GAME THEORY & NETWORK ATTACKS: HOW TO DESTROY BITCOIN

Max Fang
Philip Hayes

BLOCKCHAIN
AT BERKELEY

ABOUT MAX

HONG KONG 2017

- **UC Berkeley** Student: CS + Econ
- Got into Bitcoin Feb 2014 doing GPU-based cryptocurrency mining
- Worked at **ChangeTip** for a year
    - Acqui-hired by Airbnb
- Since 2014, run **Blockchain at Berkeley** (Previously **Bitcoin Association of Berkeley**)
    - **Berkeley Bitcoin Meetup - Fall 2016**
    - **Blockchain Consultancy - Fall 2016**
    - **Education + R&D Department - Spring 2017**
    - ~250 unique monthly visitors
- Independent research into Bitcoin mixing
- Designed, teach, and lecture for the **Cryptocurrency Decal**
    - World's only undergraduate cryptocurrency course
- Soon to be developer evangelist intern at **Lightning Network!**

Me and the ChangeTip team

BLOCKCHAIN
AT BERKELEY

BLOCKCHAIN
AT BERKELEY

ABOUT PHILIP

HONG KONG 2017

- **UC Berkeley** Student: EECS (Electrical Engineering & Computer Science)
- Ran Bitcoin miners off of High school's library computers
- Worked at **IBM**
- Researched Distributed, Decentralized Bitcoin Mixing
- **Undergraduate Researcher** under John F. Canny at UC Berkeley
    - Worked on **BIDMat/BIDMach** high-performance GPU linear algebra and machine learning libraries.
- Created/taught the **Cryptocurrency Decal**
- **Lightning Network** contributor

BLOCKCHAIN AT BERKELEY

LECTURE OUTLINE

HONG KONG 2017

1. Censorship and Pool Cannibalization
2. Selfish Mining: Analysis and Defense
3. Network Attacks and ASICBOOST
4. Stubborn Mining and Eclipse Attack Compositions
5. Collusions and Conclusions

1 CENSORSHIP AND POOL CANNIBALIZATION

1.1 CENSORSHIP

BLACKLISTING VIA PUNITIVE FORKING

HONG KONG 2017

You are a government that has jurisdiction over mining pools, say China.

Objective: Censor the Bitcoin addresses owned by certain people, say Gary Johnson, and prevent them from spending any of their Bitcoin

Block containing transactions from Gary Johnson

Normal block

Block mined by Chinese miners

BLOCKCHAIN AT BERKELEY

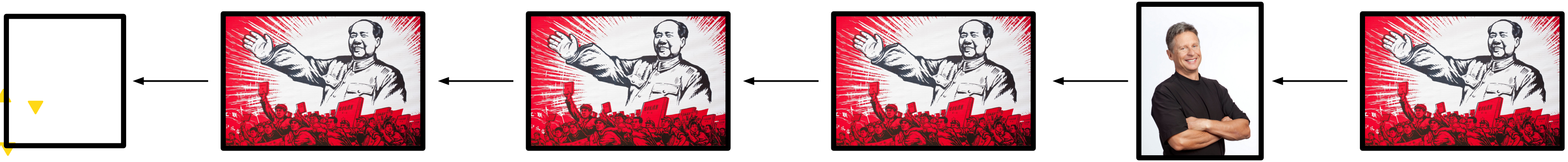
BLACKLISTING VIA PUNITIVE FORKING

HONG KONG 2017

First strategy:

Tell your country's mining pools not to include Johnson's transactions **(blacklisting)**

- Doesn't work unless you are 100% of the network
- Other miners will eventually include Gary Johnson's transactions in a block
- Can only cause delays and inconveniences



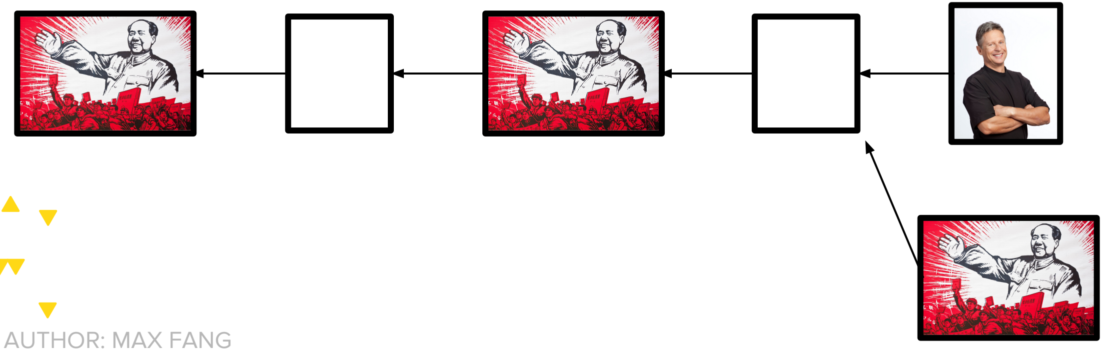
AUTHOR: MAX FANG

BLACKLISTING VIA PUNITIVE FORKING

HONG KONG 2017

Second strategy:

- Remember, you are China; you have >51% of the network hashrate
- Mandate that Chinese pools will refuse to work on a chain containing transactions spending from Gary Johnson's Bitcoin address
- Announce this to the world

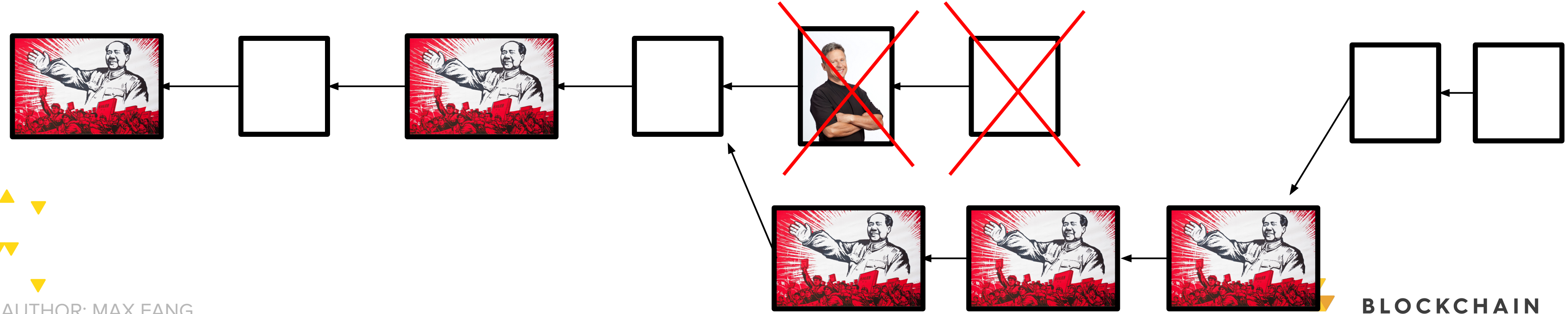


AUTHOR: MAX FANG

BLACKLISTING VIA PUNITIVE FORKING

HONG KONG 2017

- If non-Chinese miners include a transaction from Johnson in a block, China will fork and create a longer proof of work chain
- Block containing Johnson's transaction now invalidated, can never be published

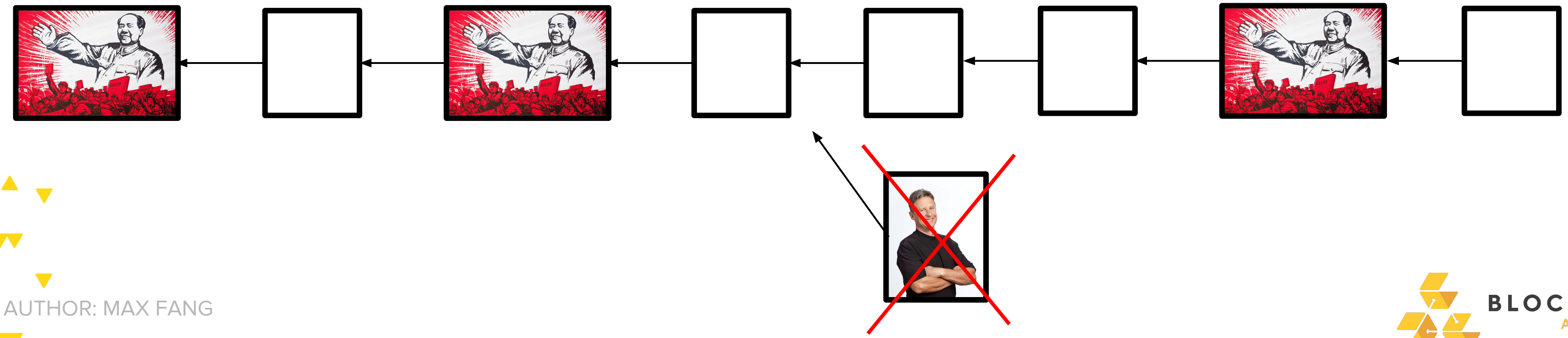


BLACKLISTING VIA PUNITIVE FORKING

HONG KONG 2017

- Non-Chinese miners eventually stop trying to include Johnson's transactions when mining blocks, since they know that their block will be invalidated by Chinese miners when they do

We have now shown how a 51% majority can prevent anyone from accessing their funds. This is called **punitive forking.**

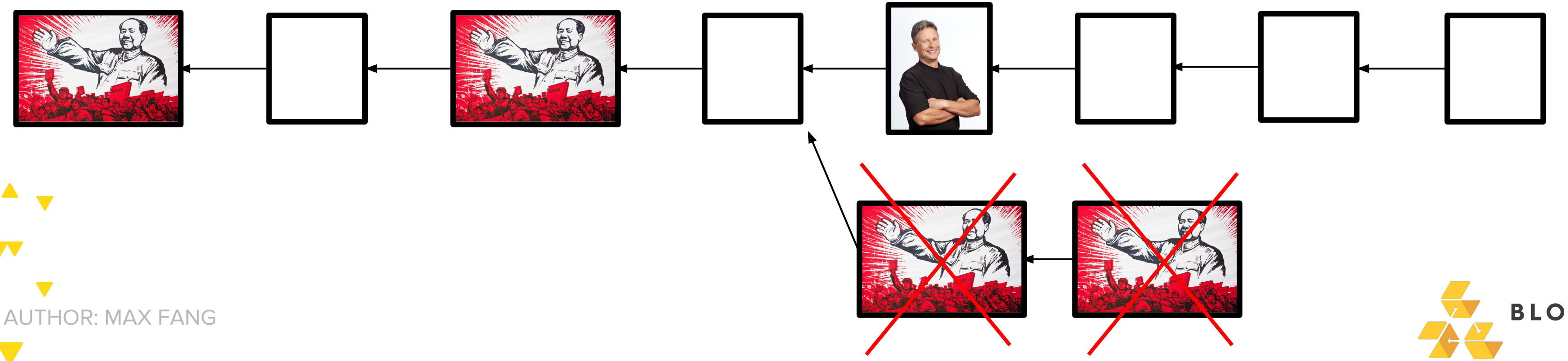


BLACKLISTING VIA FEATHER FORKING

HONG KONG 2017

Punitive forking doesn't work unless you have >51% of hashpower. Is there another way?
Yes! Called **Feather Forking**

- New strategy: Announce that you will **attempt** to fork if you see a block from Gary Johnson, but you will give up after a while
    - As opposed to attempting to fork forever; doesn't work without >51%
- Ex. Give up after block with Johnson's tx contains **k** confirmations



AUTHOR: MAX FANG

BLACKLISTING VIA FEATHER FORKING

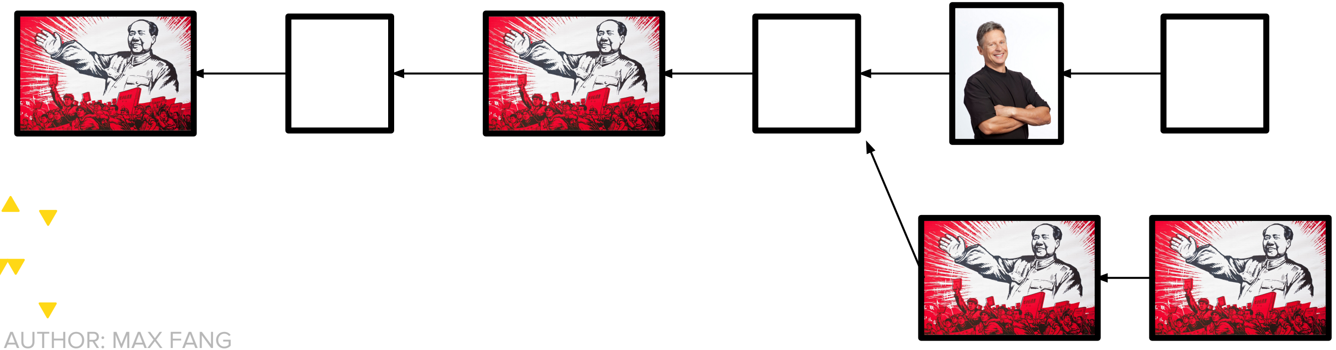
HONG KONG 2017

Let **q** equal the proportion of mining power you have, $0 < \mathbf{q} < 1$

Let **k** = 1: You will give up after 1 confirmation (one additional block)

- Chance of successfully orphaning (invalidating) the Johnson block = $\mathbf{q^2}$

If $\mathbf{q = .2}$, then $\mathbf{q^2 = 4\%}$ chance of orphaning block. Not very good



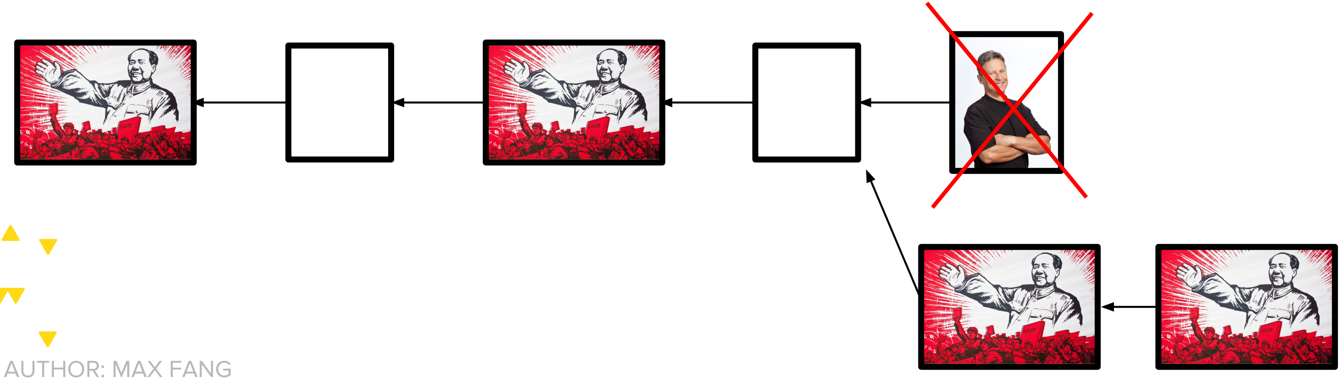
BLACKLISTING VIA FEATHER FORKING

HONG KONG 2017

But other miners are now aware that their block has a $\mathbf{q^2}$ chance of being orphaned. They must now decide whether they should include Johnson's tx in their block

EV(include) = (1 - $\mathbf{q^2}$) * BlockReward + Johnson's tx fee

EV(don't include) = BlockReward



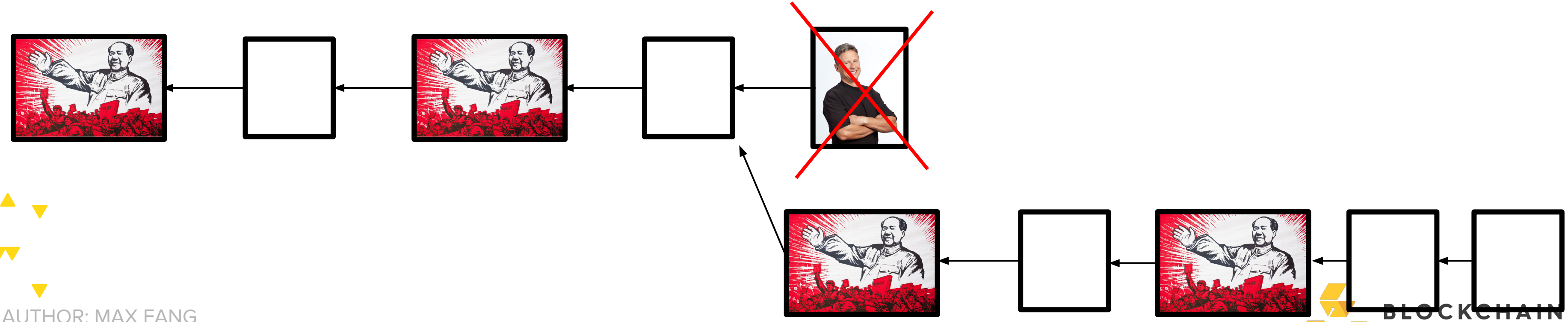
BLACKLISTING VIA FEATHER FORKING

HONG KONG 2017

EV(include) = (1 - $\mathbf{q}^2$) * BlockReward + Johnson's tx fee
EV(don't include) = BlockReward

Therefore, unless Gary Johnson pays $\mathbf{q}^2$ * BlockReward in fees for his transaction, other miners will mine on the malicious chain

- 4% * 12.5 BTC = 0.5 BTC = Johnson must pay **$900** minimum/transaction



1.2

POOL CANNIBALIZATION

POOL CANNIBALIZATION

MARTIN KOPPELMANN, SF BITCOIN DEVS

You have 30% of the hashrate. Assume 1 BTC block reward. All of the following numbers are expected value.

- 30% HR (hashrate)
  = 30% MR (Mining Reward) = 0.3 BTC

You buy more mining equipment, worth 1% of current network hashrate

Standard mining strategy

- Add 1% HR => 31/101 = 30.69% HR = .3069 BTC
  - Revenue gain = **0.0069 BTC for 1% hashrate added**

Cannibalizing Pools - Distribute your 1% equally among all other pools, withhold valid blocks.

- Rewards will still be received
- Undetectable unless statistically significant

Other pool hashrate breakdown:

- (70/71 honest, 1/71 dishonest)
  = 70% honest hashrate = .7 BTC
- You own (1/71) of other pools, so expected value of mining there is
  (1/71) * .7 = **0.0098 BTC**
- **0.0098 (cheat) > 0.0069 (honest)**

More profitable to cannibalize pools than mine honestly

AUTHOR: MAX FANG

(Originally presented by Martin Koppelmann at SF Bitcoin Devs Seminar)

THE MINER'S DILEMMA

ITTAY EYAL, "THE MINER'S DILEMMA"

- Eyal optimizes profitability under how much hashrate to dedicate to attacking
- Model attack decisions as an iterative game
    - Two players: pool A and pool B
- Each iteration of the game is a case of the Prisoner's Dilemma
    - Choose between attacking or not attacking

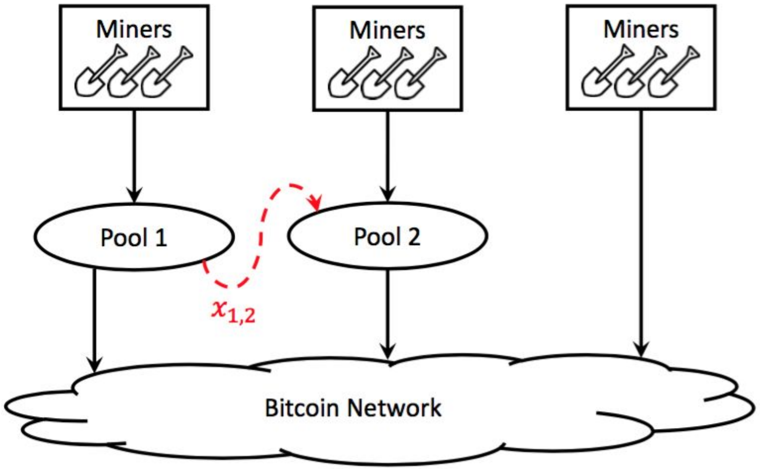


Fig. 3. The one-attacker scenario. Pool 1 attacks pool 2.

THE MINER'S DILEMMA

ITTAY EYAL, "THE MINER'S DILEMMA"

- If pool A chooses to attack pool B, pool A gains revenue, pool B loses revenue
  - Pool B can retaliate by attacking pool A and gaining more revenue
- Thus, attacking is the dominant strategy in each iteration
  - Therefore if both pool A and pool B attack each other, they will be at a Nash Equilibrium
  - **Both will earn less than they would have if neither of them attacked.**

Pool 2 \ Pool 1	no attack	attack
no attack	$(r_1 = 1, r_2 = 1)$	$(r_1 > 1, r_2 = \tilde{r}_2 < 1)$
attack	$(r_1 = \tilde{r}_1 < 1, r_2 > 1)$	$(\tilde{r}_1 < r_1 < 1, \tilde{r}_2 < r_2 < 1)$

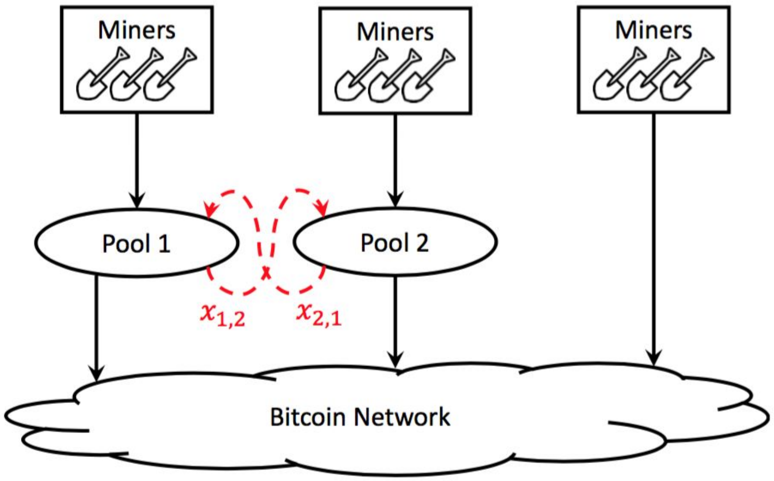


Fig. 7. Two pools attacking each other.

THE MINER'S DILEMMA

ITTAY EYAL, "THE MINER'S DILEMMA"

- No-pool-attacks is not a Nash equilibrium
  - If none of the other pools attack, a pool can increase its revenue by attacking the others
- But if the pools agree not to attack, both (or all) benefit in the long run.
  - However, this is an unstable situation since on a practical level you can attack another pool anonymously
- If pools can detect attacks then maybe an optimistic long term solution is feasible

Pool 2 \ Pool 1	no attack	attack
no attack	$(r_1 = 1, r_2 = 1)$	$(r_1 > 1, r_2 = \tilde{r}_2 < 1)$
attack	$(r_1 = \tilde{r}_1 < 1, r_2 > 1)$	$(\tilde{r}_1 < r_1 < 1, \tilde{r}_2 < r_2 < 1)$

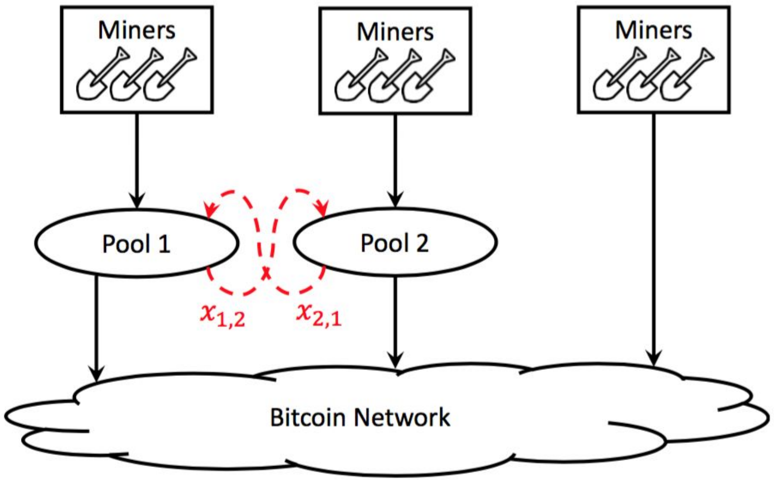


Fig. 7. Two pools attacking each other.

Nash Equilibrium is a Tragedy of the Commons

2 SELFISH MINING: ANALYSIS AND DEFENSE

2.1 REVIEW: SELFISH MINING

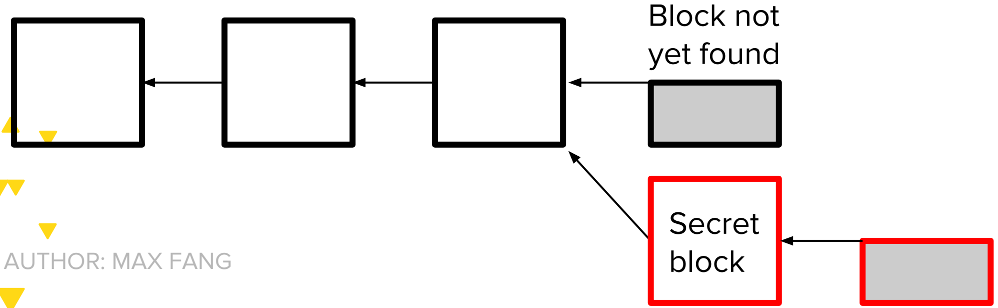
SELFISH MINING (BLOCK-WITHHOLDING)

HONG KONG 2017

You are a miner; suppose you have just found a block.

- Instead of announcing block to the network and receiving reward, keep it secret
- Try to find two blocks in a row before the network finds the next one

This is called **selfish mining** or **block-withholding**



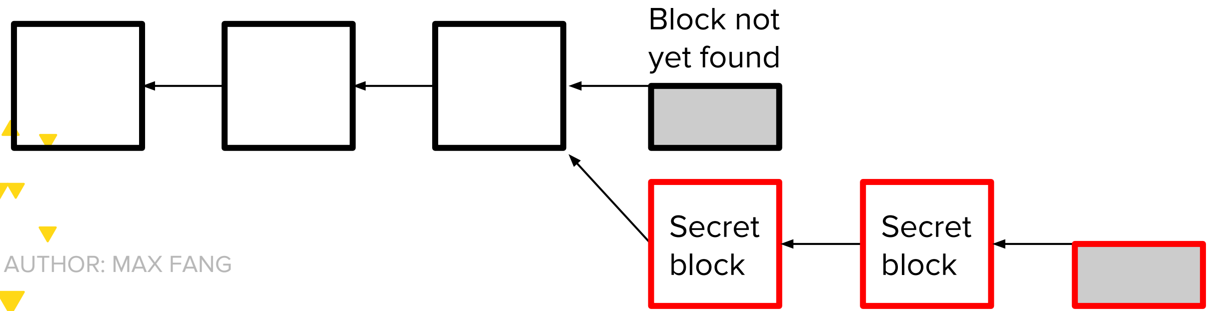
Note: "block-withholding" is also sometimes used in the context of mining pools - submitting shares but withholding valid blocks

SELFISH MINING (BLOCK-WITHHOLDING)

HONG KONG 2017

If you succeed in finding a second block, you have fooled the network

- Network still believes it is mining on the longest proof of work chain
- You continue to mine on your own chain

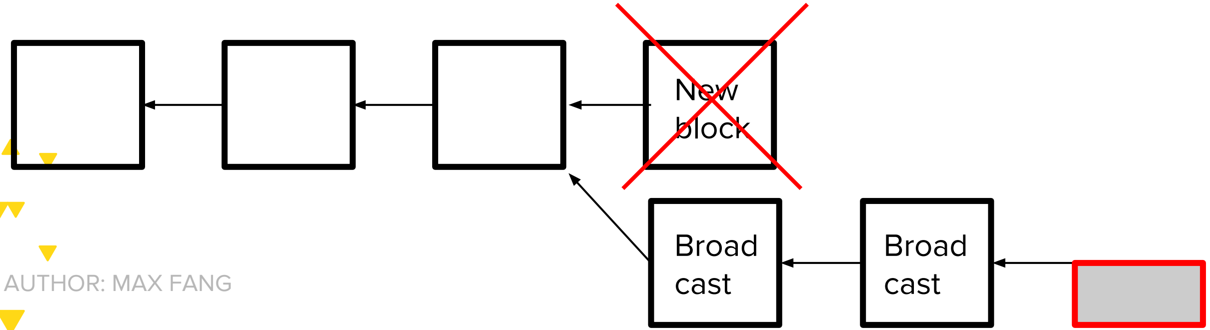


SELFISH MINING (BLOCK-WITHHOLDING)

HONG KONG 2017

If the network finds a block, you broadcast your two secret blocks and make the network block invalid

- While network was working on the invalid block, you got a bunch of time to mine by yourself... for free!
- Free time mining on network
  => higher effective proportion of hashrate **=> higher expected profits!**



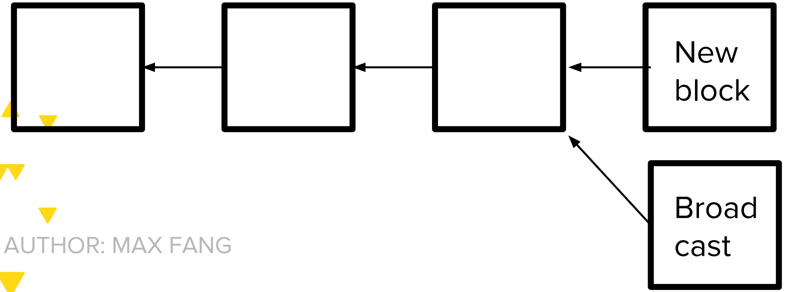
BLOCKCHAIN AT BERKELEY

SELFISH MINING (BLOCK-WITHHOLDING)

HONG KONG 2017

But what if the network found their new block before you could find a second one? **Race to propagate!**

- If on average you manage to tell 50% of the network about your block first:
  - Malicious strategy is more profitable if you have >25% mining power
- If you have >33% mining power, **you can lose the race every time and malicious strategy is still more profitable!**
  - (actual math omitted due to complexity)



BLOCKCHAIN AT BERKELEY

2.2

PUBLISH OR PERISH: A DEFENSE

DEFENSES: BLOCK VALIDATION

HONG KONG 2017

Dummy block signatures
Proposed by Schultz (2015), Solat and Potop-Butucaru (2016)

- Accompany solved blocks with signatures on dummy blocks
- Proves that the block is witnessed by the network
    - Proves that a competing block is absent before miners are able to work on it
- However, does not provide a mechanism to evaluate whether the number of proofs is adequate to continue working
- Does not discuss how to prevent Sybil attacks on signatures
    - Selfish miner generates many signatures on the dummy block

DEFENSES: FORK-PUNISHMENT

HONG KONG 2017

Fork-punishment rule

Proposed by Lear Bahack (2013)

- Competing blocks receive no block reward
- The first miner who incorporates a poof of the block fork in the blockchain gets half of the forfeited rewards
- However, honest miners suffer collateral damage of this defense
  - This defense constitutes another kind of attack

All three of these defenses require fundamental changes to the block validity and reward distribution rules

- Requires a hard fork to implement
  - We have hard enough time fixing transaction malleability

Can we do better?

DEFENSES: TIE-BREAKING

HONG KONG 2017

Uniform Tie Breaking

Proposed by Eyal and Sirer (2014)

- In the case of a tie, a miner randomly chooses which chain to mine on
  - Prevents an attacker from benefiting from network-level dominance
- Raises the profit threshold from 0% to **25%** under their strategy
  - Sapirshtein (2015) proposes a more optimal selfish mining strategy
  - Reduces Eyal and Sirer profit threshold to **23.2%**

DEFENSES: TIE-BREAKING

HONG KONG 2017

Unforgeable timestamps

Proposed by Ethan Heilman (2014)

- Each miner incorporates the latest unforgeable timestamp issued by a trusted party into the working block
    - Timestamp is publically accessible and unpredictable
    - Issued with an interval of 60s
- When two competing blocks are received within 120s, a miner prefers the block whose timestamp is "fresher"
- Claim: Raises the profit threshold to 32%

Drawbacks

- Tie-breaking rules don’t apply when the selfish mining chain is longer than the public chain
    - Only applies to a block propagation race
- If an attacker has a large amount of computational power >40% then these defenses are essentially worthless

PUBLISH OR PERISH

ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

Ren Zhang and Bart Preneel (Apr 2017) claim the best-yet defense of selfish mining

- Backwards compatible: No hard fork
- Disincentivizes selfish mining even when if the selfish miner has a longer chain

Approach: A novel **F**ork-**R**esolving **P**olicy (FRP)

- Replace the original Bitcoin FRP (length FRP), with a **weighted FRP**
  - Embed in the working block the hashes of all its uncle blocks
- Note that selfish mining is premised on the idea of first building a secret block
- Idea: Make sure this secret block does not help the selfish miner win the block race

PUBLISH OR PERISH

ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

Zhang and Preneel's **Weighted Fork Resolving Policy**:

1. If one chain is longer *height-wise* than the other(s) by **k** or greater blocks*
   a. The miner will mine on this chain
2. Otherwise, the miner will choose the chain with the largest *weight*
3. If the largest weight is achieved by multiple chains simultaneously, then the miner chooses one among them randomly

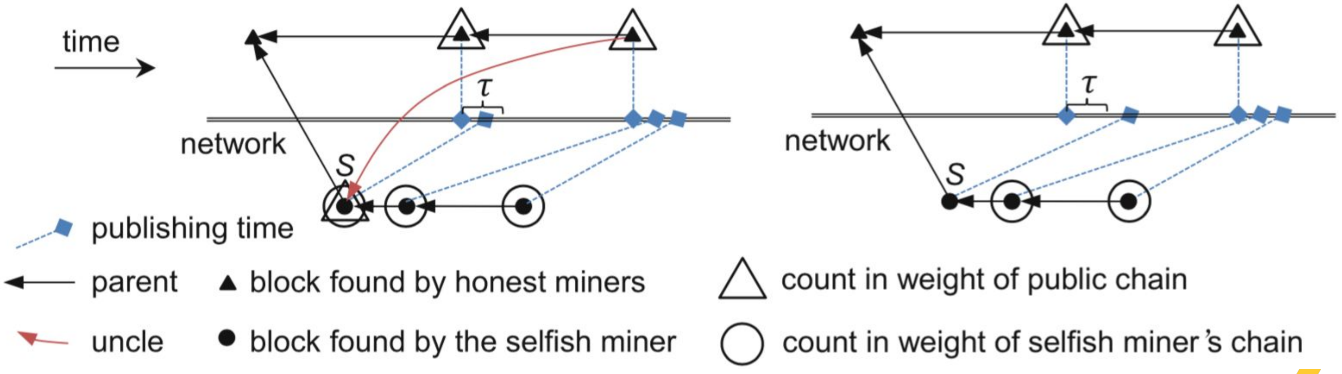
Aside: k* *is a "fail-safe parameter" that gauges the allowed amount of network partition. Note that when* ***k* *= ∞ the first rule never applies.*

PUBLISH OR PERISH

ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

Miner has one secret block. A competing block is published. **Block race**! Miner has two options:

- Option 1: If the selfish miner **publishes** their block, *the next honest block gains a higher weight* by embedding a proof of having seen this block
- Option 2: If the selfish miner **keeps their block secret**, the secret block *does not contribute* to the weight of its own chain
- In both scenarios, the secret block does not help the selfish miner win the block race



Choice 1: Publish **Choice 2: Don't publish**

PUBLISH OR PERISH

ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

Definitions

- τ: An assumed upper bound on the amount of time it takes to propagate blocks across the Bitcoin network
- **In time.** Evaluated from the miner's local perspective.
    1. Height value is greater than that of the local head OR
    2. Height value is same as that of the local head, but was propagated within τ time
- **Uncle**. Different from Ethereum's definition of an uncle
    1. The uncle of a block B is one less the height of B
    2. The uncle has to be in time

    "A block B1 is the ***uncle*** *of another block B2 if B1 is a competing in-time block of B2's parent block"*
- **Weight**. Since two competing chains always have a shared root, only consider blocks after that
    - weight = # of in time blocks + # of uncle hashes embedded in these blocks

PUBLISH OR PERISH

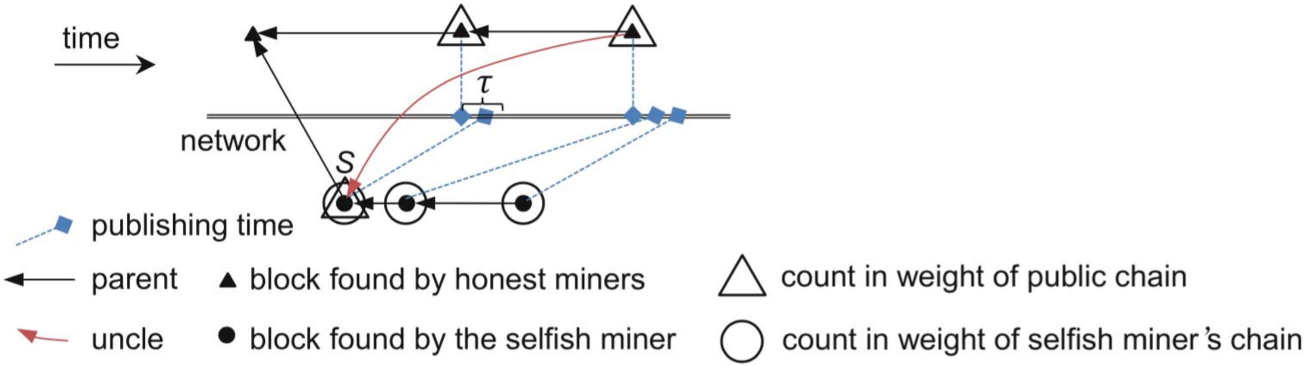
ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

Same scenario revisited. More rigorously, let S be the first selfish block

Choice 1: Selfish miner publishes S

- S will be an uncle of the next honest block
  - (since it was published *in time* and its *height is one less*)

=> S counts into the weight of **both** the honest and the selfish chain



Choice 1: Publish

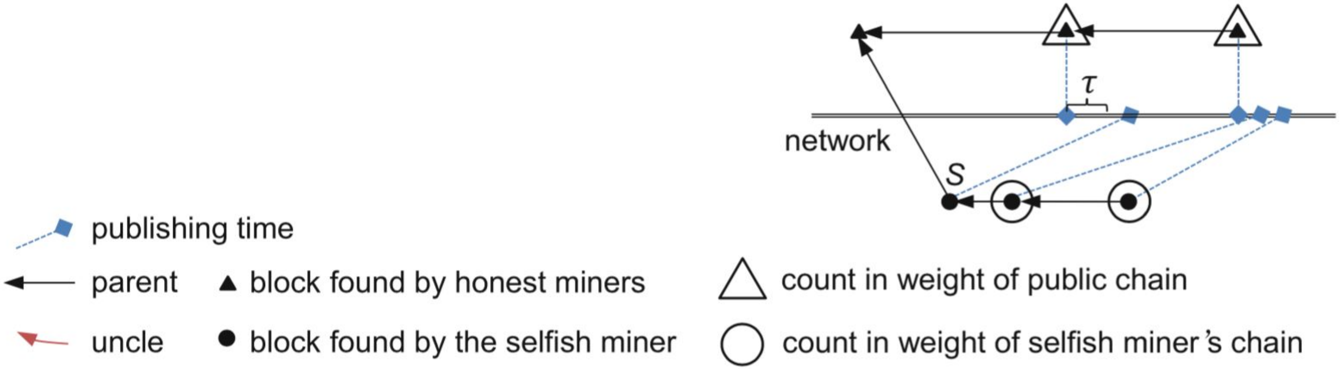
PUBLISH OR PERISH

ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

Choice 2: Selfish miner doesn't publish S

- Selfish miner waits, and publishes it later as a part of the selfish chain
- Honest miners do not count S into the weight of the selfish chain because S is not *in time.*
  - It is a **late block**
- S is not an *uncle* of the next honest block because the honest miners did not see it

=> S contributes to **neither** the weight of the honest nor the selfish chain



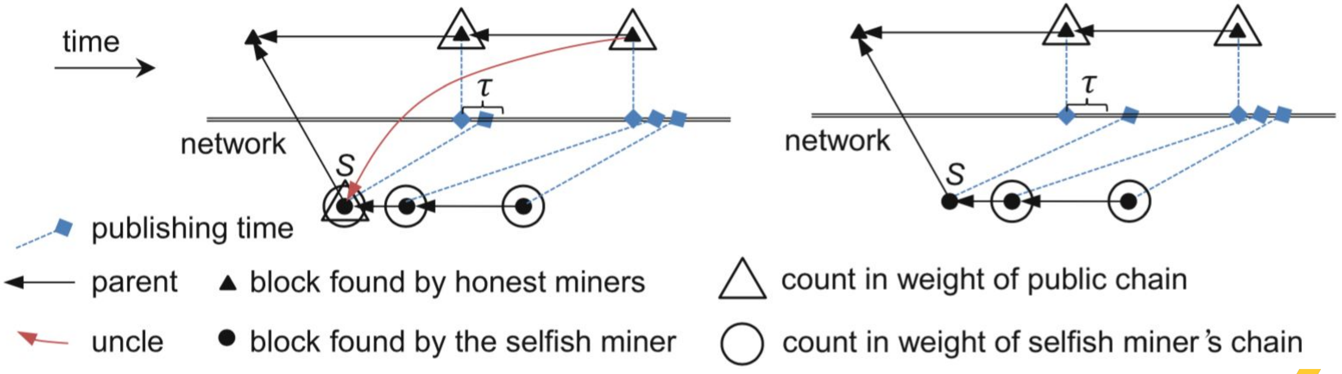
Choice 2: Don't publish

PUBLISH OR PERISH

ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

Result: Regardless of which option is chosen...

- S will **not** contribute to **only** the weight of the selfish chain.
  - Will only contribute to **both** or **neither**
- Completely nullifies the advantage of the secret block S !



Choice 1: Publish **Choice 2: Don't publish**

PUBLISH OR PERISH

ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

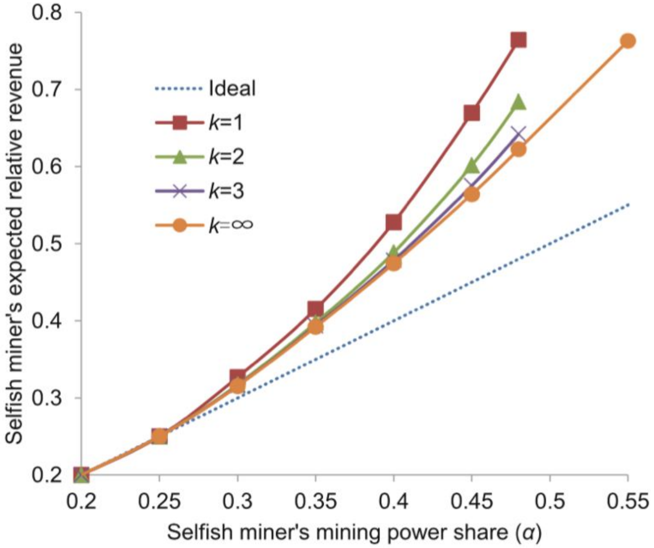


Fig. 4. Relative revenue of the selfish miner within our defense

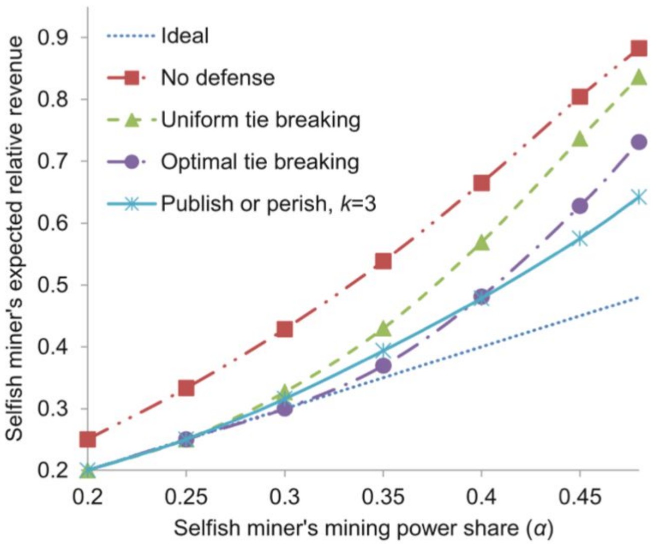


Fig. 5. Comparison with other defenses

PUBLISH OR PERISH: LIMITATIONS

ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

Limitations

- Bitcoin aims be asynchronous, Publish and Perish assumes synchronicity
  - (Because it assumes an upper bound of block propagation time)
  - Because of this, it's basically useless
- When the fail-safe parameter k > 1, an attacker may broadcast blocks right before they are late to cause inconsistent views among the honest miners
  - Several other selfish mining defenses also require a fixed upper bound on the block propagation time in order to be effective
- During the transition period to weighted FRP, an attacker can launch double-spend attacks
- Neglects real world factors:
  - Does not permit the occurrence of natural forks
  - Does not consider transaction fees on the selfish miner’s strategy
  - Does not consider how multiple selfish miners could collude and compete with each other
- Does not achieve incentive compatibility, but is the closest scheme to date

AUTHOR: MAX FANG

3 NETWORK ATTACKS AND ASICBOOST

3.1 NETWORK ATTACKS

DOS ATTACK - MALICIOUS MINERS

HONG KONG 2017

Malicious miners can **denial-of-service attack (DoS) competing miners**, effectively taking the competing miners out of the network and increasing the malicious miner's effective hash power!

Miners with access to a distributed Botnet have a competitive advantage.

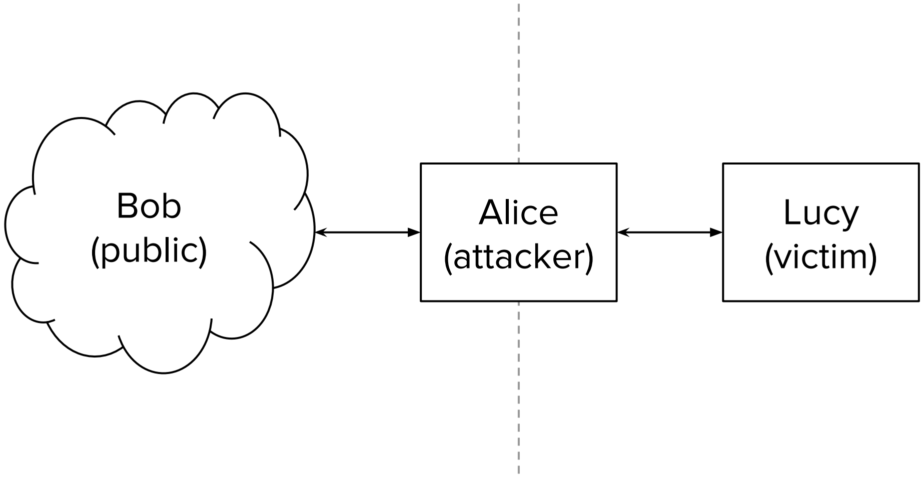
Caveat: need to maintain DoS for 2 weeks so block difficulty adjusts.

ECLIPSE ATTACK

HEILMAN ET AL.

Force network partition between public network and a specific miner

- Set up yourself, Alice, as a MITM between Bob (the public network) and Lucy (the victim miner).
- If successful, the attacker can
  - control the blocks Lucy sees
  - force Lucy to mine on Alice's chain
  - N-confirmation double spend



ECLIPSE ATTACK

HEILMAN ET AL.

Bitcoin Networking Background

- Most nodes have 8 *outgoing connections* and 117 *incoming connections*.
- **`tried` table:** stores IP address that a node has successfully made incoming or outgoing TCP connections.
- **`new` table:** stores IP addresses received from DNS seeders or ADDR messages.
    - ADDR messages contain a list of known peer IP addresses (up to 1000).

- **When a node restarts**
    - Randomly selects an IP address from either `tried` or new tables
    - If the connection succeeds, add that IP address as a new *outgoing connection*.
    - Repeat until 8 *outgoing connections*.

AUTHOR: PHILIP HAYES

ECLIPSE ATTACK

HEILMAN ET AL.

Eclipse Attack Details

1. Acquire large number of IP addresses, e.g., control a distributed botnet.
2. Fill the `tried` table with attacker-controlled addresses by having them connect to Lucy.
3. Overwrite addresses in the new table with “trash” IP addresses.
    a. Have attacker peers send unsolicited ADDR messages filled with “trash”.
    b. “Trash” addresses can be, e.g., “reserved for future use” like 252.0.0.0/8 block.
4. Force or wait for Lucy to restart.
5. With high probability, when Lucy restarts, she forms all of her outgoing connections with attacker controlled IP addresses.

3.2 ASICBOOST EXPLAINED

ASICBOOST EXPLAINED

JEREMY RUBIN

ASICBOOST Overview

- ASIC mining optimization that increases hash rate by **~20-30%**.
- Exploits reusable, partially computed, intermediate SHA256 states.
    - Precompute merkle tree permutations with the same 4-byte merkle tail

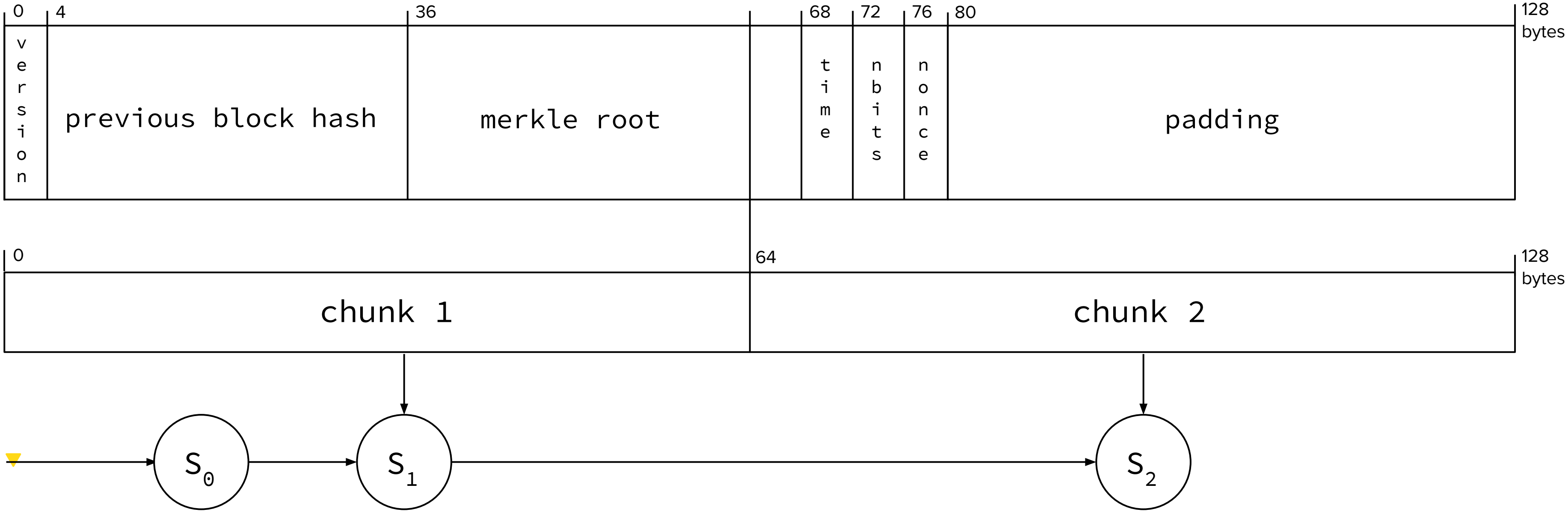
Recall: Bitcoin Block Header

0	4	36	68	72	76	80 bytes
version	previous block hash	merkle root	time	nbits	nonce	

ASICBOOST EXPLAINED

JEREMY RUBIN

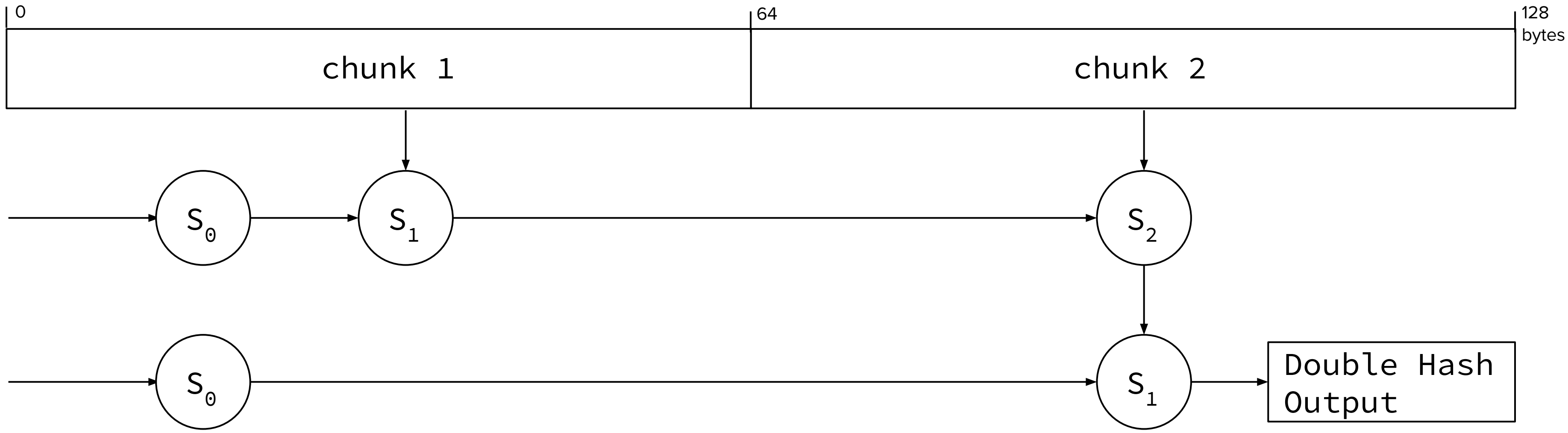
SHA-256 of Bitcoin Block Header



ASICBOOST EXPLAINED

JEREMY RUBIN

SHA-256 of Bitcoin Block Header (cont.)

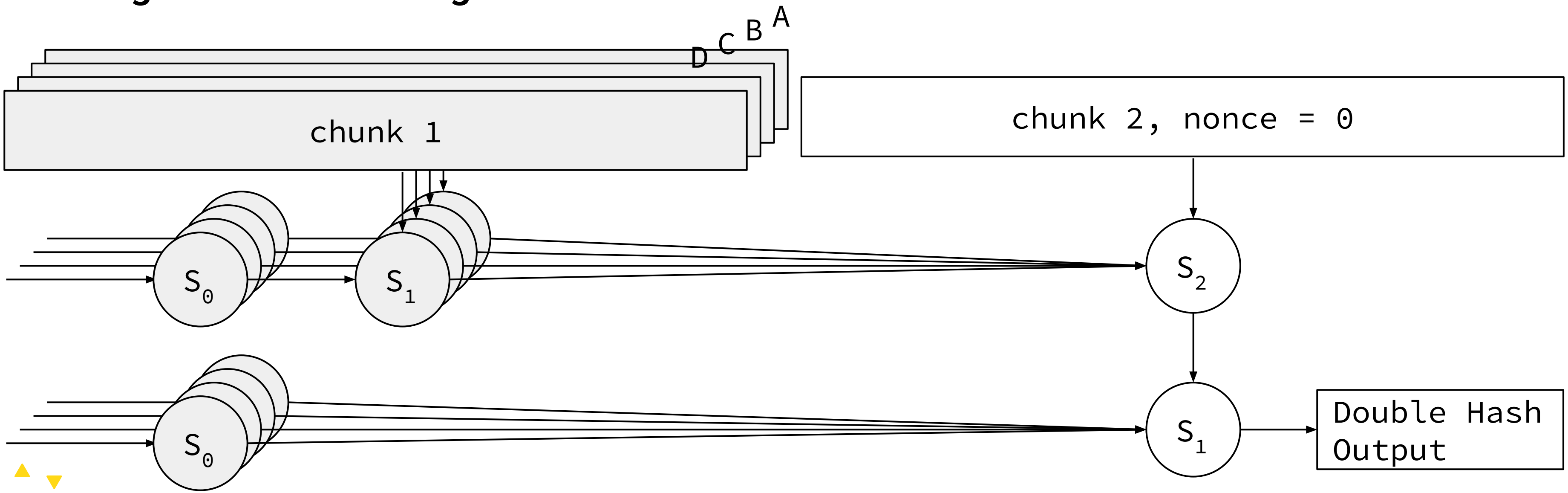


ASICBOOST EXPLAINED

JEREMY RUBIN

If we generate many block headers where the second chunk is the same, we can avoid recomputing the S_1's when we update the nonce!

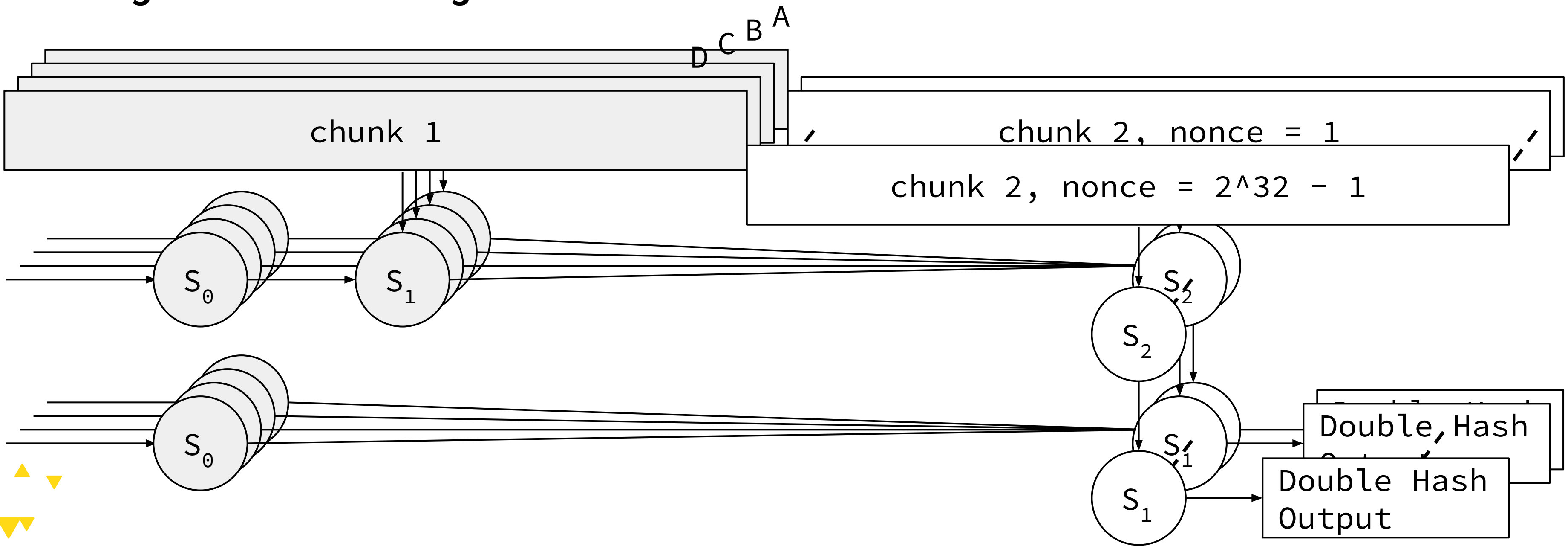
Saving work with colliding chunks



ASICBOOST EXPLAINED

JEREMY RUBIN

Saving work with colliding chunks



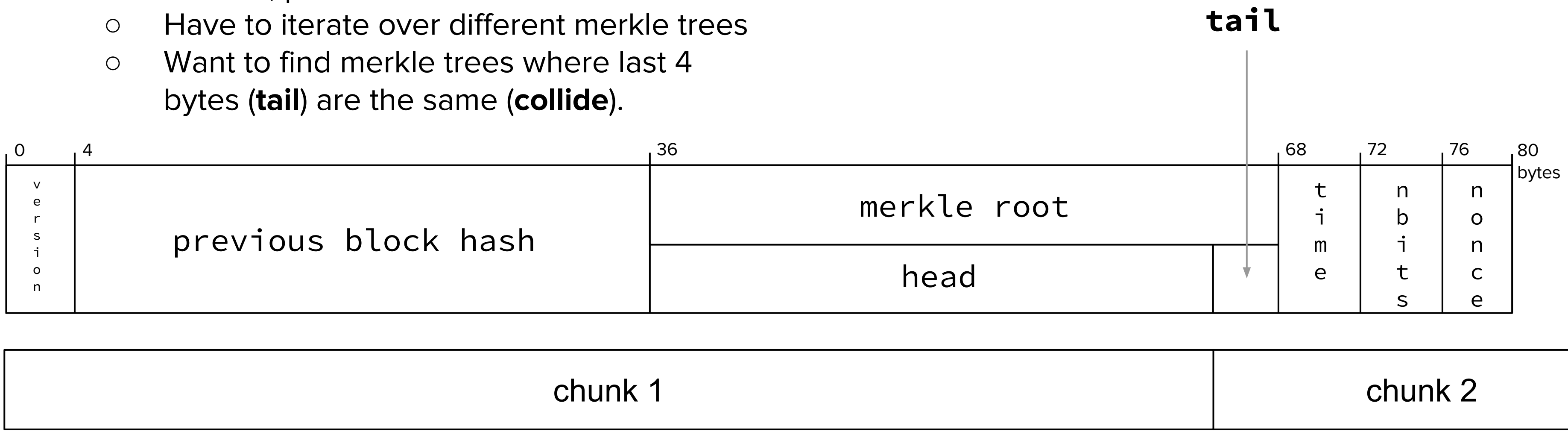
BLOCKCHAIN AT BERKELEY

ASICBOOST EXPLAINED

JEREMY RUBIN

How do we get block headers with equal chunk 2's?

- Since version, prev. block hash are fixed:
  - Have to iterate over different merkle trees
  - Want to find merkle trees where last 4 bytes (**tail**) are the same (**collide**).
-

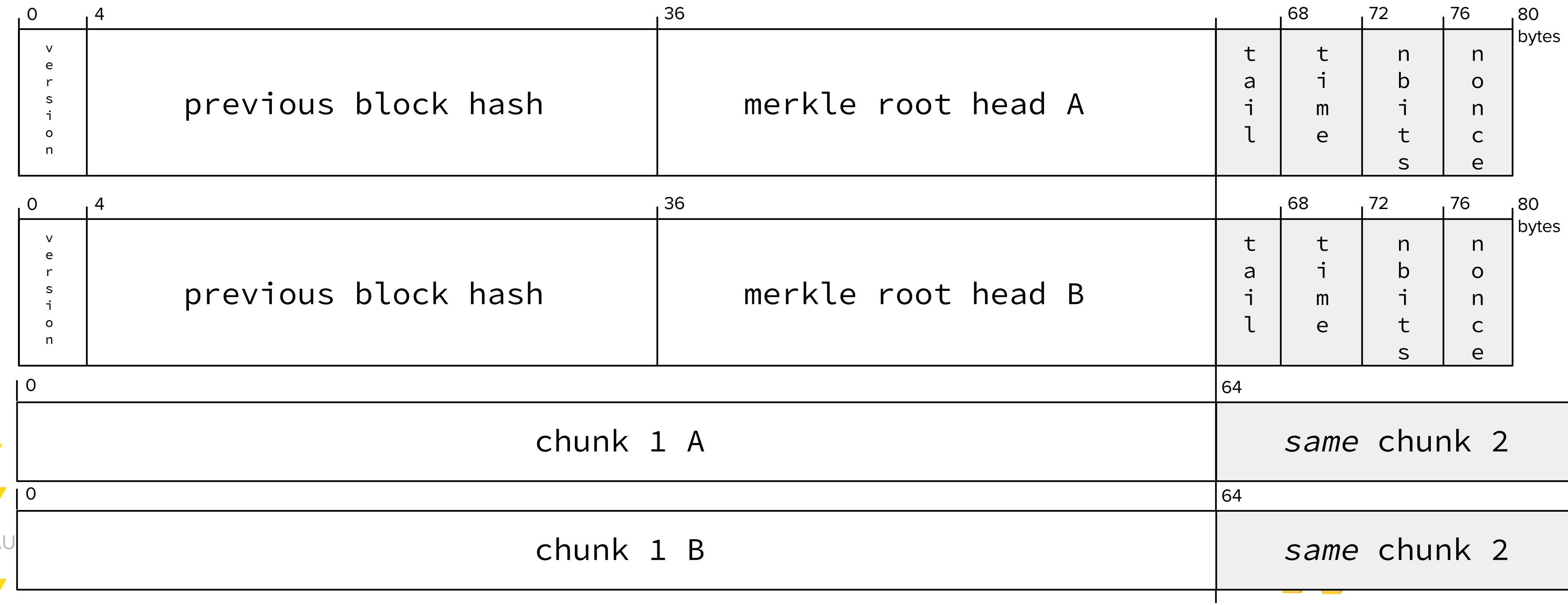


BLOCKCHAIN AT BERKELEY

ASICBOOST EXPLAINED

JEREMY RUBIN

Two Block Headers, A & B, with Colliding Merkle Tails

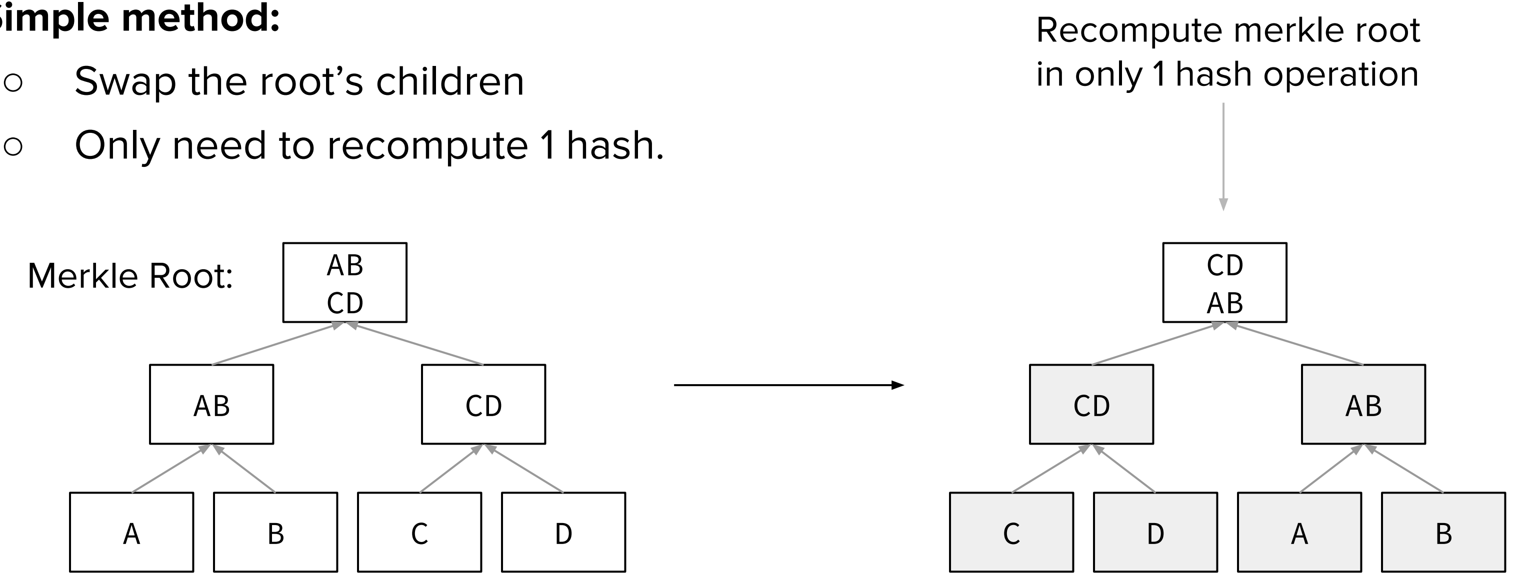


ASICBOOST EXPLAINED

JEREMY RUBIN

Iterate over Candidate Merkle Trees for the block

- **Simple method:**
  - Swap the root's children
  - Only need to recompute 1 hash.



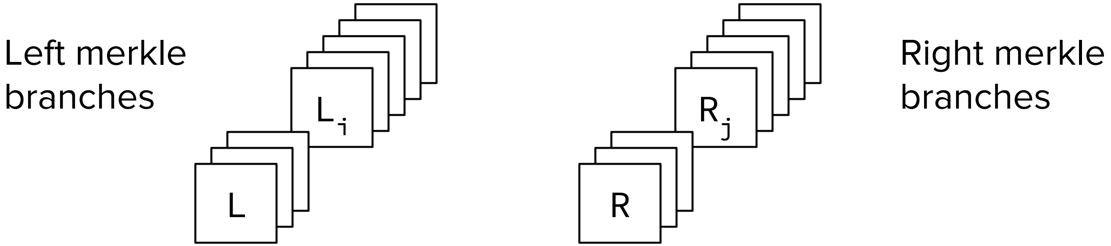
ASICBOOST EXPLAINED

JEREMY RUBIN

Iterate over Candidate Merkle Trees for the block

- **Efficient method:**
  - Generate N merkle root permutations
  - Runs in O(N) time!

1. Generate sqrt(N) left merkle branches and sqrt(N) right merkle branches.



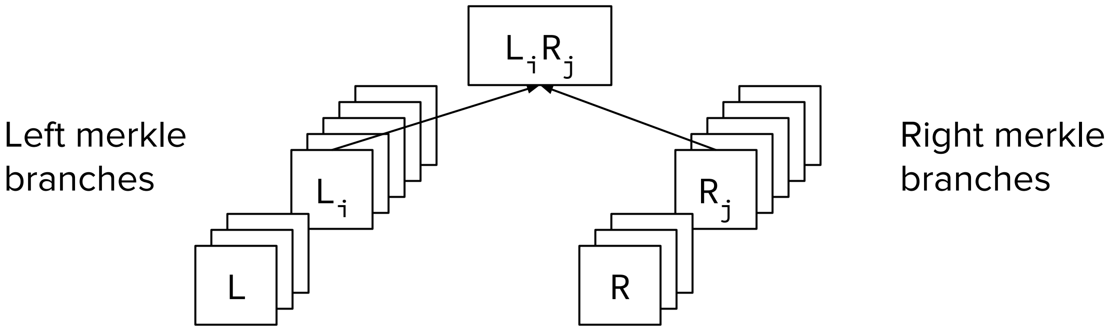
ASICBOOST EXPLAINED

JEREMY RUBIN

Iterate over Candidate Merkle Trees for the block

- **Efficient method:**
  - Generate N merkle root permutations
  - Runs in O(N) time!

1. Generate sqrt(N) left merkle branches and sqrt(N) right merkle branches.
2. Iterate over all i, j and hash the i^{th} left branch with the j^{th} right branch.



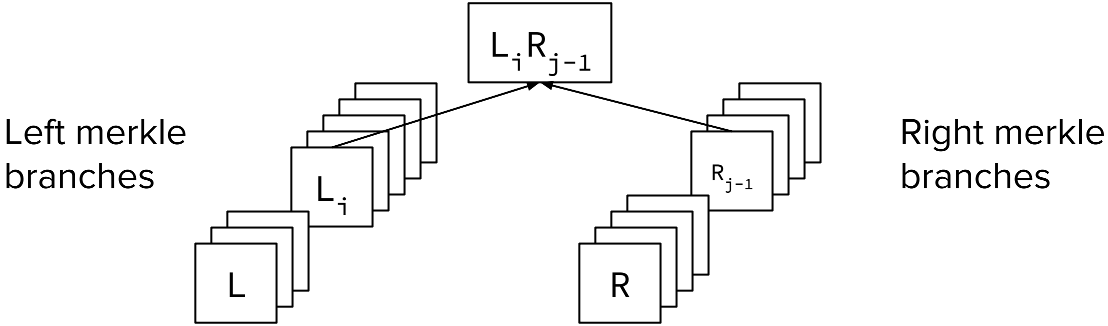
ASICBOOST EXPLAINED

JEREMY RUBIN

Iterate over Candidate Merkle Trees for the block

- **Efficient method:**
  - Generate N merkle root permutations
  - Runs in O(N) time!

1. Generate sqrt(N) left merkle branches and sqrt(N) right merkle branches.
2. Iterate over all i, j and hash the i^{th} left branch with the j^{th} right branch.



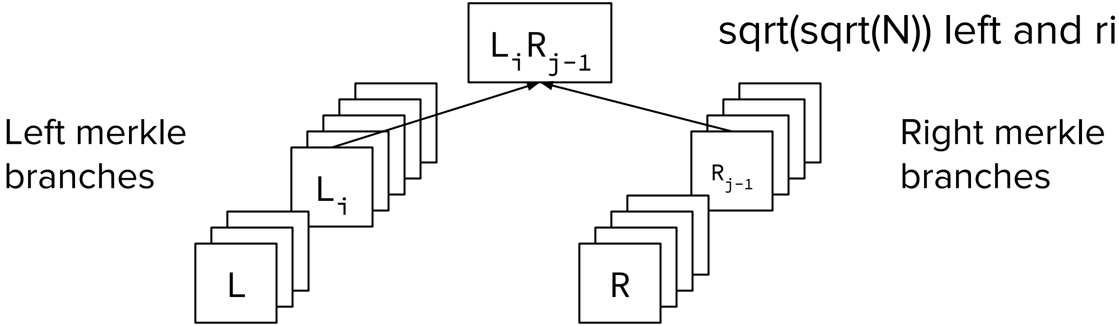
ASICBOOST EXPLAINED

JEREMY RUBIN

Iterate over Candidate Merkle Trees for the block

- **Efficient method:**
  - Generate N merkle root permutations
  - Runs in O(N) time!

1. Generate sqrt(N) left merkle branches and sqrt(N) right merkle branches.
2. Iterate over all i, j and hash the i^{th} left branch with the j^{th} right branch.
3. Apply this recursively to generate sqrt(sqrt(N)) left and right branches.



ASICBOOST EXPLAINED

JEREMY RUBIN

Finding N-Way collisions

- Instance of *Generalized Birthday Paradox*
- To find 4-Way merkle tail collision with **>50%** probability requires:
  - $N = 2^{25}$ hashes = 1 GiB of space

$$0.49 \approx 1 - e^{-\binom{2^{25}}{4}(2^{32})^{-3}}$$

- **~20-30% hash power advantage!**

How does SegWit stop ASICBOOST?

- (aka why Bitmain is acting shady)
- SegWit adds a **Witness Commitment** in coinbase of left-most merkle leaf.
- Requires miner to commit to transaction ordering.
- Can't permute merkle branches like previously described, since that changes the transaction ordering.

4

STUBBORN MINING AND ECLIPSE ATTACK COMPOSITIONS

4.1 STUBBORN MINING

STUBBORN MINING

NAYAK, KUMAR, MILLER, SHI

Generalizing selfish mining strategies

- Include block propagation race in our model.
- Extend selfish mining to include strategies that
  - withhold blocks longer
  - attempt to catch up to a longer public chain
  - are more or less risky with block propagation races

Certain strategies outperform selfish mining

- for different hash rates and block race win rates.
- Strategy metric:
  - Strategy A > Strategy B if we earn more BTC on average

FORMAL MODEL

NAYAK, KUMAR, MILLER, SHI

Main Variables

- α (Alice): Attacker's proportion of network hashrate
- **β** (Bob): Honest network hashrate
- γ: Proportion of Bob's network that will mine on Alice's block when Alice and Bob have released a block at approximately the same time, resulting in an equal length fork

PLAUSIBLE VALUES OF α AND γ

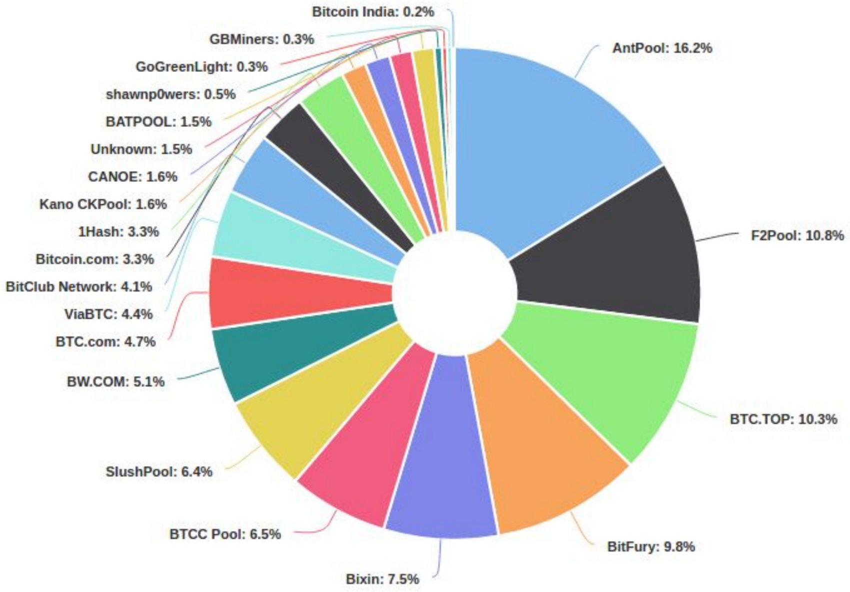
NAYAK, KUMAR, MILLER, SHI

Plausible values of α - Alice mining power:

- 37% : Three largest mining pools collude
- 26% : Two largest mining pools collude
- 16% : Largest mining pool

Not Plausible / Not Considered:

- >50% : Consensus breaks down anyway
    - mining strategy not relevant
    - can mine every block
    - private fork will always outpace public network given enough time



BLOCKCHAIN AT BERKELEY

PLAUSIBLE VALUES OF α AND γ

NAYAK, KUMAR, MILLER, SHI

Plausible values of γ:

- γ = Proportion of network that mines on our block in a propagation race.
- Depends on:
  - Latency between Alice and other nodes
  - Latency between other miner and other nodes
  - Network topology, e.g., how well-connected is Alice; large miners usually well-connected.
- ⇒ Consider all values γ in [0, 1]

Alice can actively attack network to improve γ:

- Monopolize other nodes' incoming connections
  - Bitcoin nodes have a finite number of incoming TCP connections
  - Alice can spawn sybils to fill up available incoming connection slots.

Public Network Defenses :

- Miner backbone, e.g., FIBRE block relay

FORMAL MODEL

NAYAK, KUMAR, MILLER, SHI

Stubborn Mining Strategies

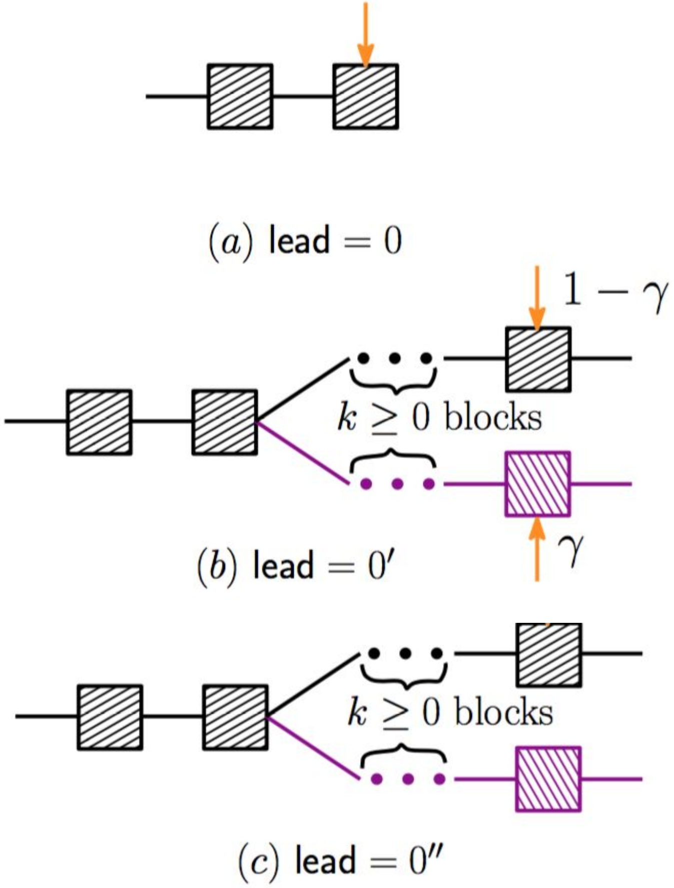
- H: Honest strategy
- S: Selfish mining strategy
- L-stubborn: Lead stubbornness
  - Does not broadcast blocks even with a high lead over the public
- F-stubborn: Fork stubbornness
  - Will not give up during an equal fork
- T_j-stubborn: Trail stubbornness
  - Does not merge with public unless it is trailing the public by more than j blocks

FORMAL MODEL

NAYAK, KUMAR, MILLER, SHI

Markov Chain State Representation

- ***lead***: how much Alice's chain is ahead of Bob's. Is some integer ***N***
- ***N'***: There is a fork, and
  - The revealed portion of the fork is of equal length
  - Bob's mining power is split on this fork according to γ
- ***N''***: Same as ***N'***, but all of Bob's mining power is on their own fork (i.e. $\gamma = 0$)



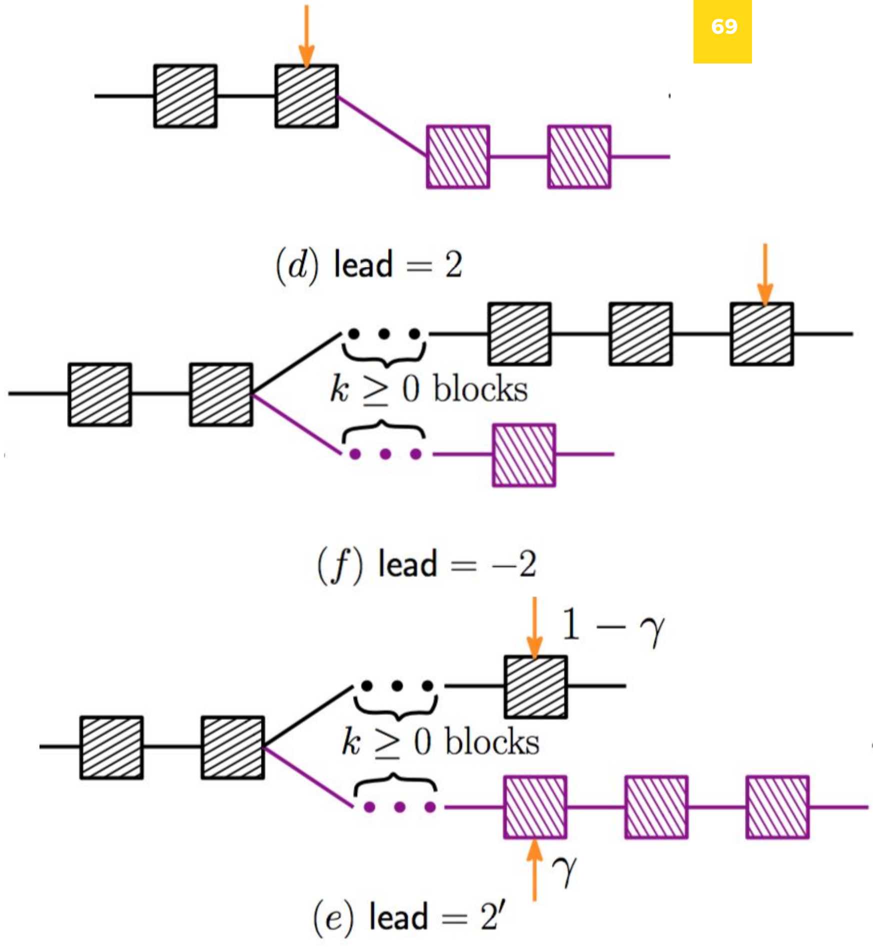
BLOCKCHAIN AT BERKELEY

FORMAL MODEL

NAYAK, KUMAR, MILLER, SHI

Markov Chain State Representation

- ***lead***: how much Alice's chain is ahead of Bob's. Is some integer ***N***
- ***N'***: There is a fork, and
  - The revealed portion of the fork is of equal length
  - Bob's mining power is split on this fork according to γ
- ***N''***: Same as ***N'***, but all of Bob's mining power is on their own fork (i.e. γ = 0)



BLOCKCHAIN AT BERKELEY

EXAMPLE: SELFISH MINING

NAYAK, KUMAR, MILLER, SHI

A mining strategy consists of two decisions:

1. When to broadcast your private chain
2. When to mine off the public chain's head

Honest mining: Always broadcast as soon as block is found, and always accept longest chain

Selfish mining

- When *lead* = 2 and Bob mines the next block: Reveal Alice's entire private chain to Bob (resulting in *lead* = 0)
- When *lead* = 0' and Alice mines the next block: Reveal Alice's private chain to Bob (resulting in lead = 0).
- • When *lead* = 0 or *lead* > 0 and Alice mines the next block: Do not reveal Alice's private chain
- Like in honest mining, Alice always accepts the longest chain.

SELFISH MINING: MARKOV REPRESENTATION

NAYAK, KUMAR, MILLER, SHI

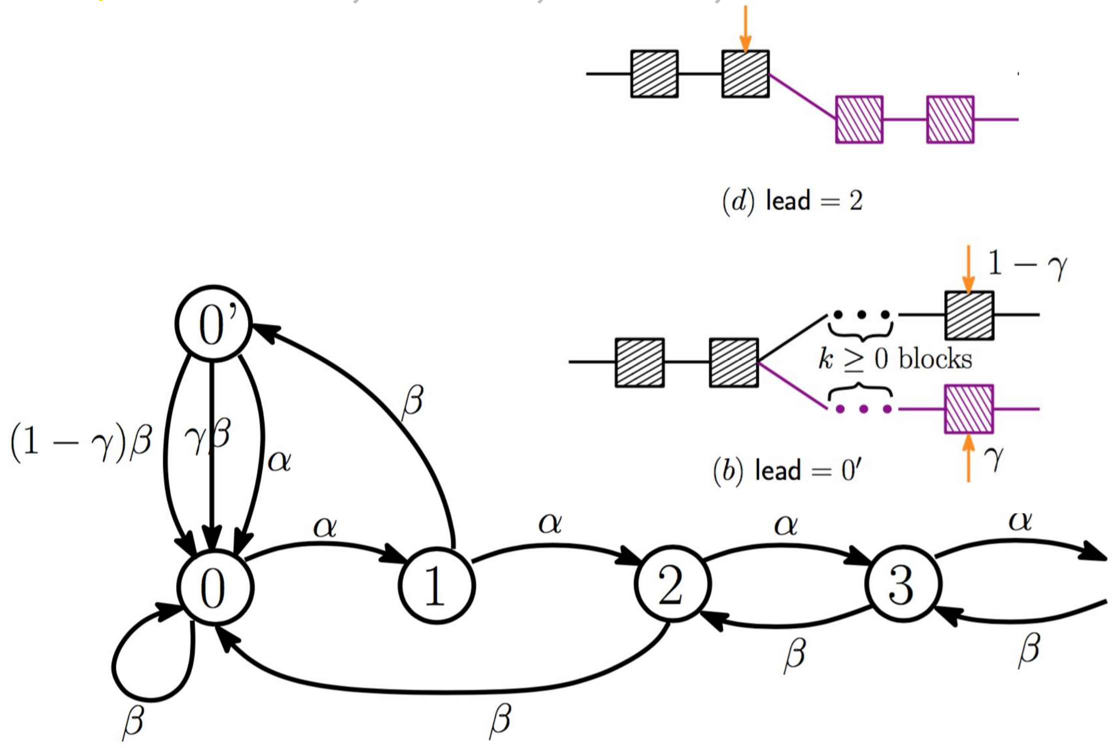


Figure 2: **Selfish Mining [9].**

Selfish mining

- When *lead* = 2 and Bob mines the next block: Reveal Alice's entire private chain to Bob (resulting in *lead* = 0)
- When *lead* = 0' and Alice mines the next block: Reveal Alice's private chain to Bob (resulting in lead = 0).
- When *lead* = 0 or *lead* > 0 and Alice mines the next block: Do not reveal Alice's private chain
- Like in honest mining, Alice always accepts the longest chain.

LEAD STUBBORN STRATEGY

NAYAK, KUMAR, MILLER, SHI

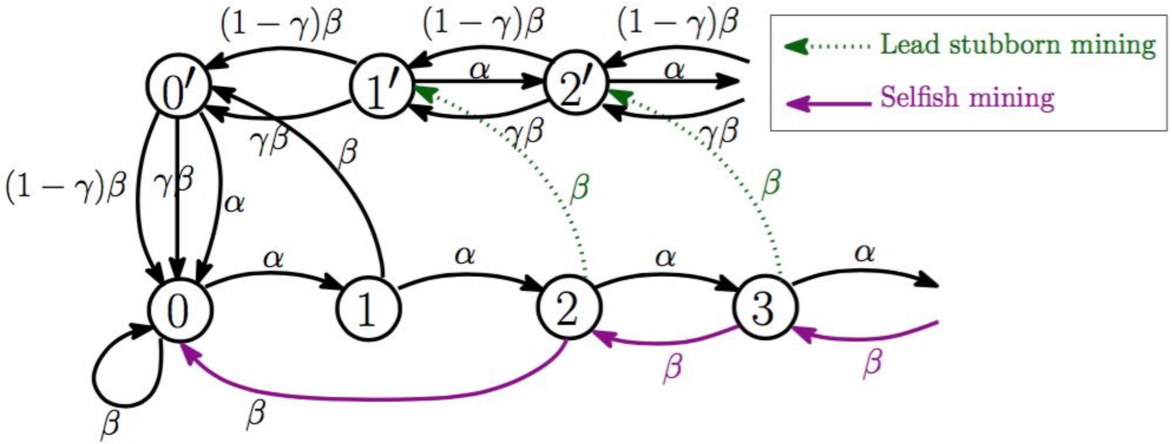
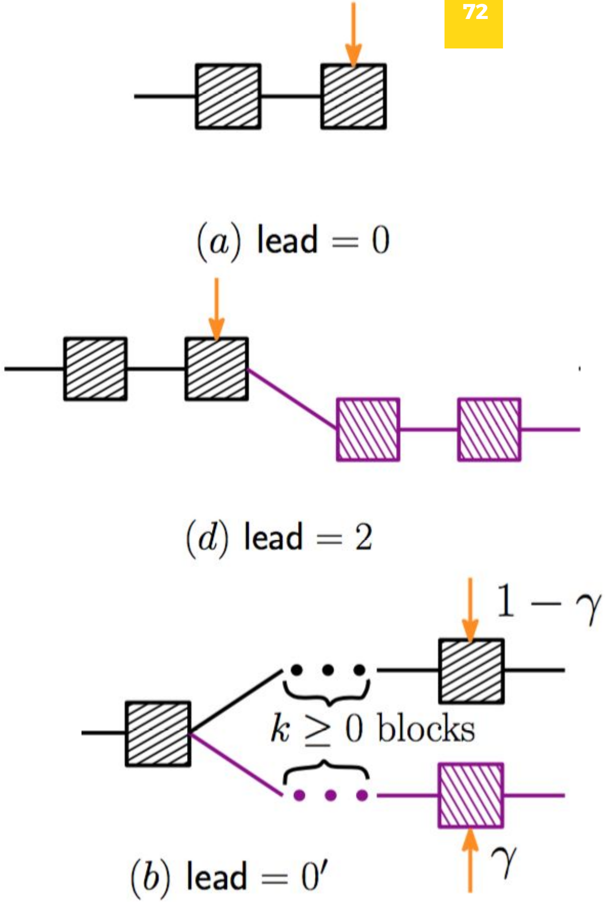
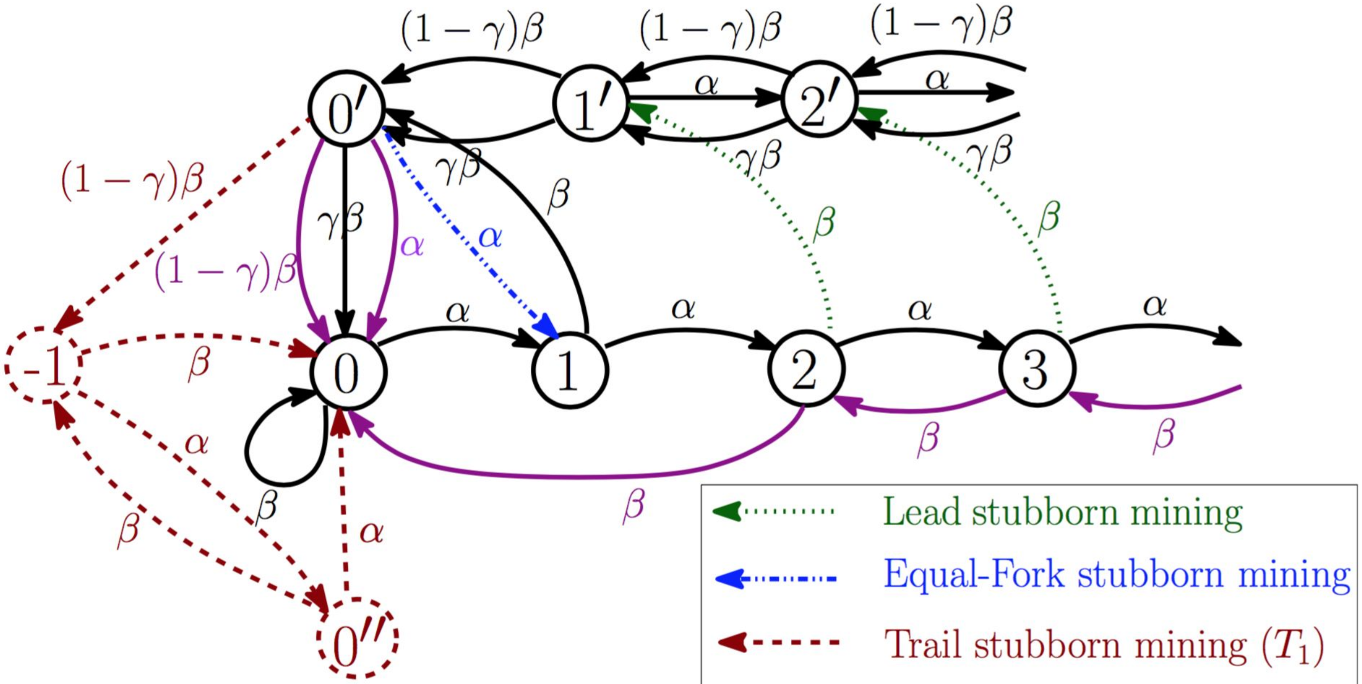


Figure 4: **Lead-stubborn mining.** Black + magenta transitions define selfish mining. Black + green transitions define lead-stubborn mining. Markov chain states are defined in Section 2.2 and Figure 1.



TRAIL STUBBORN, EQUAL FORK STUBBORN

NAYAK, KUMAR, MILLER, SHI



PROFITABILITY CALCULATION

NAYAK, KUMAR, MILLER, SHI

Profitability measurements

- Profitability (for Alice) is measured as relative gain of Alice compared to Bob

$$\text{relative\_gain}(X, Y) = \frac{\text{gain}_X - \text{gain}_Y}{\alpha}$$

gain_X is proportion of blocks earned by Alice under strategy X.

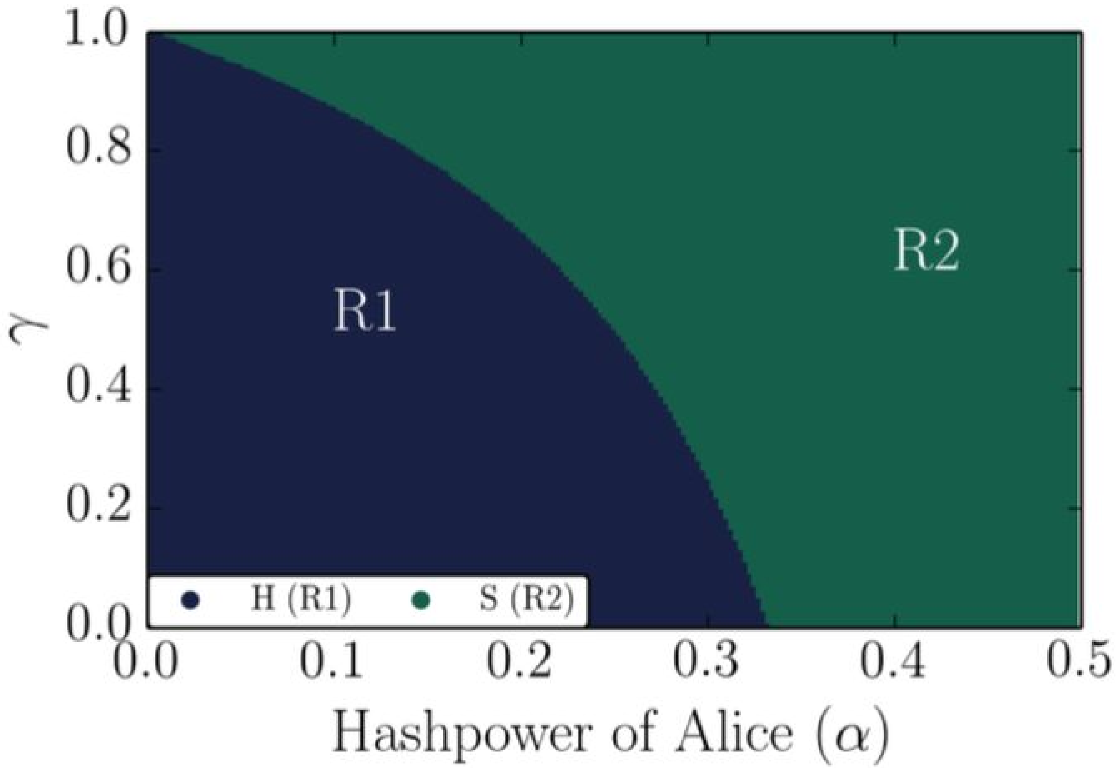
$$\text{relative\_gain}(SM, H) = \frac{\text{gain}_{SM} - \alpha}{\alpha}$$

Gains are normalized w.r.t. $\boldsymbol{\alpha}$

- $\boldsymbol{\alpha}$ is what Alice would have received honestly

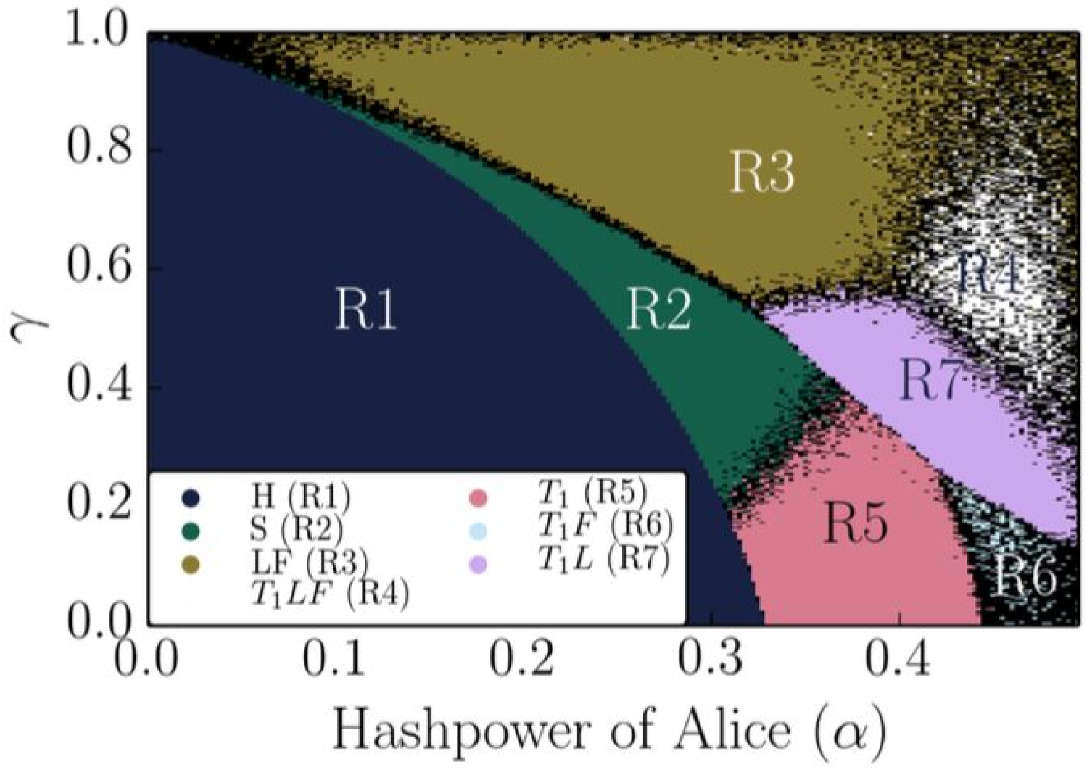
DOMINANT STRATEGY SPACE

NAYAK, KUMAR, MILLER, SHI



DOMINANT STRATEGY SPACE

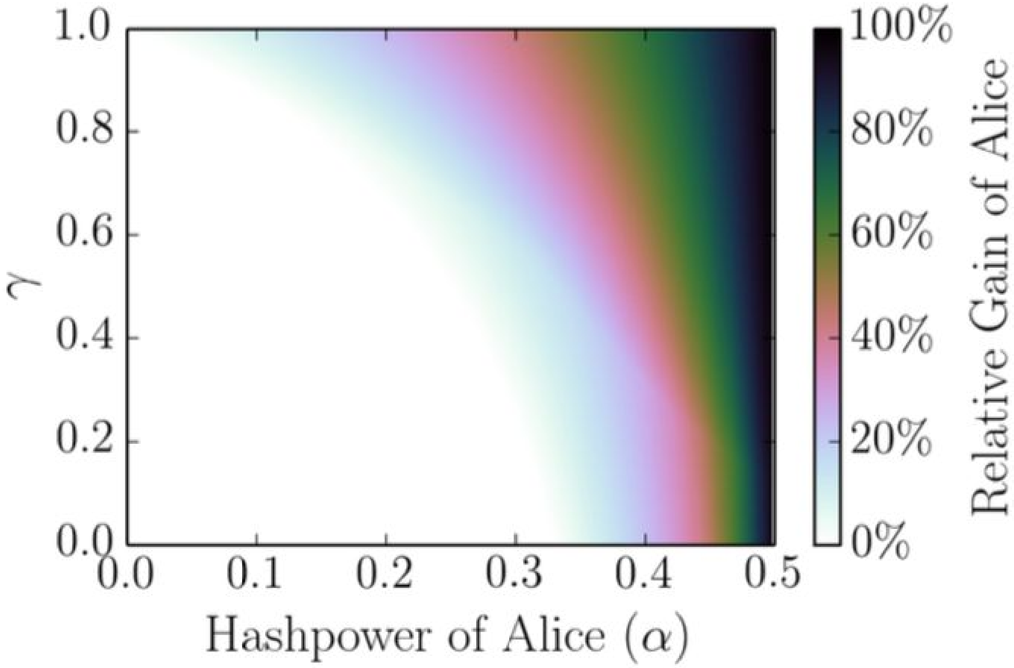
NAYAK, KUMAR, MILLER, SHI



BLOCKCHAIN AT BERKELEY

RELATIVE PROFITABILITY

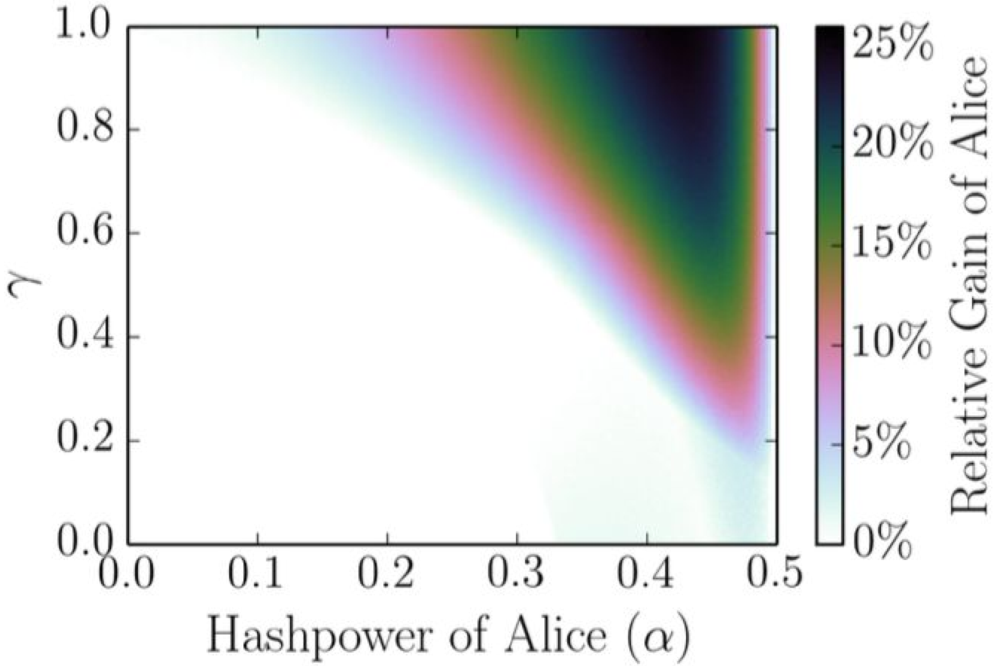
NAYAK, KUMAR, MILLER, SHI



(a) Compared to honest mining.

RELATIVE PROFITABILITY

NAYAK, KUMAR, MILLER, SHI



(b) Compared to selfish mining.

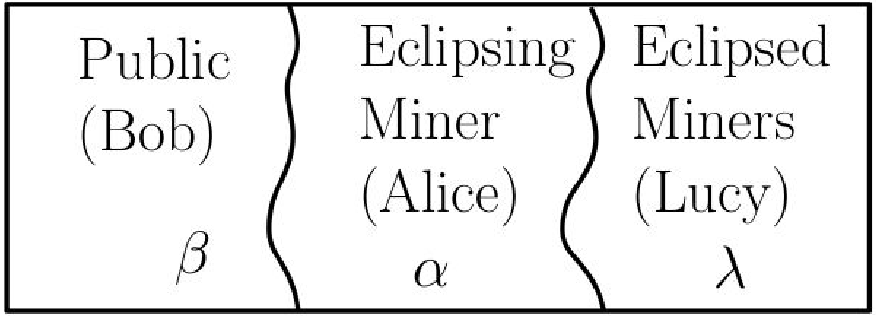
4.3 ECLIPSE ATTACK COMPOSITIONS

NAIVE ECLIPSE STRATEGY

NAYAK, KUMAR, MILLER, SHI

Combining Stubborn Mining and Eclipse Attacks

- λ : Lucy's hash power = the eclipsed miners' hash power
- *D : Destroy the Eclipsed Victim*
    - Alice ignores all of Lucy's mined blocks, effectively removing Lucy from the network
- *C : Collude with the Eclipsed Victim*
    - Alice forces Lucy to mine on Alice's private chain by feeding Lucy only Alice's blocks.

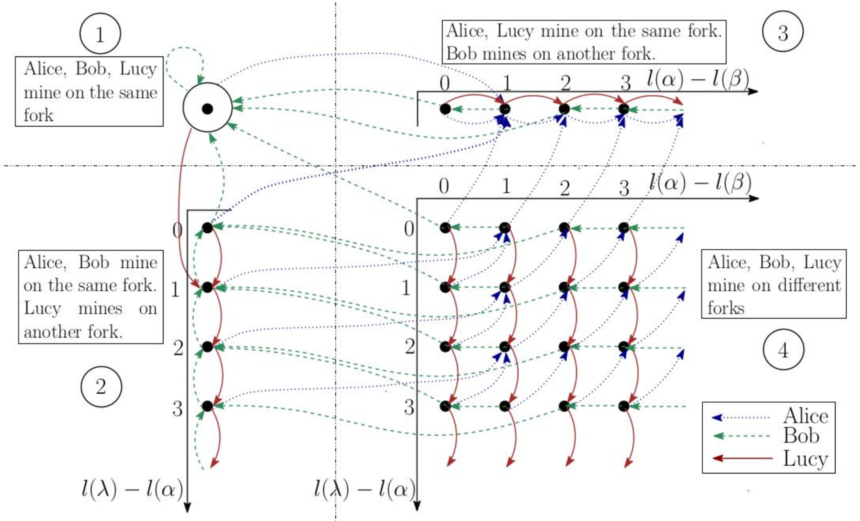


SOPHISTICATED ECLIPSE STRATEGY

NAYAK, KUMAR, MILLER, SHI

More Sophisticated Eclipse Strategies

- *DNS : Destroy if No Stake*
  - Only broadcast Lucy's blocks if she's mining on our private chain.
  - Otherwise, if Lucy finds a block on the main chain (lead = 0), we ignore her blocks

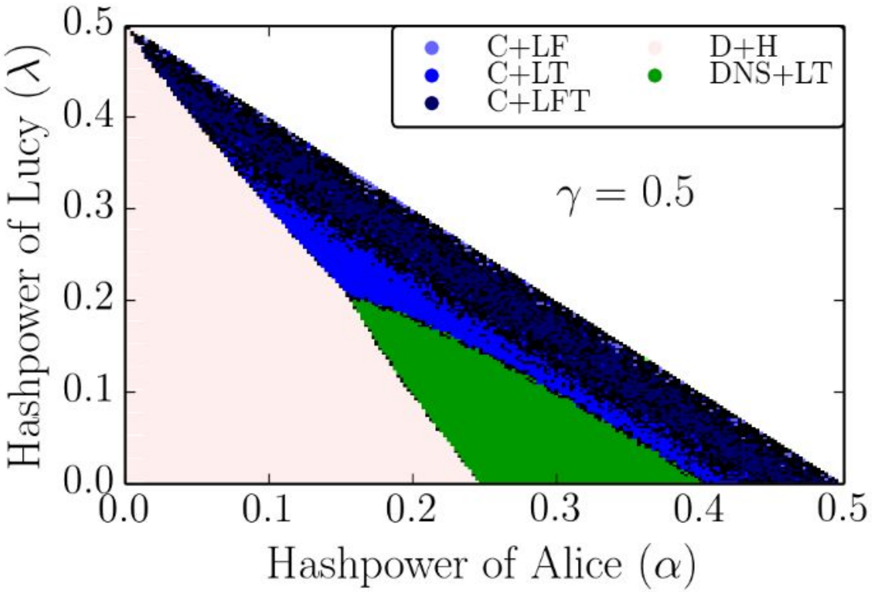


SOPHISTICATED ECLIPSE STRATEGY

NAYAK, KUMAR, MILLER, SHI

Optimal Strategy Space

- In some cases, Lucy is actually incentivized to collude with Alice!



(c) $\gamma = 0.50$

5 COLLUSIONS AND CONCLUSIONS

POST-BLOCK REWARD BITCOIN

MARTIN KOPPELMANN, SF BITCOIN DEVS

Assumption: average Bitcoin user holds $100,000 in Bitcoin, willing to pay $1000 in fees

- (This is when Bitcoin is near 0 block reward)
- Is mining based off transaction fees sustainable?
- Money must move, must be paid in transaction fees so that miners can collect it as mining reward
- Amount of hashpower going into Bitcoin dependent on mining reward

Therefore

- (average fees paid) / (avg holdings) = (network fees paid) / (market cap) = (cost of attacking) / (market cap)
- In our example, attacker only needs to pay 1% of the market cap of Bitcoin to gain 50% of the hashrate
  - Since that is the amount of money going into mining
- More realistic scenario: Attacker only needs 0.1% of market cap to attack

Post reward Bitcoin must have a high velocity of money to be secure

LEMMAS 1 & 2

MARTIN KOPPELMANN, SF BITCOIN DEVS

Computational power requires electricity, which, requires $$, reaches equilibrium if miners are breaking even or profitable

- Lemma 1: **Mining Reward = Mining Cost**

If you are roughly breaking even with the capital you invest, there is little to no marginal cost to getting more hashrate. You simply need more capital to attain 51%

- Lemma 2: **Cost of acquiring 51% ≈ 0 < Mining cost**

LEMMA 3

MARTIN KOPPELMANN, SF BITCOIN DEVS

What profits can you get from owning >51% of the hashrate

- Crash the currency? No problem. Regain value (and then some) by shorting Bitcoin on an exchange

You can effectively get 100% of the mining reward

- Only mine on your own blocks
    - Can prevent anyone else from mining - you always produce longest PoW chain

How this would affect the price depends on threshold

- q = 51% => 49% of blocks are orphaned (eh)
- q = 80% => 20% of blocks are orphaned
    - Average Bitcoin user not really affected, still able to make transactions

Lemma 3: **Value of 51% attack > Mining Reward**

LEMMA COMBINATION AND CONCLUSION

MARTIN KOPPELMANN, SF BITCOIN DEVS

Lemmas:

- Lemma 1: Mining Reward = Mining Cost
- Lemma 2: Cost of acquiring 51% < Mining Cost
- Lemma 3: Value of 51% attack > Mining Reward

Therefore, Value of 51% attack > Cost of acquiring 51%

If math is correct, Game Theory says that 51% attacking Bitcoin **is profitable**

(Originally presented by Martin Koppelmann at SF Bitcoin Devs Seminar)

5.1 COLLUSIONS

POOL COLLUSIONS

MARTIN KOPPELMANN, SF BITCOIN DEVS

Bitcoin mining is zero sum

- In general, to increase earnings, someone else needs to be excluded

Members-only Mining

- Let hashrate join a collusion until 80% of the network is in, then exclude the rest
- No incentive not to join
  - Attack succeeds, get increased reward
  - Attack wouldn't fail: conduct attack in such a way that it wouldn't start until the threshold is reached
- Therefore Game Theory dictates that this would always happen

Naive Example

- 3 pools collude, own more than 51%
- Ignore every 10th block of another pool
- How to detect?

More profitable than honest strategy

Thought: How many of these are going on today?

5.2 CONCLUSIONS

DESTROYING MINING POOLS

MARTIN KOPPELMANN, SF BITCOIN DEVS

Pool Block-Withholding: Countermeasures?

- Want: A way to offset costs incurred by pool wars
- Idea: More orphaned blocks are a sign of pool wars
- Martin Koppelmann proposes **insurance contracts** that pay out to bitcoin stakeholders based on number of orphaned blocks

What are the implications?

- Yaron Velner, Jason Teutsch, and Loi Luu explore this concept in detail
- "Smart Contracts Make Bitcoin Mining Pools Vulnerable" (2017)

DESTROYING MINING POOLS

VELNER, TEUTSCH, LUU (2017)

"Insurance contracts" incentivizes pool block-withholding

- Recall: Pay-per-share mining pool reward scheme pays per every partial hash solution found
  - No incentive to submit valid blocks to the pool beyond the reward for that share
- Incentivizing someone from submitting a valid block only requires offsetting the cost of a single share
  - Called the **block purchasing budget**

Let's trustlessly (and anonymously) set up a way for someone to receive a reward for withholding a block from a pool. Sketch:

- Set up a contract on Ethereum, which has a scripting language rich enough to parse Bitcoin blocks
- Reward a tuple (b_1, b_2, b_2', b_3) where
  - b_1, b_2, b_2', b_3 are block headers; and
  - b_2 and b_2' both extend b_1; and
  - b_3 extends only b_2.

(b'$_2$ is the withheld block)

This **proof-of-stale-work** can even be targeted at a specific pool

DESTROYING MINING POOLS

VELNER, TEUTSCH, LUU (2017)

Velner, Teutsch, and Luu do some napkin math:

- **D** the difficulty of the entire Bitcoin network
- **d** the difficulty of a single share. **s = d / D**
- **r** be the block reward,

The reward per share is thus **r * (d / D)**

- Highest recommended **d** = 4,096. Network parameter **D** > 2.53 x 10^9 (Nov 2016)

$$(5) \qquad s = \frac{d}{D} \approx 2 \cdot 10^{-8},$$

$$r \cdot s \approx (12.5 \text{ btc}) \cdot s = 2.5 \cdot 10^{-7}$$

Result: Reward from a single share (in Nov 2016) is 2.5×10^{-7} = **$0.02 per share**

DESTROYING MINING POOLS

VELNER, TEUTSCH, LUU (2017)

Some more napkin math:

- α attacker's mining power, as a fraction of the network
- β the proportion of blocks that are withheld from the network
- If attacker manages to discard β fraction of blocks, their effective hashrate in the network is

  a = α **/ (1 -** β**)**
- Attacker's additional revenue from destroying fraction of blocks is

$$a \cdot r - \alpha \cdot r = \frac{\alpha\beta \cdot r}{1-\beta}. \qquad (2)$$

- When β fraction of miners don't submit valid blocks, network difficulty decreases by multiplicative factor **(1 -** β**)** and expect to find β **/ (1 -** β**)** blocks.
- Expected revenue is thus **s** * **r** times the above quantity, yielding revenue

$$\frac{\beta \cdot s \cdot r}{1-\beta} \qquad (3)$$

Combining (2) and (3), both attacker and participating miners are profitable if

$$\alpha > s.$$

DESTROYING MINING POOLS

VELNER, TEUTSCH, LUU (2017)

Both attacker and participating miners are profitable if

$$\alpha > s.$$

What is **s**? From our previous calculations,

$$(5) \quad s = \frac{d}{D} \approx 2 \cdot 10^{-8},$$

which is **1/50,000,000** of the network or **0.000002%**

0.000002% of the network is equivalent to **4 TH/s** mining power

- Less than a single ASIC

Conclusion:

- With a single ASIC worth of mining power, an attacker can incentivize all miners a given PPS mining pool to cannibalize the pool
  - Attacker even profits!
- Theoretical result: Easily destroy all mining pools this way with one ASIC

But how realistic is this **really**?

CAVEATS TO RATIONALITY ASSUMPTIONS

HONG KONG 2017

Practical caveats to game-theoretical attacks

- Attack requires significant risk or capital
  - Poor game-theoretical assumption
- Hard to write and deploy custom exploitative software
- Insufficiently motivated attackers
- Miners may support Bitcoin
- Social costs: vigilante attackers
  - Lack of anonymity

CONCLUSIONS

HONG KONG 2017

The world is not rational!

- We're bound by the paradox of choice
- Rationality assumptions conjecture no need for the work of academics
    - We would have exploited these attacks in practice already!

Conclusion

- Bitcoin is not game-theoretically secure
    - Something else is keeping Bitcoin alive
    - More emphasis should be placed on behavioral economics, psychology, and sociology

RELATED WORK NOT COVERED

HONG KONG 2017

Additional great game theoretical work:

- Joseph Bonneau paper on in-band double spend bribery
  - Whale transactions
- (Claimed) Incentive compatible PoW blockchains
  - Fruitchains (2016) by Elaine Shi
  - Meshcash and some more
- In-depth Post Block-Reward analysis
  - Narayanan et al 2016: "On the Instability of Bitcoin Without the Block Reward"
- Vlad Zamfir on Casper

X

EXTRA CONTENT

PUBLISH OR PERISH

ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

•

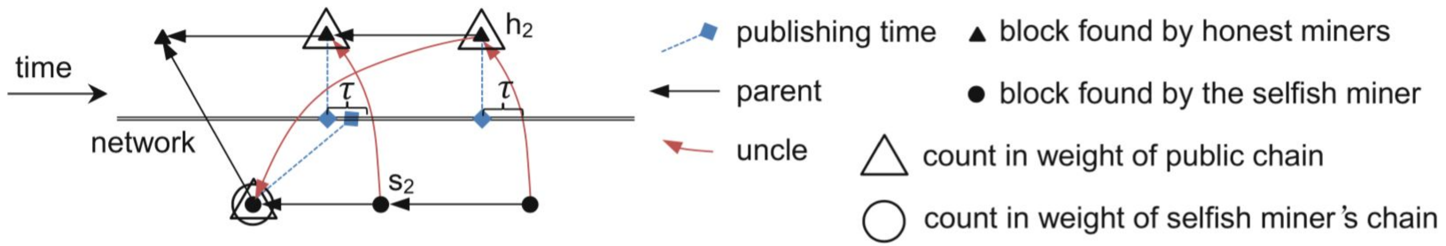


Fig. 2. A block race. Selfish block s_2 is not published in time after its competitor h_2 is mined, thus becomes a late block.

IN-BAND SMART CONTRACT BRIBERY

BONNEAU, "WHY BUY WHEN YOU CAN RENT?"

DOUBLE SPENDS: WHALE TRANSACTIONS

HONG KONG 2017

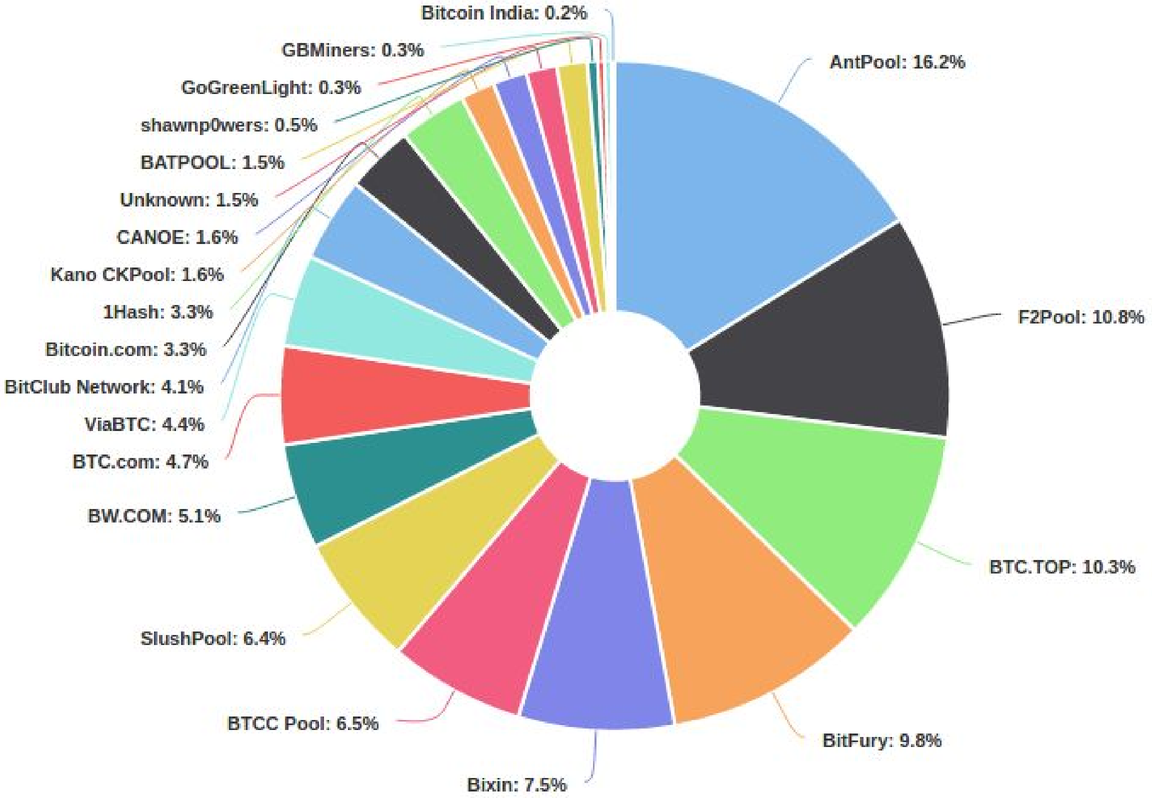
MINING POOL HASHRATE DISTRIBUTION

Community exhibits backlash against large mining pools

- Ex: GHash.io in 2014

Single entity might be be participating in multiple pools

- Called "**Laundering hashes**"
- Actual concentration of control over mining hardware **is unknown**



Source:
blockchain.info/pools (2017-05-15)

AUTHOR: MAX FANG

BLOCKCHAIN AT BERKELEY

MINING POOLS - SHARES

HONG KONG 2017

How to prove that you are contributing to the pool?

Submit **Shares**: 'Near-valid' blocks

- Producing shares implies computational power being expended
- Pool operator pays for valid shares
  - Rewards distributed proportional to # of shares submitted
- Valid blocks are shares as well
  - Individual who finds valid block is not rewarded any extra coins

FAQ: Why can't someone submit shares in a pool and keep the reward of the valid block for themselves?

- The valid block is based on the Merkle root given by the pool operator.
- Pool public key ➞ Coinbase tx ➞ Merkle Root

AUTHOR: MAX FANG

MINING POOLS - BASIC REWARD SCHEMES

HONG KONG 2017

Pay-per-share

Pool pays out **at every share submitted.** By default will be proportional to work done by individuals

1. More beneficial for **miners**
2. Individual miners have no risk from reward variance
   a. Pool takes on the risk completely
3. Problem: No incentive for individuals to actually submit valid blocks
   a. Individuals are paid regardless

Proportional

Pool pays out **when blocks are found,** proportional to the work individuals have submitted for this block

1. More beneficial for the **pool**
2. Individual miners still bear some risk in variance proportional to size of the pool
   a. Not a problem if pool is sufficiently large
3. Lower risk for pool operators - only pay out when reward is found
   a. Individuals thus incentivized to submit valid blocks

POOL HOPPING

HONG KONG 2017

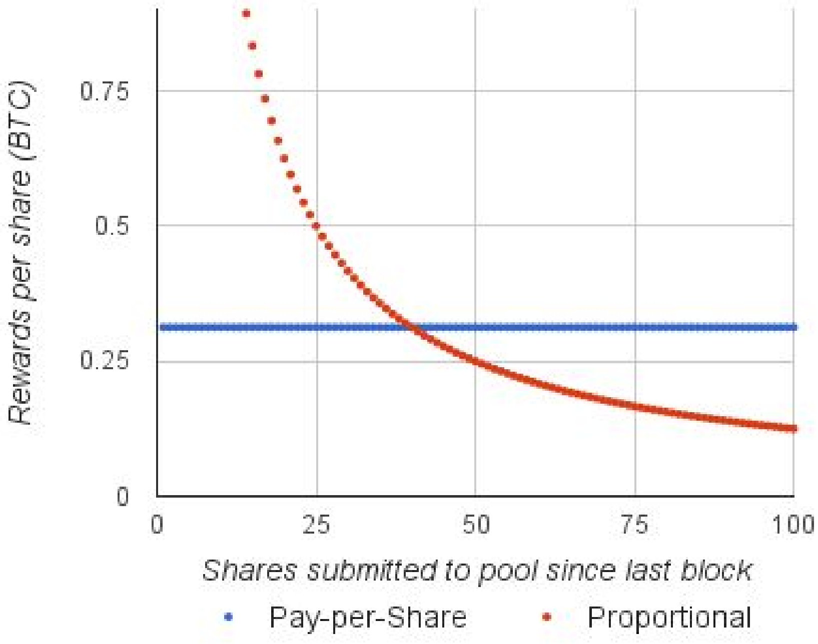
Pool hopping: switching between pools to increase total rewards

- Proportional pool pays larger amount per share if a block is found quickly

Example clever strategy:

- Mine at **proportional** pool shortly after a block was found (while rewards are high)
- Switch to **pay-per-share** pool when once proportional pool is less profitable

Rewards per share for Pay-Per-Share and Proportional Pools



Parameters:

- Pool has 10% of network hashrate
- 4 shares expected per valid block

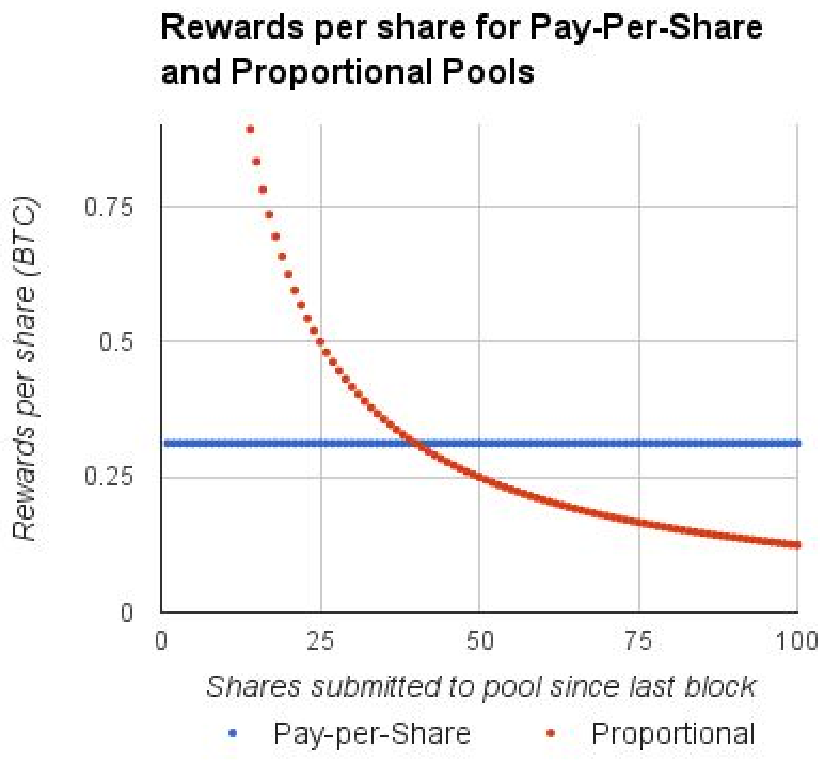
POOL HOPPING

HONG KONG 2017

Therefore, proportional pools are **not feasible in practice**

- Honest miners who stay loyal to one pool are cheated out of their money

Designing a mining pool reward scheme with aligned incentives that is not vulnerable to pool hopping remains an **open problem**



Parameters:

- Pool has 10% of network hashrate
- 4 shares expected per valid block

AUTHOR: MAX FANG

BLOCKCHAIN AT BERKELEY

EXTRA

HONG KONG 2017

(Originally presented by Martin Koppelmann at SF Bitcoin Devs Seminar)

1. Use something better than QT
2. This works
    a. If you reach 51%, you get a higher reward, so it's sustainable

Hashrate / PoW does not secure Bitcoin/transactions - full nodes do! PoW only distributes votes

Other mechanics for vote distribution are maybe fine

5 STEPS TO DO A 51% ATTACK

1. **Publish mining software with higher EV**
    1. Mine on new headers (but validate it asap)
    2. More „flexible" 2 hours rule
    3. Decide for fork with own block version number
    4. Make miner aware of „Goldfinger" reward
    5. „Members only" functionality
2. **Create a pool with stickiness**
    1. New members will receive only 90% for shares in the first 2 weeks, after 2 weeks 110% (ponzi scheme)
3. **Create unwanted coalitions (timestamp attack)**
4. **Atack other pools with cannibalizing pools**
5. **Eventually switch to members only**

TRANSACTION MALLEABILITY

HONG KONG 2017

Transaction Malleability:

Nodes relaying a fresh transaction can tweak certain fields to make a version of the transaction with a different hash image, yet the *digital signature still verifies!*

TRANSACTION MALLEABILITY

HONG KONG 2017

Transaction Malleability:

Nodes relaying a fresh transaction can tweak certain fields to make a version of the transaction with a different hash image, yet the *digital signature still verifies!*

For example: in ECDSA, the following signature pairs are equivalent:

(r, **s** (mod N)) and (r, **-s** (mod N))

Both validate the same transaction data, but now **the hash image changes.**

Additionally, the **scriptSig** field can (sometimes) have extraneous script ops tacked on the end.

TRANSACTION MALLEABILITY

HONG KONG 2017

When is this an issue? In some cases, transactions rely on a chain of previous transactions (common in micro payments / lightning network).

Changing the hash image of a prior transaction in the chain will invalidate every subsequent transaction!

TRANSACTION MALLEABILITY

HONG KONG 2017

Example: Mt. Gox Incident

1. Mt. Gox sees attacker made withdrawals
2. Attackers used transaction malleability to change txid in-flight
3. To Mt. Gox, it looks like the transaction didn’t go through. Meanwhile, the BTC was actually sent to the attack!
4. Mt. Gox doesn't deduct amount from attacker's account, but still sent BTC.

Solutions? The new SegWit (Segregated Witness) proposal stores transaction signatures in a *separate* merkle tree, effectively fixing transaction malleability. *(Not yet in production)*