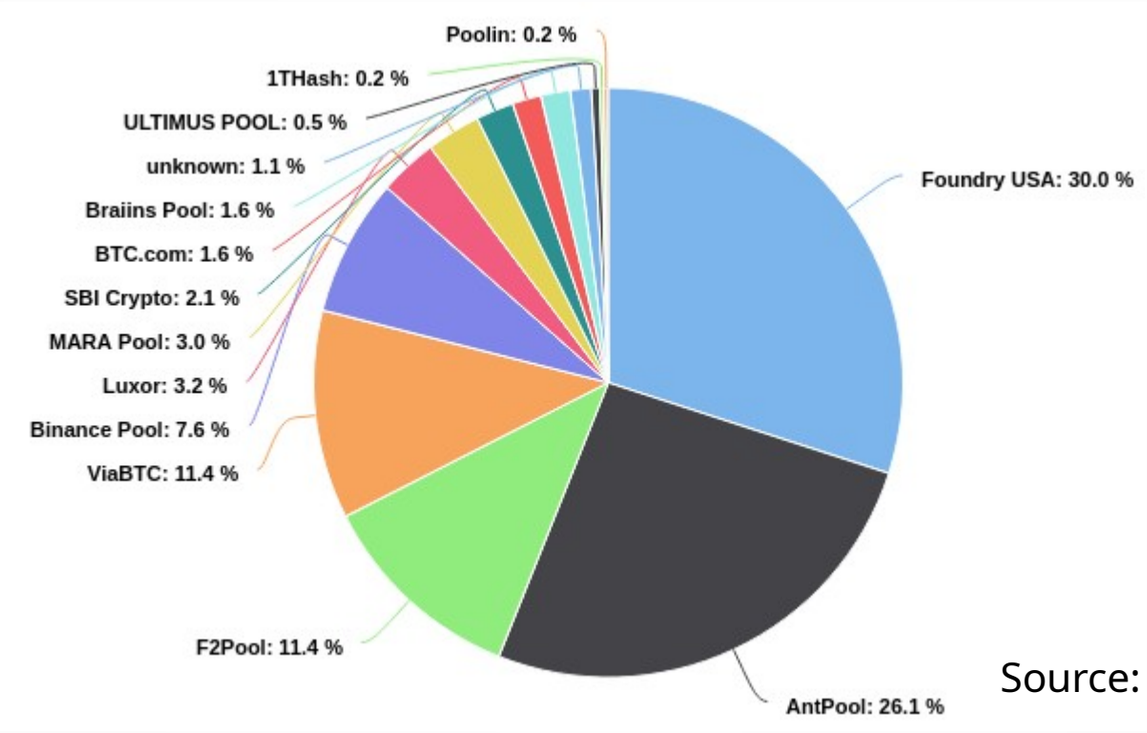
Are 51% Attacks Feasible? (2)

- What about today's Bitcoin?
  - You need to buy and maintain specialized ASIC (Application-Specific Integrated Circuit) equipment
  - The largest part of the bill is electricity
- Difficult to estimate easily
- Let's try another approach…

Are 51% Attacks Feasible? (3)

- Back-of-the-envelope calculation, October 2024:
  - Peak hash rate (source: blockchain.com) has been $6.39*10^{20}$ hash/second (639 exahash!)
  - Such a rate results in 38.4 million US$ per day earning for the miner (source: coinwarz.com)
  - Over a year, that's 14 billion US$ (9.3B$ last year)
- A miner has to operate at a profit, so costs must be lower
- Feasible long term if a powerful player is motivated
  - 0.62% of Italy's GDP, 0.08% of EU GDP, 0.05% of USA GDP
  - ~14% of Apple's yearly profits

Mining Pool Situation, October 2023



Source: btc.com (not existent anymore)

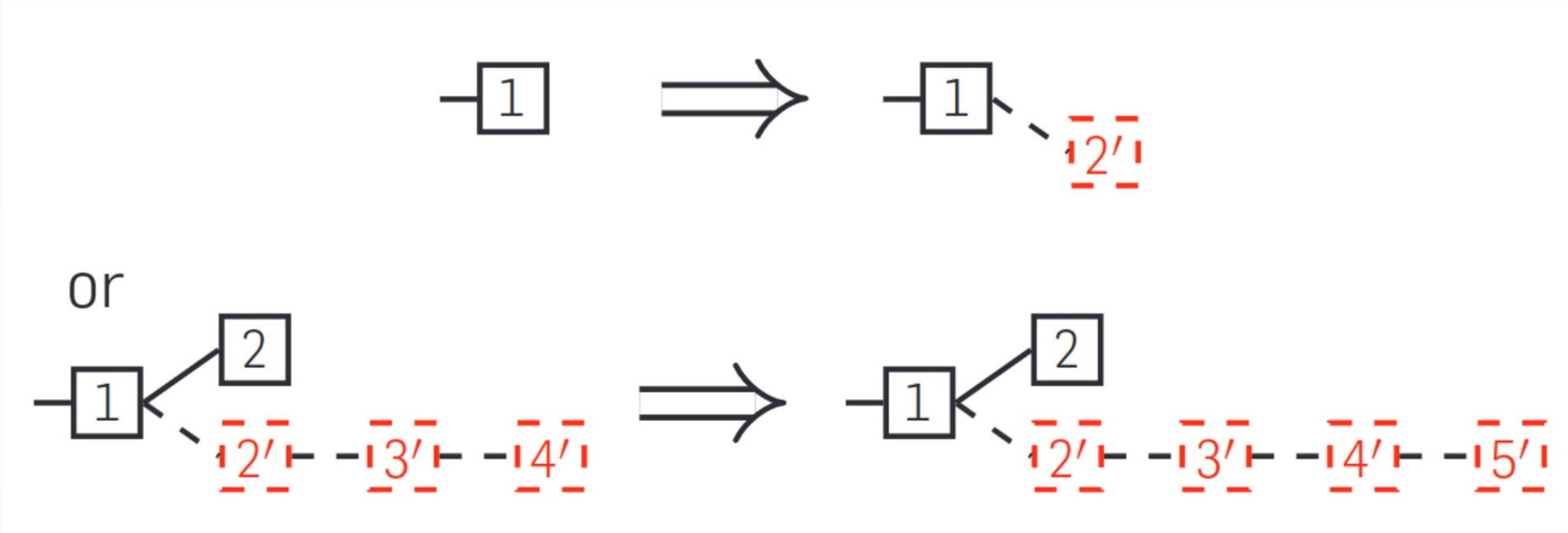
- Current data is quite similar
- Unlike Ghash (the 2014 55% pool), Foundry USA appears to be not anonymous
- Collusion between the top two pools would get a 51% attack

Selfish Mining

A Counterintuitive Strategy

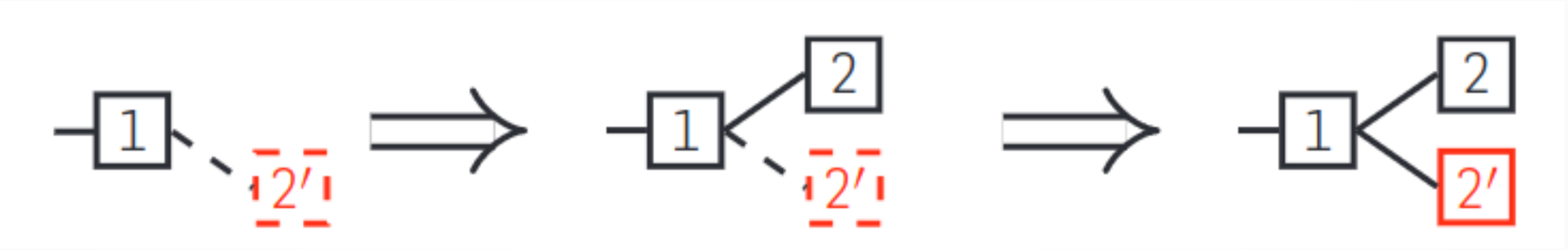
- Majority Is Not Enough: Bitcoin Mining Is Vulnerable, (Eyal and Gün Sirer, 2018)
  - (Login with your university password to access the PDF)
- A miner (more realistically: a pool) can gain more by **not respecting the protocol**, **withholding** (i.e., not revealing immediately) blocks they mined
- Idea: the selfish pool has a **secret fork** they are working on that other miners have no access to

Selfish Mining Strategy (1)



- When the pool finds a block, in most cases, they keep working on their secret (red dashed boxes) fork
  - Exception: two branches of length 1 (we'll see later)

Selfish Mining Strategy (2)



- If the pool was ahead by 1 and others find a block, it **immediately publishes** (solid red) the withheld block
  - The pool keeps working on their fork
  - Other peers will be split between the two (protocol: work on the first you've heard about)
  - In which fractions will the miners split between the two forks? **Key part we'll see later**

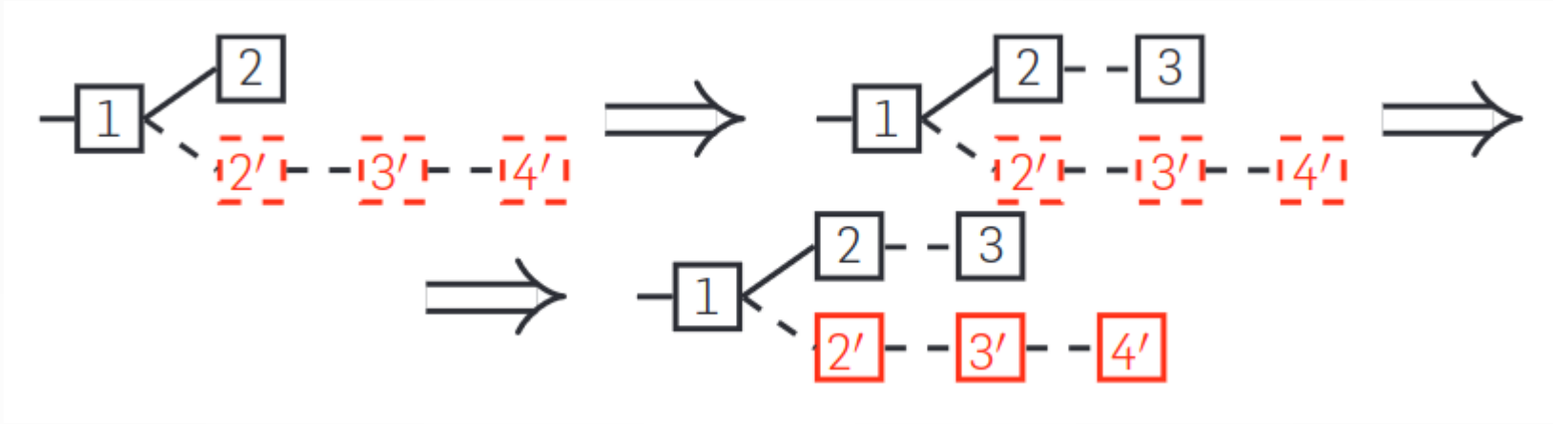
Selfish Mining Strategy (3)

- Depending on who finds the next block we see if the pool “wins” on the bet they made
  - They lose if **others** find the next block **and** they add it to the “**honest” fork**

Selfish Mining Strategy (4)

- If the pool was ahead by more than 2 and the others find a block, no problem at the moment
  - The pool can however reveal a block now to get their rewards right away

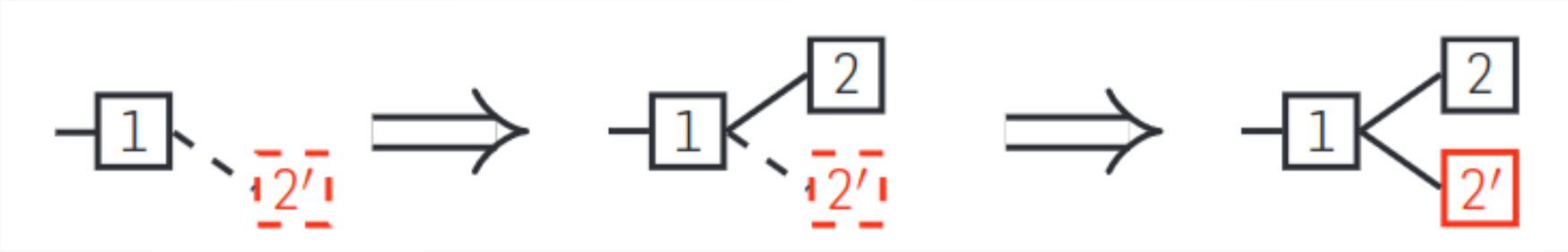
Selfish Mining Strategy (5)



- If the lead drops to one, the pool **immediately publishes** their fork

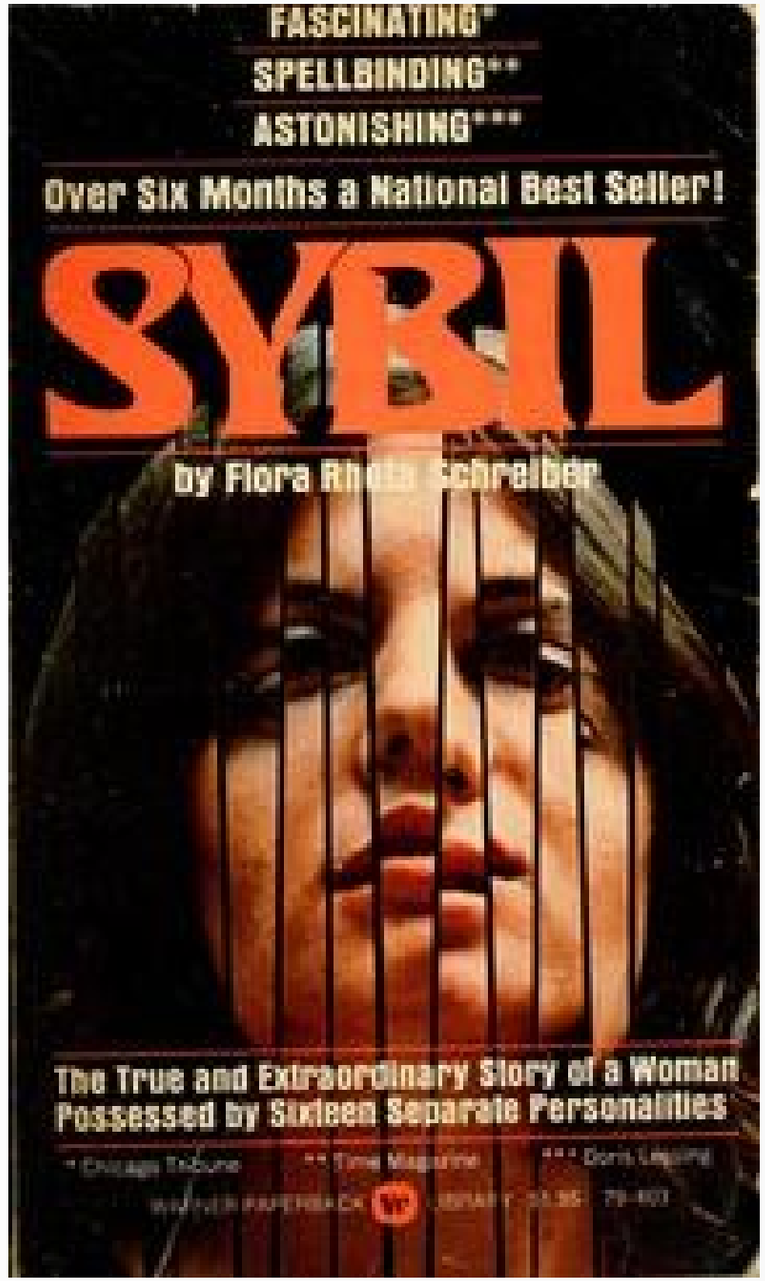
Analysis

- Idea: compute the **fraction of rewards** that goes to the selfish pool
  - In the long run, **after difficulty adjustments**, the total rewards for miners will be **the same**
- Key parameter γ: the fraction of nodes that will work on the selfish fork



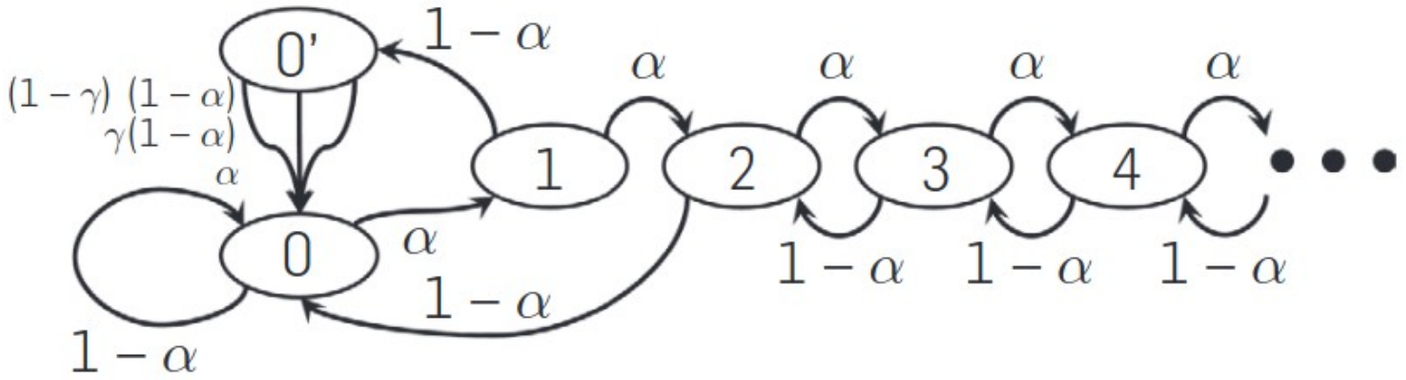
Sybil Attack

- Name from a book about a woman with 16 personalities
- In P2P systems: an attacker creates a very large number of nodes to subvert the system
- In the tie situation, Sybils join the Bitcoin P2P network and **only propagate the attackers' block**, thereby **raising ɣ**
- Proposed countermeasure: in case of competing tied forks, honest nodes propagate all of them



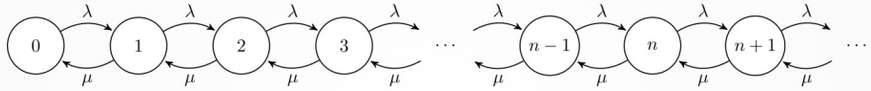
Analysis (1)

Figure 1. State machine with transition frequencies.



- Does this remind you of anything?

M/M/1 Equilibrium



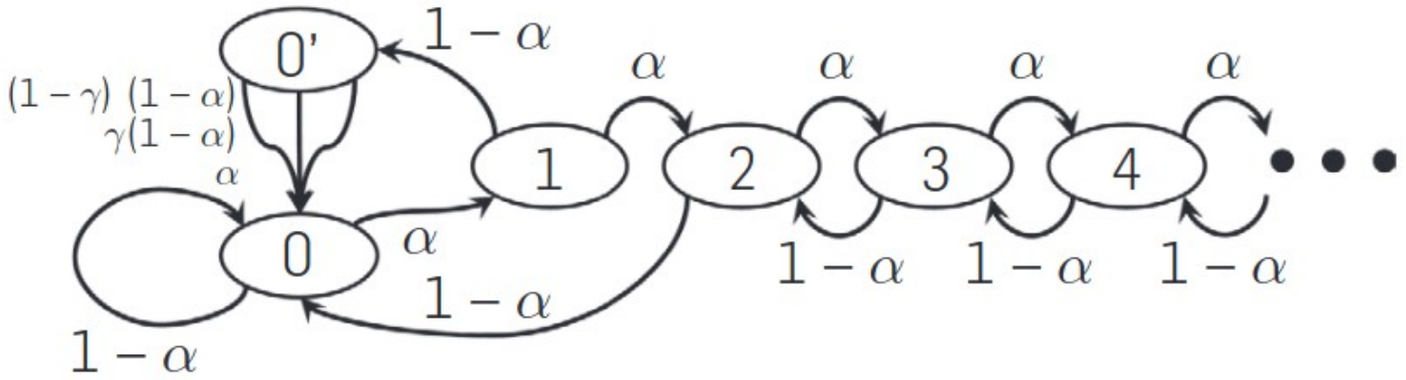
- To be an equilibrium we need
  - $\lambda<\mu$, otherwise the number of elements in the queue will keep growing
  - That in any moment the probability of moving from state *i* to *i+1* is the same of moving in the opposite direction:

$$\lambda\, p_i\, dt = \mu\, p_{i+1}\, dt$$

$$p_{i+1} = \frac{\lambda}{\mu}\, p_i$$

Analysis (2)

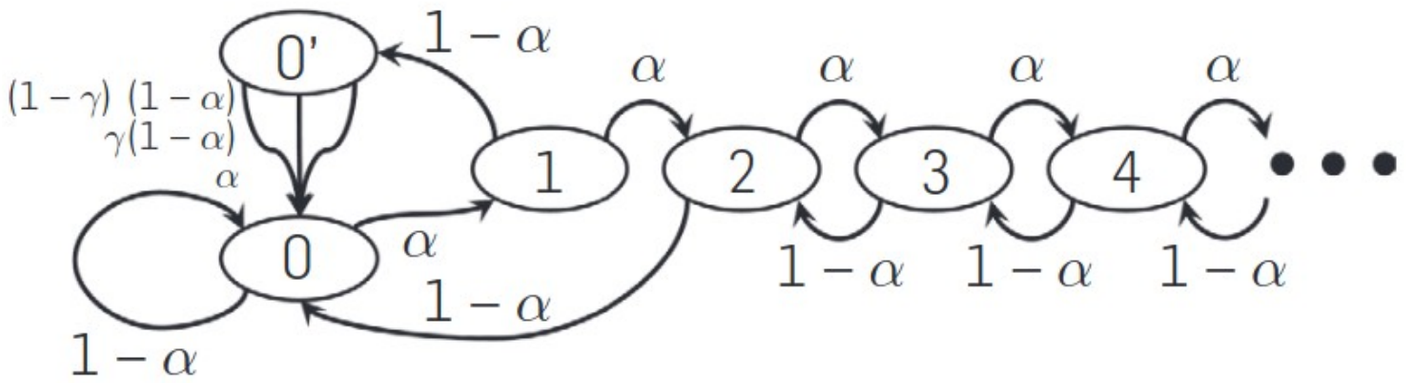
Figure 1. State machine with transition frequencies.



- **Markov Chain** analysis of how the system behaves
- Numbers are the **selfish fork's lead**
- The **0' state** is the **tie situation** in which a fraction ɣ of "honest" nodes works on the attacker's fork

Analysis (3)

Figure 1. State machine with transition frequencies.



- Find the **equilibrium distribution** of **probabilities to be in a given state**
- Associate a **reward** to every transition
- Sum up all **rewards weighted by their frequency** and compare with **those of the pool**

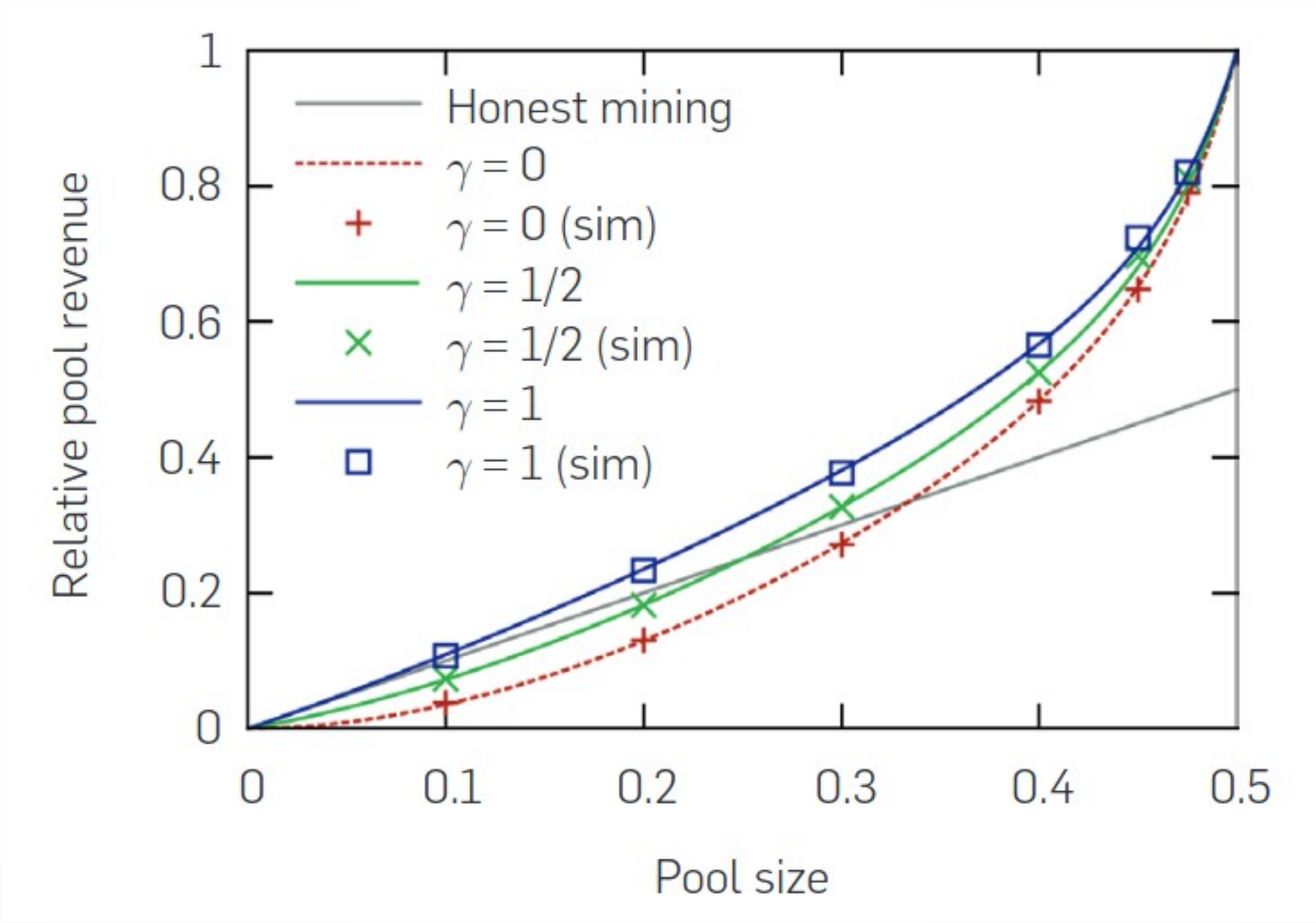
Analysis (4)

$$r_{\text{others}} = \overbrace{p_{0'} \cdot \gamma(1-\alpha) \cdot 1}^{\text{Case (c)}} + \overbrace{p_{0'} \cdot (1-\gamma)(1-\alpha) \cdot 2}^{\text{Case (d)}} + \overbrace{p_0 \cdot (1-\alpha) \cdot 1}^{\text{Case (e)}}$$

$$r_{\text{pool}} = \overbrace{p_{0'} \cdot \alpha \cdot 2}^{\text{Case (b)}} + \overbrace{p_{0'} \cdot \gamma(1-\alpha) \cdot 1}^{\text{Case (c)}} + \overbrace{p_2 \cdot (1-\alpha) \cdot 2}^{\text{Case (g)}} + \overbrace{P[i>2](1-\alpha) \cdot 1}^{\text{Case (h)}}$$

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

Results, Finally!



Announcement

Bitcoin Is Broken

bitcoin security broken selfish-mining

Ittay Eyal and Emin Gün Sirer

November 04, 2013 at 10:30 AM

← Older

Newer →

Bitcoin is broken. And not just superficially so, but fundamentally, at the core protocol level. We're not talking about a simple buffer overflow here, or even a badly designed API that can be easily patched; instead, the problem is intrinsic to the entire way Bitcoin works. All other cryptocurrencies and schemes based on the same Bitcoin idea, including Litecoin, Namecoin, and any of the other few dozen Bitcoin-inspired currencies, are broken as well.

(link)

Public Discussion

- Incredulity and skepticism in the Bitcoin community
- Vitalik Buterin (at the time editor of Bitcoin Magazine):
  ”No honest (or semi-honest) miner would want to join a selfish pool," he suggested. "Even if they do have a small incentive to [join], they have an even greater incentive to not break the Bitcoin network to preserve the value of their own Bitcoins and mining hardware.”
- Authors were skeptical about this and other critiques
- Follow-up paper (slides) dealing with the interplay with the difficulty adjustment algorithm

“Uncle” Blocks

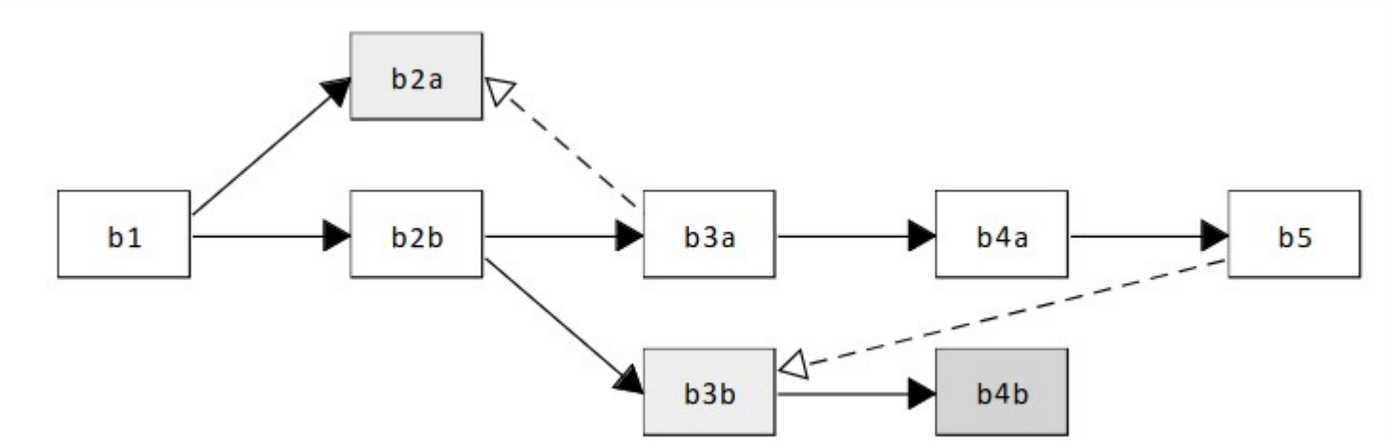


Image from
Ritz and Zugenmaier

- Some cryptocurrencies (e.g., Ethereum, which had proof-of-work) reward linking “orphan” blocks in the blockchain
- Ritz and Zugenmaier (slides) show that selfish mining is still profitable: the selfish miner risks less in ties

Did It Really Happen, Though?

- Nobody has observed real-world selfish mining yet
- As described, it is detectable:
  - “Orphan” blocks (outside of the main chain) do happen due to latency
  - There are pattern: longer orphan chains, timing
- Bahrani and Weinberg proposed an “untraceable” variant (video) wrt orphan chains
  - Requires 38.2% of mining power
  - Impossible to differentiate between honest network and slightly increased latency
  - Can it be done to make timing still look natural?

Reasons Why It Didn't Happen

Bahrani and Weinberg (Appendix A):

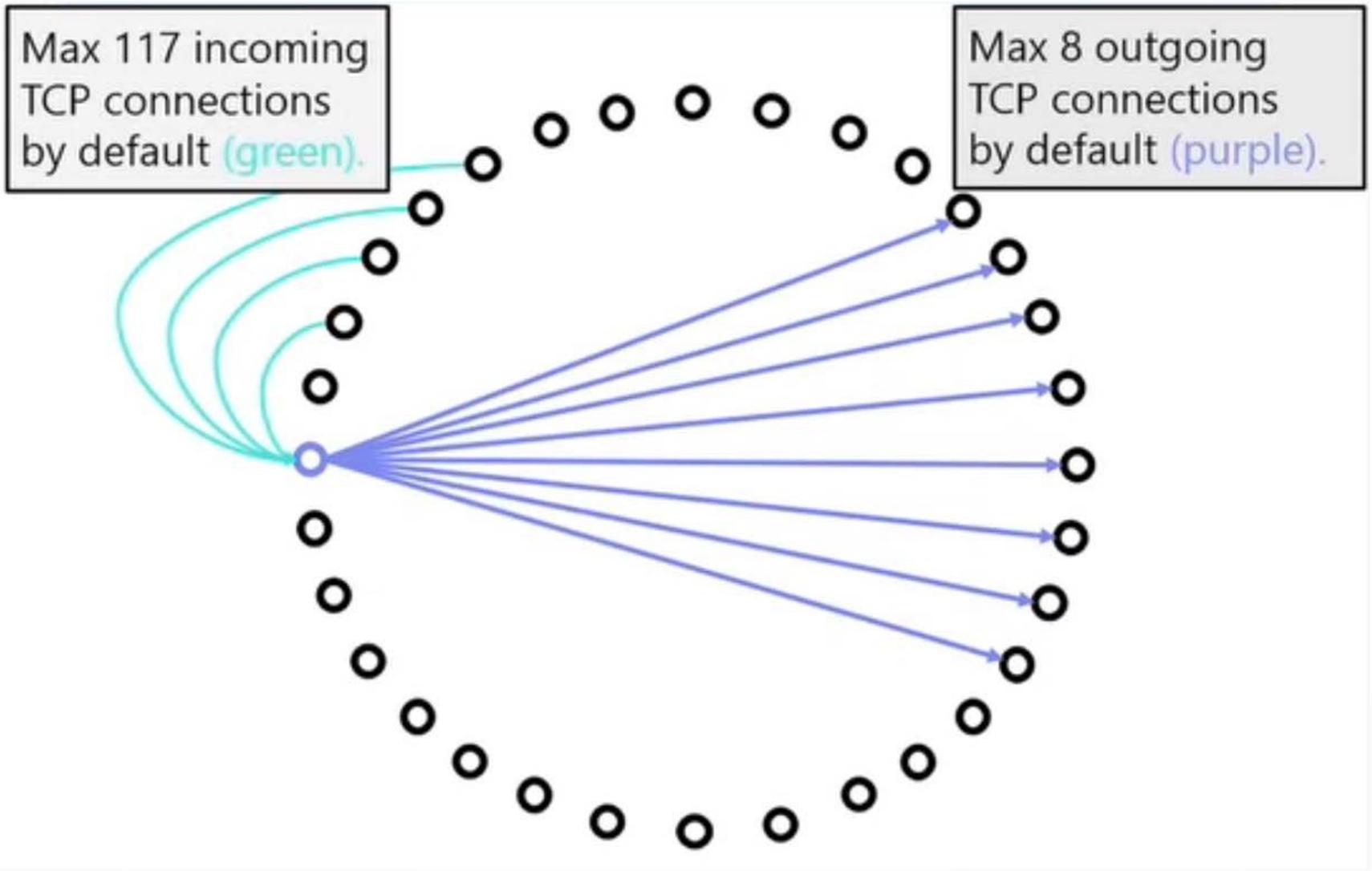
- No entity has enough hashpower/connectivity to be profitable
  - Makes sense only on large cryptocurrencies like Bitcoin
  - In public pools, you'd need public instructions to selfishly mine
- Engineering costs (e.g., dealing with ASICs) might outweigh benefits
  - Even assuming this is true now, it might not be on the long run
- If it's traceable, there may be targeted repercussion
  - E.g., leaving the pool or not accepting the selfish miners' blocks or transactions
- If it's statistically detectable, the value of the cryptocurrency may suffer as a result

Eclipse Attack

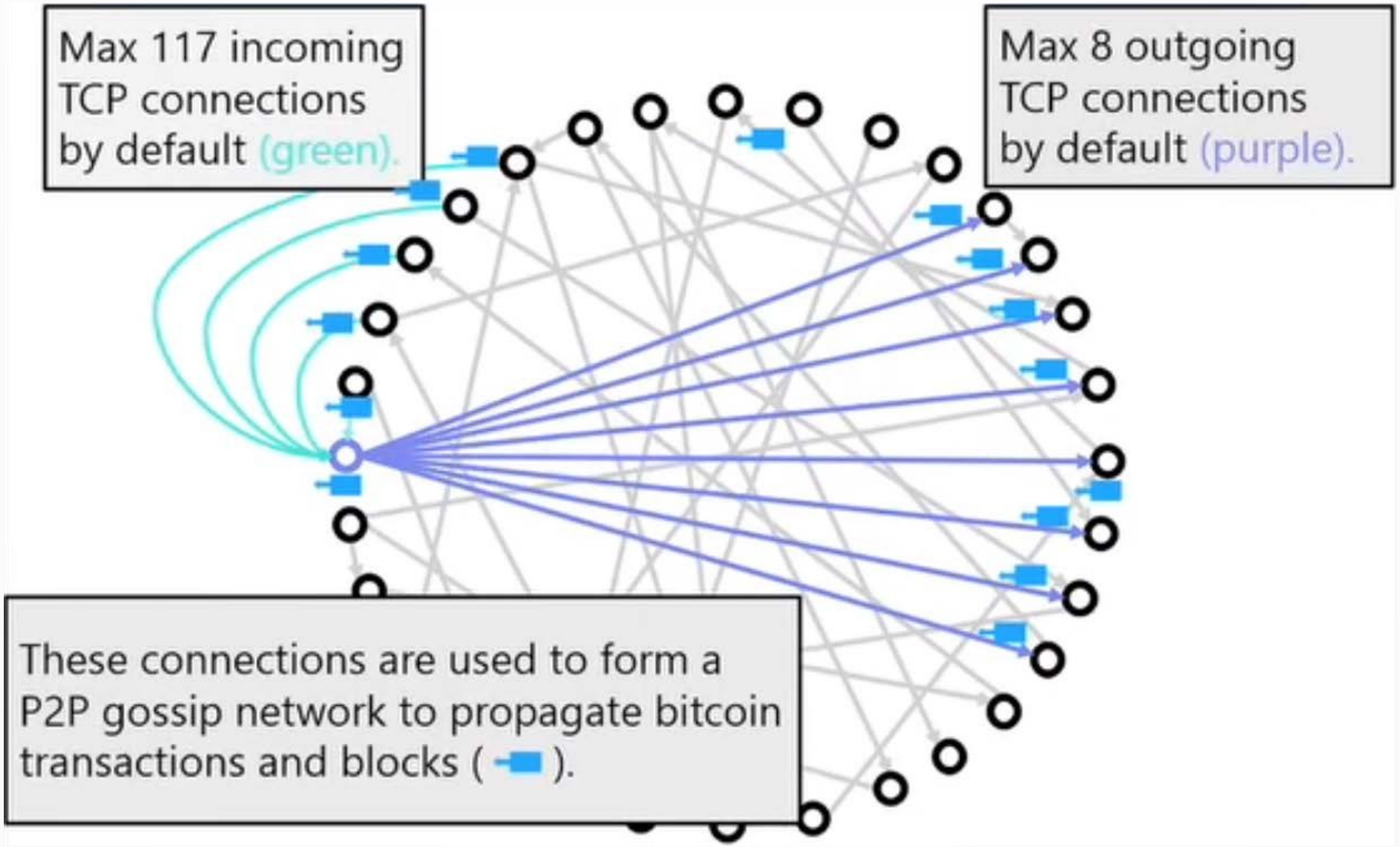
Source

- Heilman et al.,
  Eclipse attacks on Bitcoin’s Peer-to-Peer Network, USENIX Security Symposium 2015
  - The link contains both the article and a video of the presentation

The Bitcoin P2P Network (1)

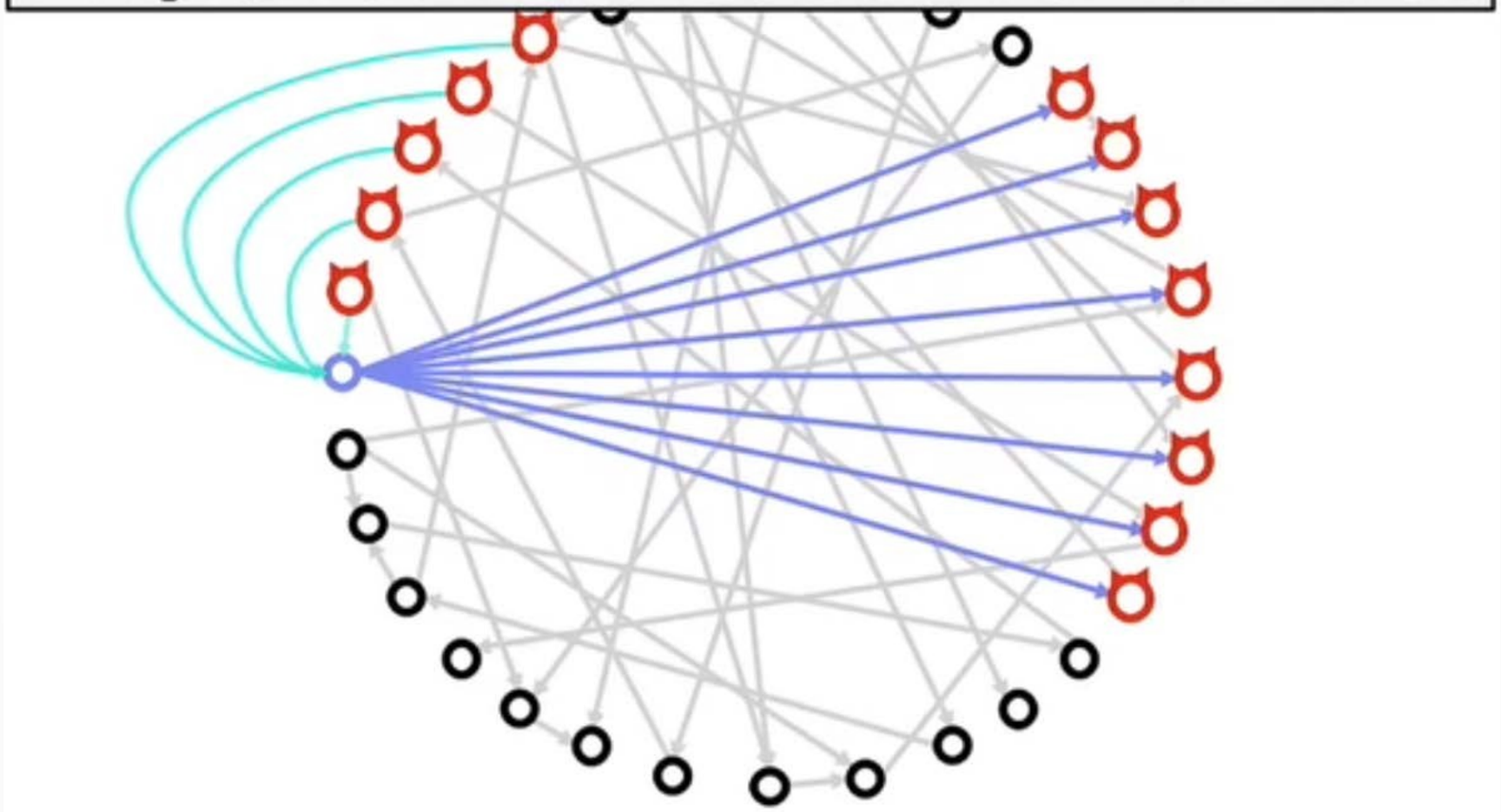


The Bitcoin P2P Network (2)



Eclipse Attack (1)

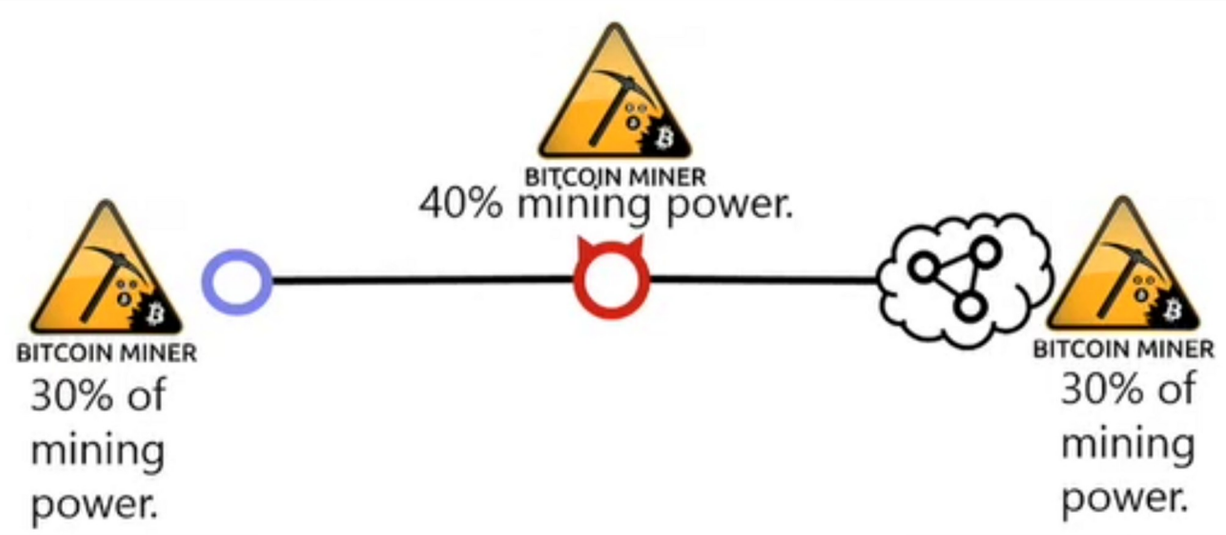
Information Eclipse Attack (def):
Gaining control over a node's access to information in a P2P network.



Eclipse Attack (2)

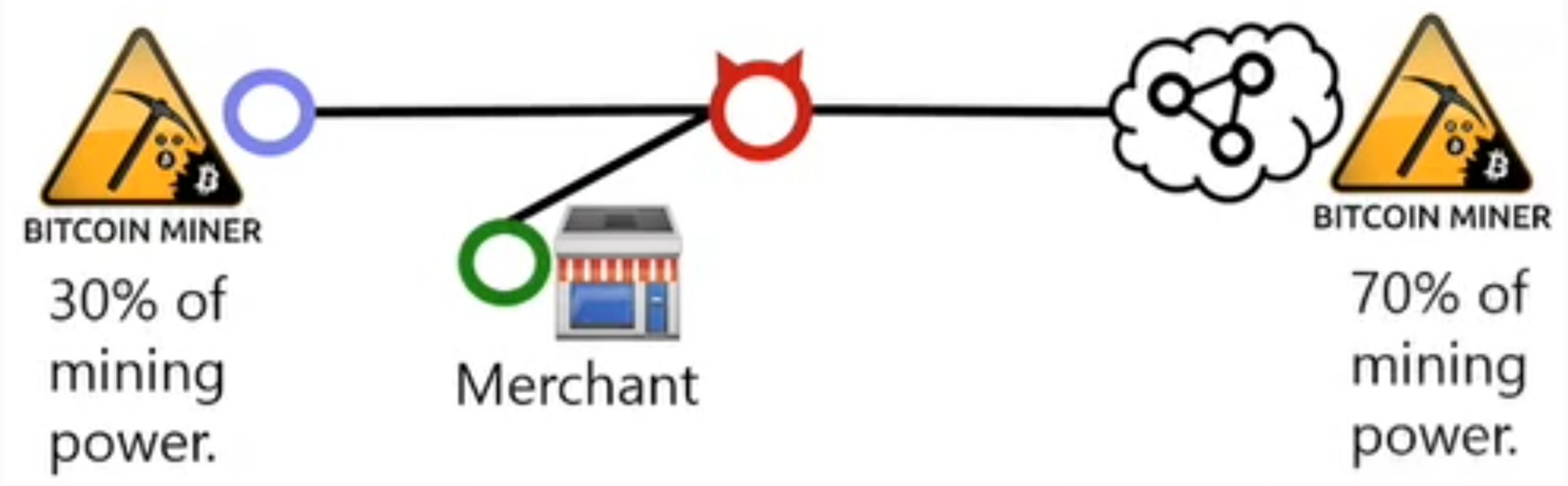
- The attacker **eclipses** the attacked node from the rest of the network
- It can control **everything** the attacked node sees
- Note: the attacker is not **physically** between the attacked node and everybody else, it is only in the **P2P overlay**

51% Attack with Less Mining Power



- The attacker **doesn't relay blocks** between the two sides and **outcompetes both**
- An eclipse attack can be used to **improve selfish mining** by **eclipsing others' blocks**

Double Spending



- Spend some BTCs at the merchant's and **use the eclipsed** ("honest") **miner's blocks** to confirm it
- Spend the same BTCs **somewhere else** in the network
- The first transaction will be in a **shorter blockchain**, hence **cancelled**

“Tried” and “New” Tables

- “Tried”: **64 buckets** (size 64 each) containing nodes with which connections were **already established**
- “New”: **256 buckets** for not yet tested nodes, gossiped by other nodes (“hey, try these new nodes”)
- Each address is split between “first 16 bits” and “rest”; the first 16 (e.g., “130.251” in “130.251.222.83”) is the **group** which is hashed to determine 4 buckets, then the full address determines the final one
- A countermeasure against this kind of attack

How the Attack Works

- The attacker sets up a collection of **Sybil nodes**
- They **contact the attacker node** and gossip information about **other attacking or non-existing nodes**
  - Filling up both “tried” and “new”
- Wait for a node **restart**
- If there are enough “groups” and with some luck, the attacked node will **connect only to the attacker**

Restarts

- A restart (disconnection/reconnection) is needed for the Eclipse attack to work
- In the Bitcoin network, machines restart quite frequently
  - Biryukov and Khovratovich show that a machine with a public IP has a 25% chance of going offline in 10h
- Security updates: either restart or stay vulnerable
- Some attacks can force restarts

Is the Attack Realistic?

- The per-IP address group hashing strategy is the strongest countermeasure faced by the attack
- A “normal” attacker will have IP addresses only from a **few groups**, hence filling only a few buckets
- Unfortunately, **botnets** (i.e., groups of vulnerable devices violated and controlled by a single actor) have large IP diversity
- Authors show the attack is **realistic** even for pretty small botnets (by analysis and attacking their own nodes)

Vulnerabilities and Countermeasures

- Authors proposed six countermeasures to patch Bitcoin's client
- They were eventually all included in the Bitcoin Core client
- Interestingly, they are based on ideas inspired by **botnets defenses**
  - Good guys often try to **eclipse botnets** to make them unreachable by the attacker

Countermeasure 1: Eviction

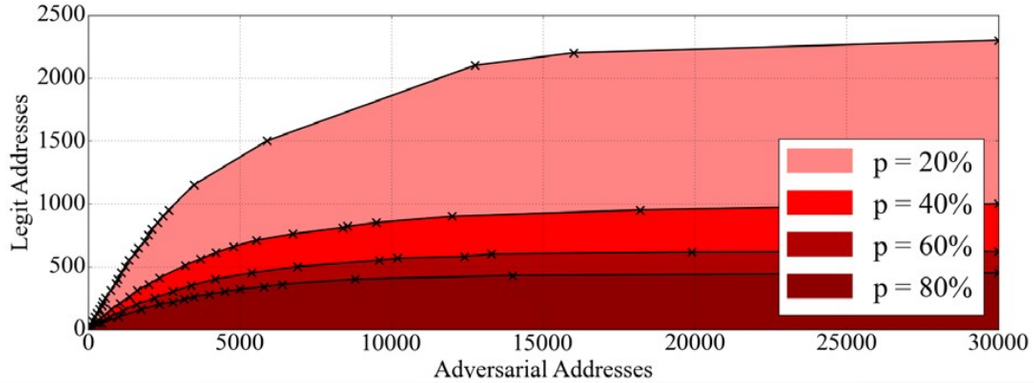
- The “Tried” and “New” tables had this eviction policy, when a bucket got full:
  - Take 4 random IPs in the bucket
  - Evict (remove) the one with the **oldest timestamp** (last time seen/heard of)
  - Put the new node in its place
- The attacker could avoid eviction by making sure its **timestamps were fresher**
- Countermeasure: just hash the IP to get a single slot in the bucket

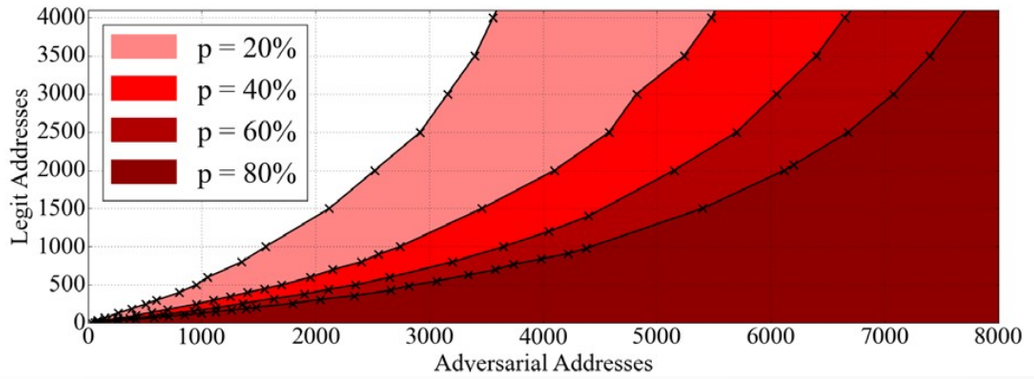
Countermeasure 2: Peer Selection

- A node would connect to peers randomly, but with a strong bias towards **“fresher” peers** (once again, those with newer timestamps)
- Once again, the attacker can ensure its nodes have fresh timestamps
- Countermeasure: select nodes uniformly at random (i.e., each node in Tried and in New has the same probability)

Countermeasure 3: Test Before Evict

- Rather than evicting nodes in the Tried table, try to connect to them and evict them only if they don’t respond
- Plots show the probability of success for the attacker
- If enough honest nodes are in the table, at some point the probability of success “plateaus” for the attacker





Countermeasure 4: Feeler Connections

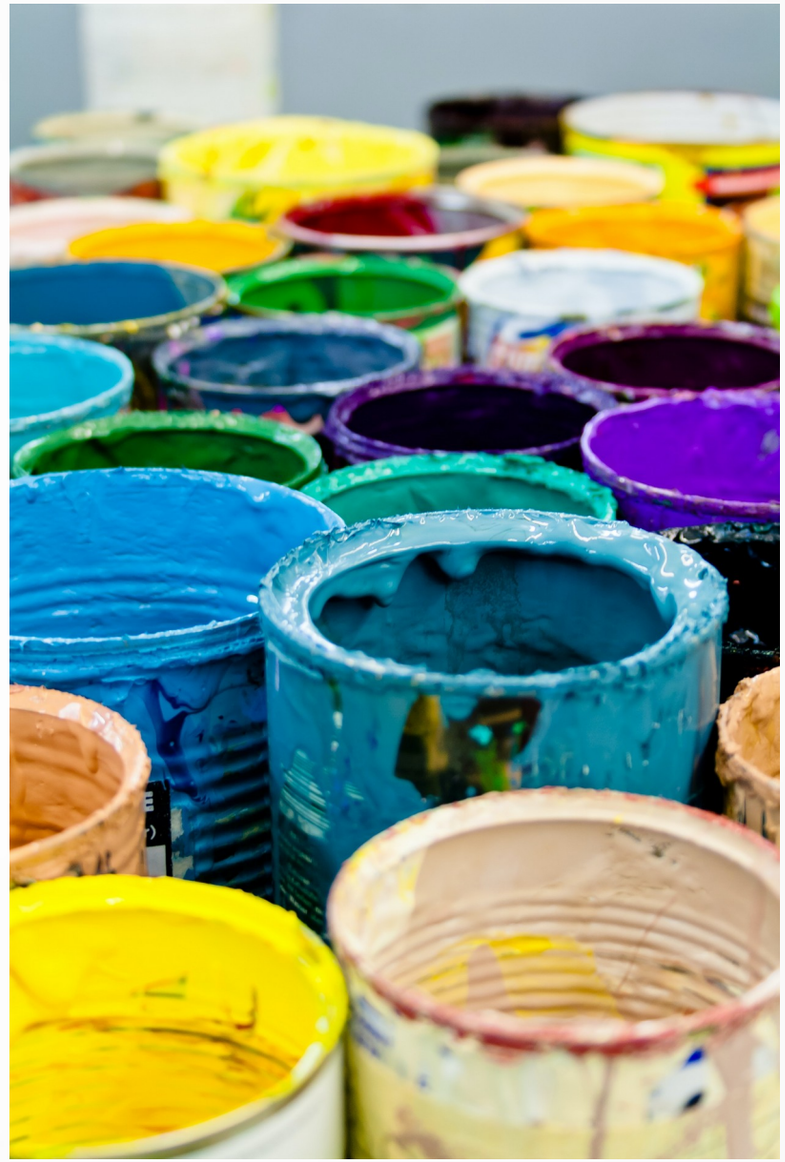
- Run test connections towards random nodes in New:
  - If they succeed, move the address from New to Tried
  - If they fail, remove them from New
- This countermeasure cleans trash from New and increase the number of nodes in Tried that are likely available when a node restarts

Countermeasure 5: Anchor Connections

- A new table: Anchor nodes
- Inspired by a mechanism implemented in Tor, nodes connect to the two oldest Anchor nodes that accept incoming connections
- The attacker needs to disrupt communications to Anchor nodes to make the Eclipse attack feasible

Countermeasure 6: More Buckets

- Quite obviously, more buckets require more Sybil IPs to be filled
- New goes from 256 to 1024 buckets, Tried goes from 64 to 256

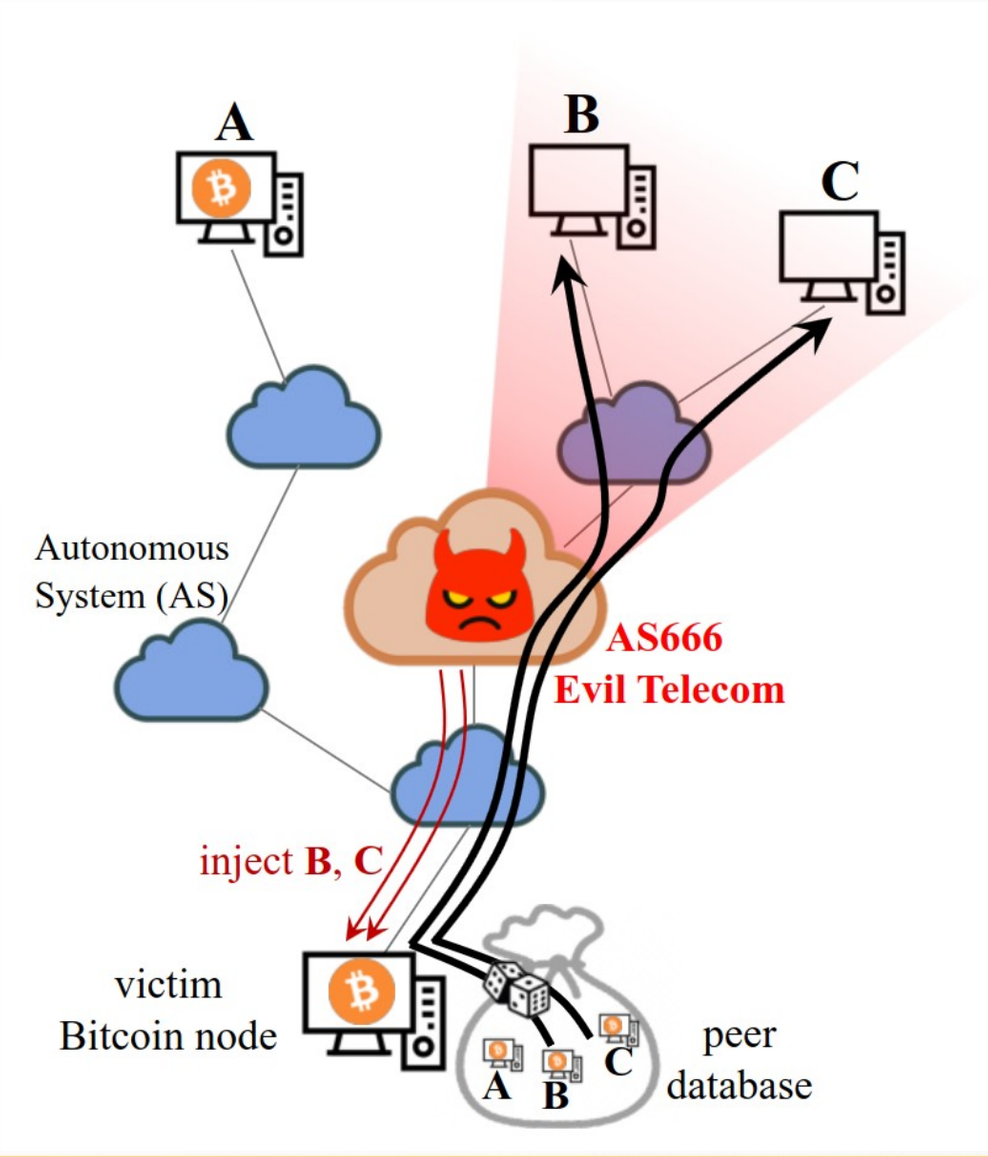


Erebus: A More Powerful Attacker

- Sources:
  - Tran et al., IEEE S&P 2020 (attack)
  - Tran et al., USENIX Security 2021 (countermeasure)

The Attack

- Here we have a **network adversary**: an entity, such as a telecom, that forwards traffic for **many IPs**
- The telecom uses **shadow IPs**, i.e., runs malicious replicas of legitimate IP addresses
- Very difficult to detect



The Defense

- Routing Aware Peering: diversifying based on routing
  - Unfortunately, there are some **false negatives**
  - The attacker can **exploit them**
- Tweaks (changing parameters such as table sizes) to improve resilience
  - Alone, limited effectiveness
- Strategies are effective when combined