Lecture 7: HOW TO DESTROY BITCOIN: GAME THEORY AND ATTACKS

LECTURE OVERVIEW

1. POOL STRATEGIES
2. FORKING & DOUBLE SPENDING
3. CENSORSHIP
4. SELFISH MINING
5. DEFENSES DISCUSSION

1 POOL STRATEGIES

POOL REWARD SCHEMES

PAY-PER-SHARE

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

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

POOL REWARD SCHEMES

PROPORTIONAL

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

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

POOL REWARD SCHEMES

INCENTIVE MISALIGNMENT

Is it possible that mining pools are vulnerable to some incentive misalignment? Can miners take advantage of the differences in these two types of payout schemes?

POOL REWARD SCHEMES

INCENTIVE MISALIGNMENT

Is it possible that mining pools are vulnerable to some incentive misalignment? Can miners take advantage of the differences in these two types of payout schemes?

YES!

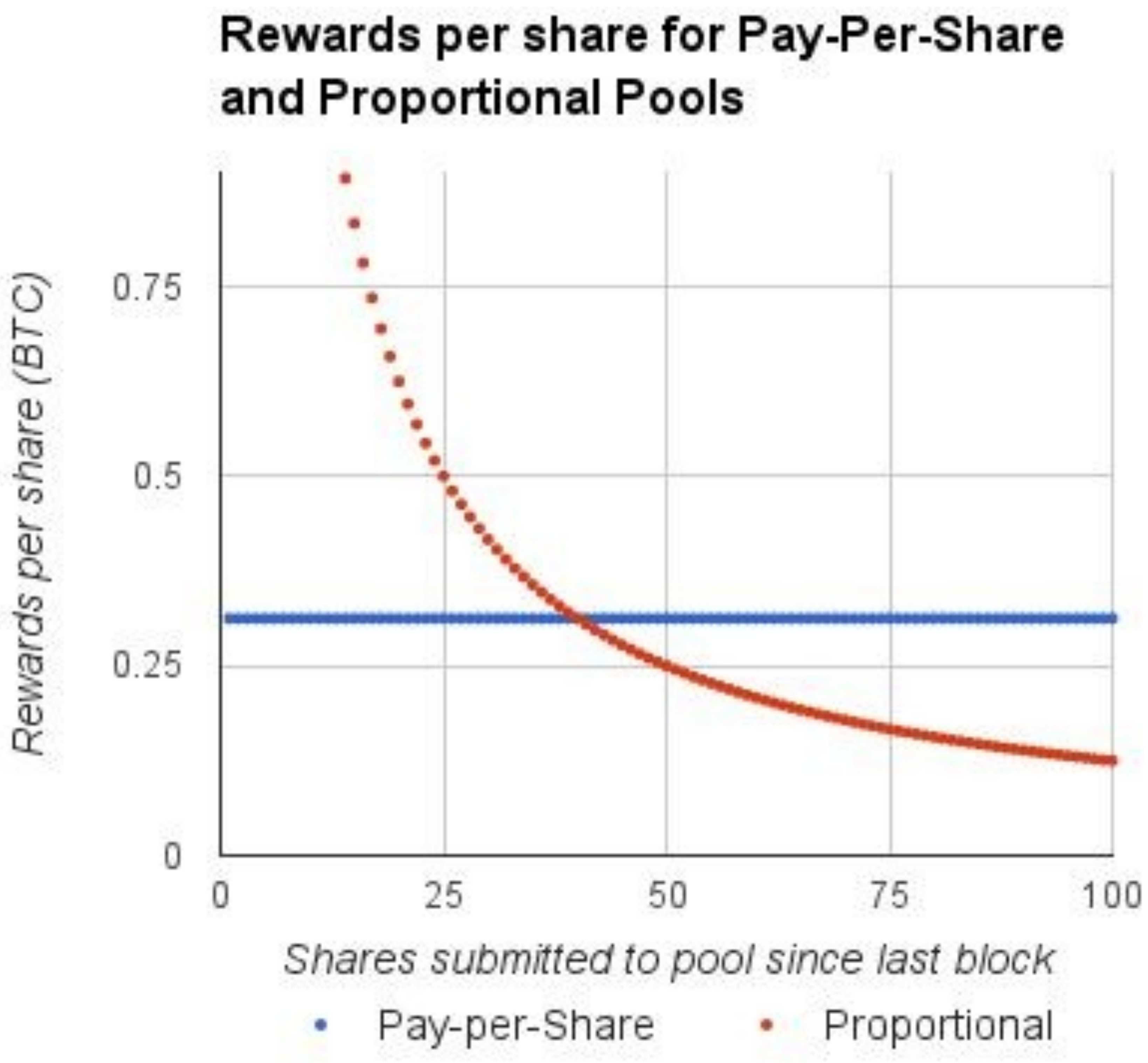
POOL REWARDS

POOL HOPPING

Proportional pool rewards per share high when just starting on a new block, but inversely proportional to number of shares submitted

Pay-per-share pool rewards per share constant no matter how many shares submitted

How does a miner maximize their profit?



Parameters:

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

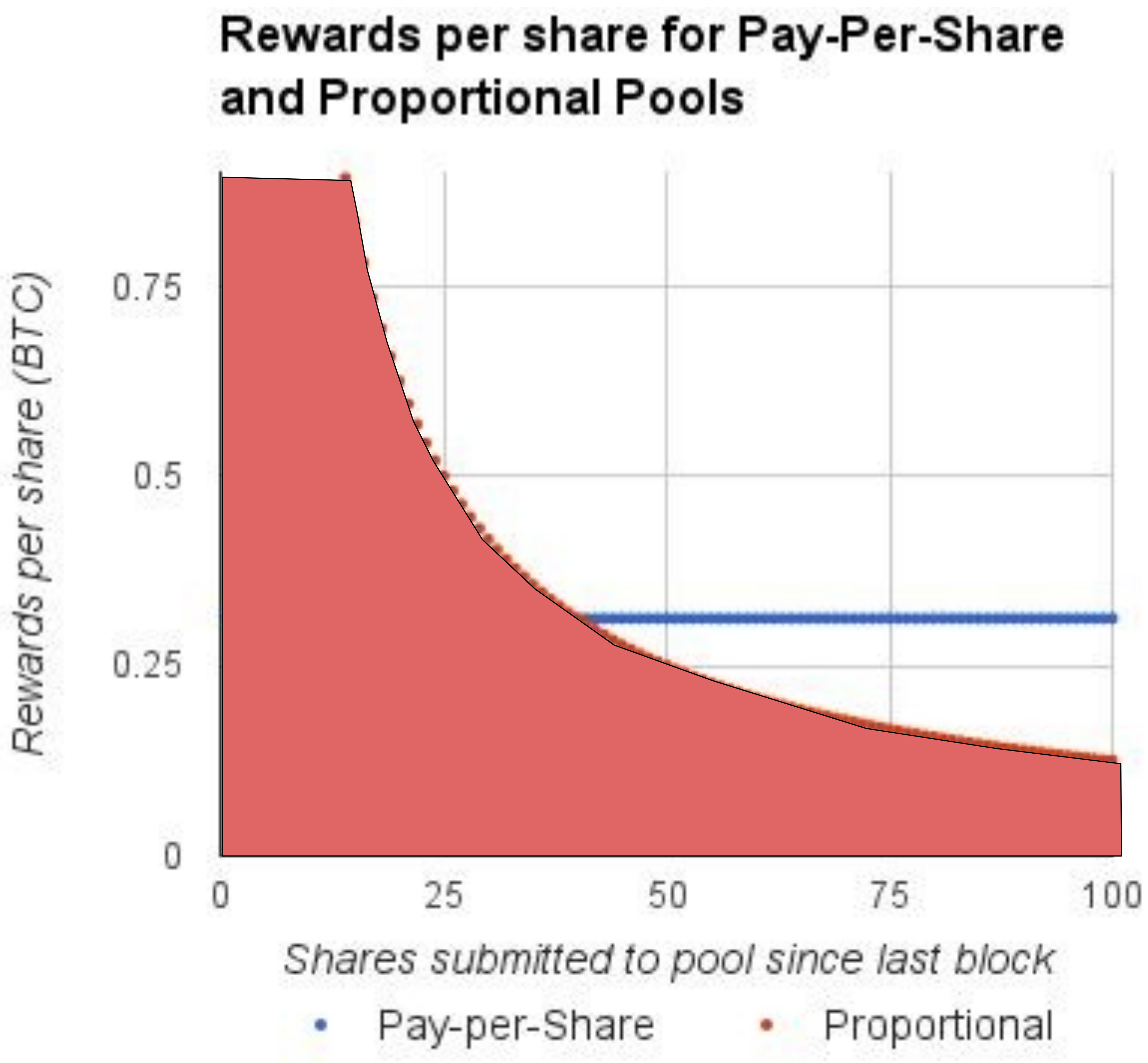
POOL REWARDS

POOL HOPPING

Proportional pool rewards per share high when just starting on a new block, but inversely proportional to number of shares submitted

Pay-per-share pool rewards per share constant no matter how many shares submitted

How does a miner maximize their profit?



Parameters:

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

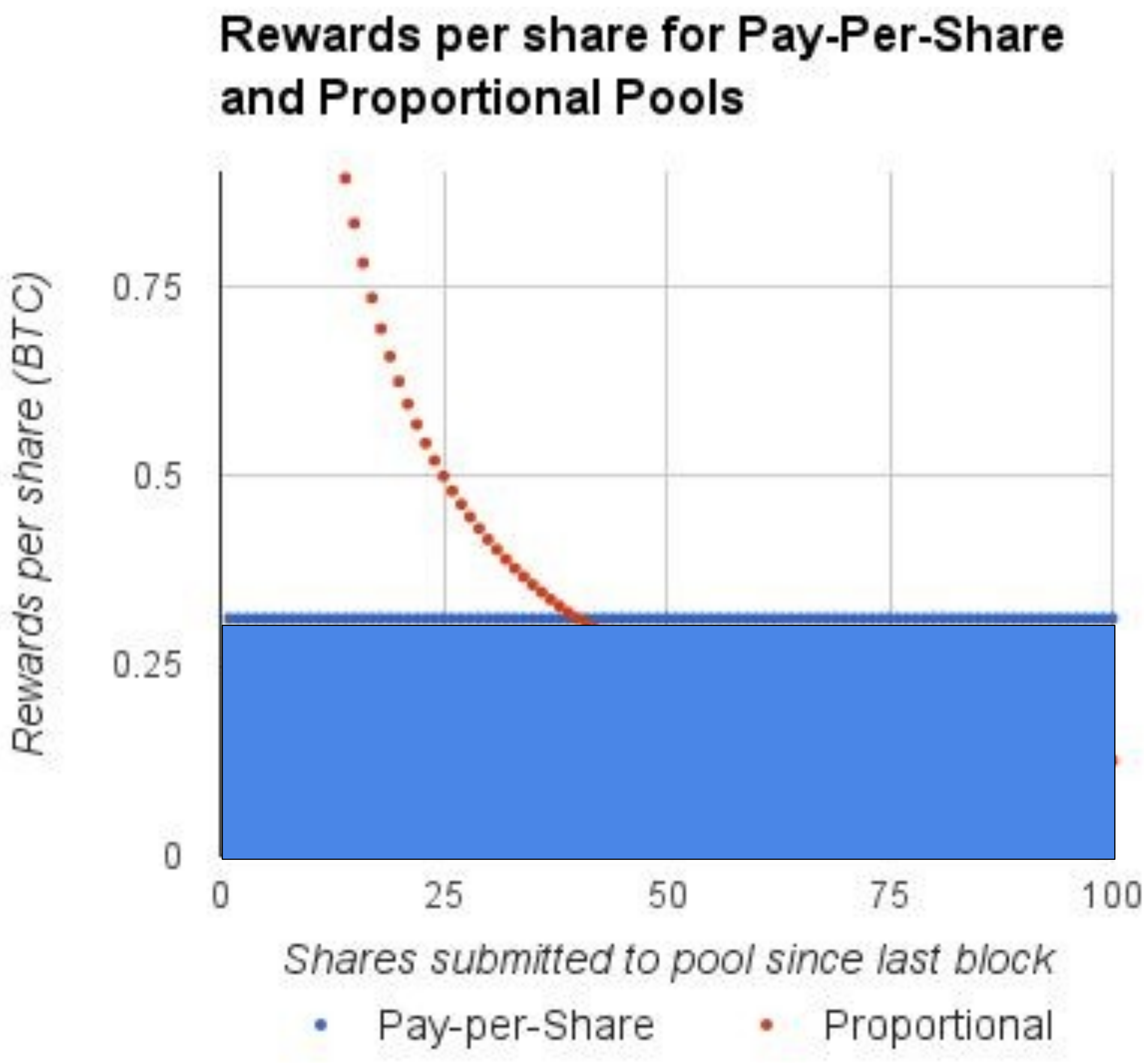
POOL REWARDS

POOL HOPPING

Proportional pool rewards per share high when just starting on a new block, but inversely proportional to number of shares submitted

Pay-per-share pool rewards per share constant no matter how many shares submitted

How does a miner maximize their profit?



Parameters:

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

POOL REWARDS

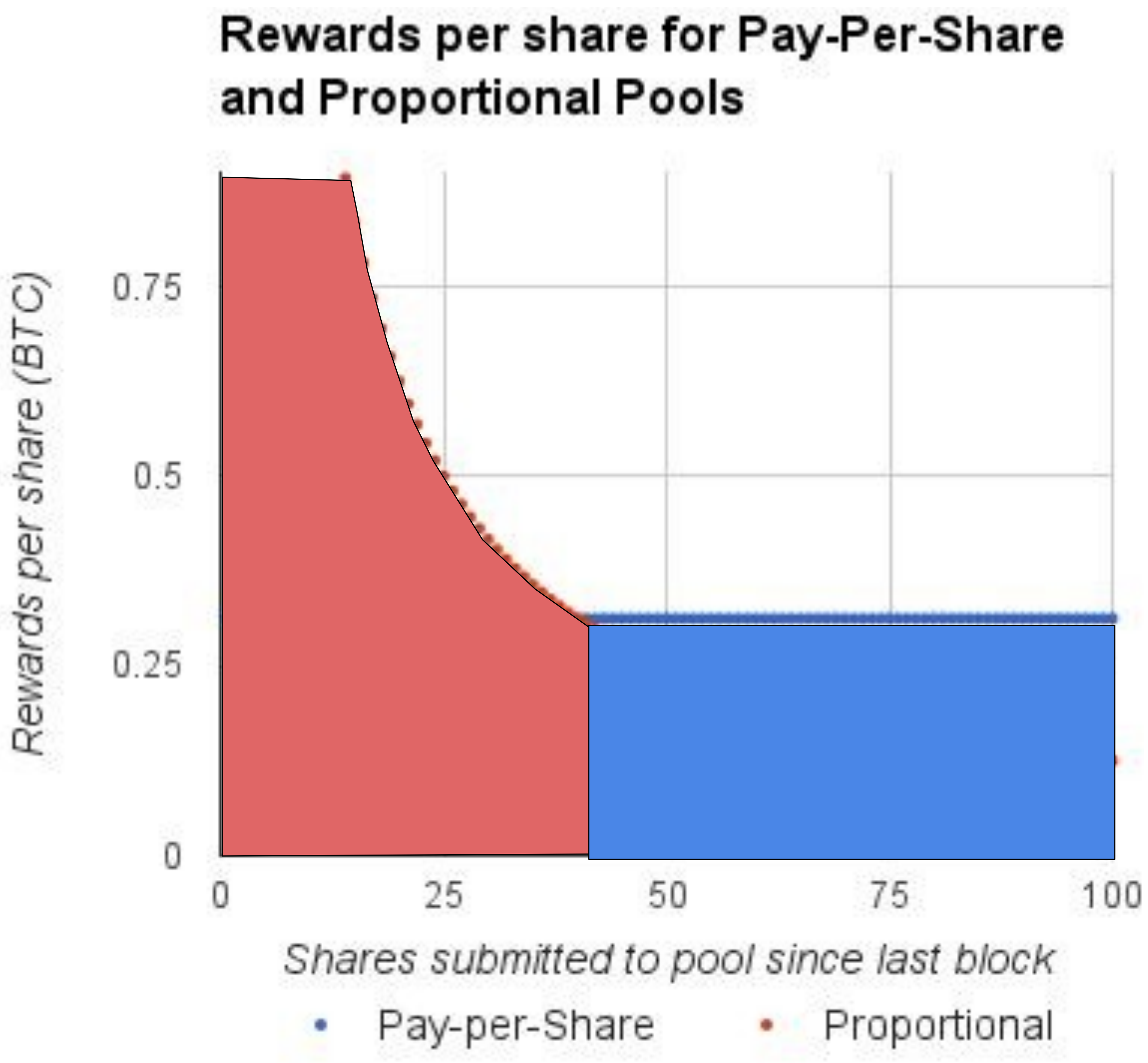
POOL HOPPING

Pool hopping: switching between pools to increase total rewards

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

Example profit maximizing 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



Parameters:

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

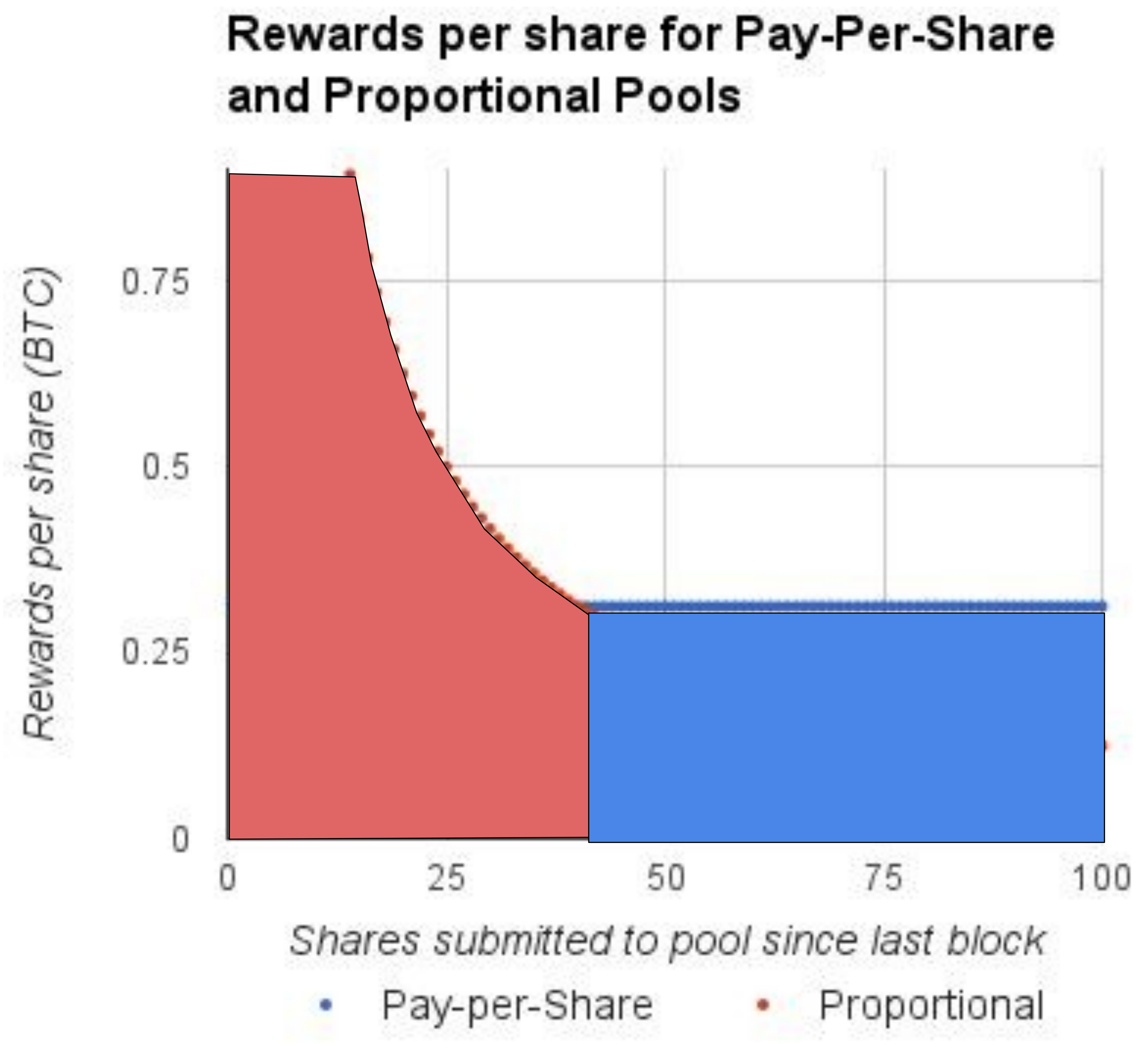
POOL REWARDS

POOL HOPPING

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

POOL CANNIBALIZATION

NOM NOM NOM NOM

Cannibalizing Pools - Distribute some small % of mining power equally among all other pools, withhold valid blocks.

- Rewards will still be received
- Undetectable unless statistically significant

Image source: http://cdn3-www.craveonline.com/assets/uploads/2014/08/Hannibal-Cookbook.jpg

POOL CANNIBALIZATION

EXAMPLE

Givens:

- We have 30 H/s. Total network is 100 H/s.
- Assume 1BTC block reward. All of the following numbers are expected value.
- 30% HR (hashrate)
  - = 30% MR (Mining Reward)
  - = 0.3 BTC

Let's say we buy more mining equipment, worth 1H/s

Standard mining strategy:

- Add 1% HR => 31/101 = 30.69% HR
- Reward: 0.3069 *1BTC = 0.3069 BTC
  - Revenue gain = **0.0069 BTC for 1% hashrate added**

New Hashrate	31/101 ≈ 30.69%
Mining Reward	0.3069 BTC
Revenue Gain	0.0069 BTC

POOL CANNIBALIZATION

EXAMPLE

Pool cannibalizing strategy:

Distribute 1H/s among all other pools

- **DON'T SUBMIT VALID BLOCKS**

Other pool hashrate breakdown:

- 70/71 honest, 1/71 dishonest
  - 70% **effective** hashrate = 0.7 BTC
- You own 1/71 of other pools
  - Expected value of mining: (1/71)*0.7 BTC = 0.0098 BTC

New Hashrate	31/101 ≈ 30.69%
Mining Reward	0.3098 BTC
Revenue Gain	0.0098 BTC

POOL CANNIBALIZATION

EXAMPLE

Givens:

- We have 30 H/s. Total network is 100 H/s.
- Assume 1BTC block reward. All of the following numbers are expected value.
- 30% HR (hashrate)
  - = 30% MR (Mining Reward)
  - = 0.3 BTC

Let's say we buy more mining equipment, worth 1H/s

Honest	
Mining Reward	0.3069 BTC
Revenue Gain	0.0069 BTC

Cheat	
Mining Reward	**0.3098 BTC**
Revenue Gain	**0.0098 BTC**

More profitable to cannibalize pools than mine honestly

POOL WARS

THE GAME OF LIFE

- Attack decisions resemble an iterative game
  - Two main players: Pool 1 and Pool 2
- Each iteration of the game is a case of the Prisoner's Dilemma
  - Choose between attacking or not attacking

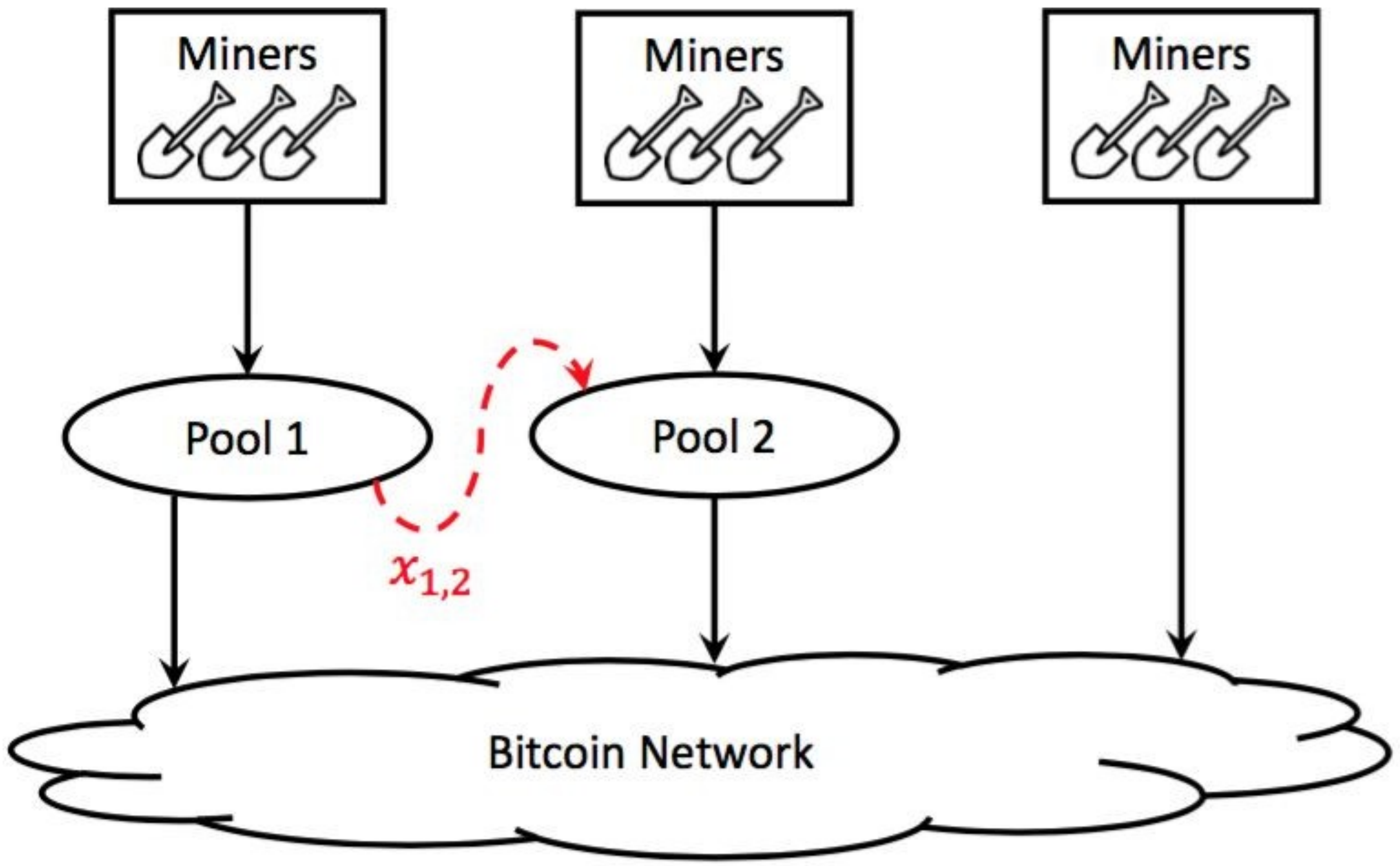


Fig. 3. The one-attacker scenario. Pool 1 attacks pool 2.

Prisoner's Dilemma

		PRISONER 2	
		Confess	Lie
PRISONER 1	Confess	-8 , -8	0 , -10
	Lie	-10 , 0	-1 , -1

POOL WARS

NASH EQUILIBRIUM

- If Pool 1chooses to attack Pool 2, Pool 1 gains revenue, Pool 2 loses revenue
  - Pool 2 can retaliate by attacking Pool 1 and gaining more revenue
- Thus, attacking is the dominant strategy in each iteration
  - Therefore if both Pool 1and Pool 2 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	$(r_1 = \tilde{r}_1 < 1, r_2 > 1)$	$(\tilde{r}_1 < r_1 < 1, \tilde{r}_2 < r_2 < 1)$

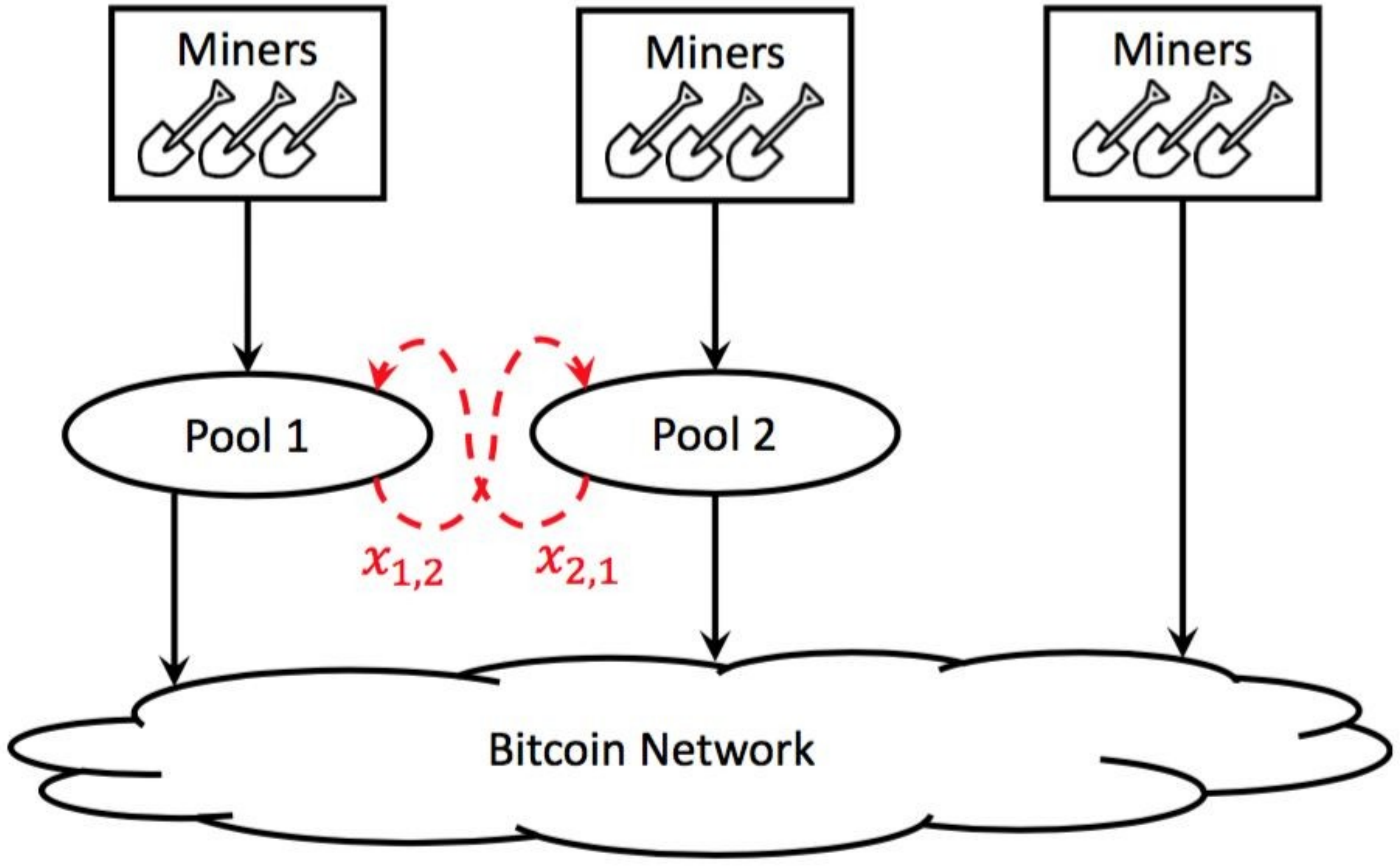


Fig. 7. Two pools attacking each other.

POOL WARS

TRAGEDY OF THE COMMONS

- 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

Nash Equilibrium is a Tragedy of the Commons

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	$(r_1 = \tilde{r}_1 < 1, r_2 > 1)$	$(\tilde{r}_1 < r_1 < 1, \tilde{r}_2 < r_2 < 1)$

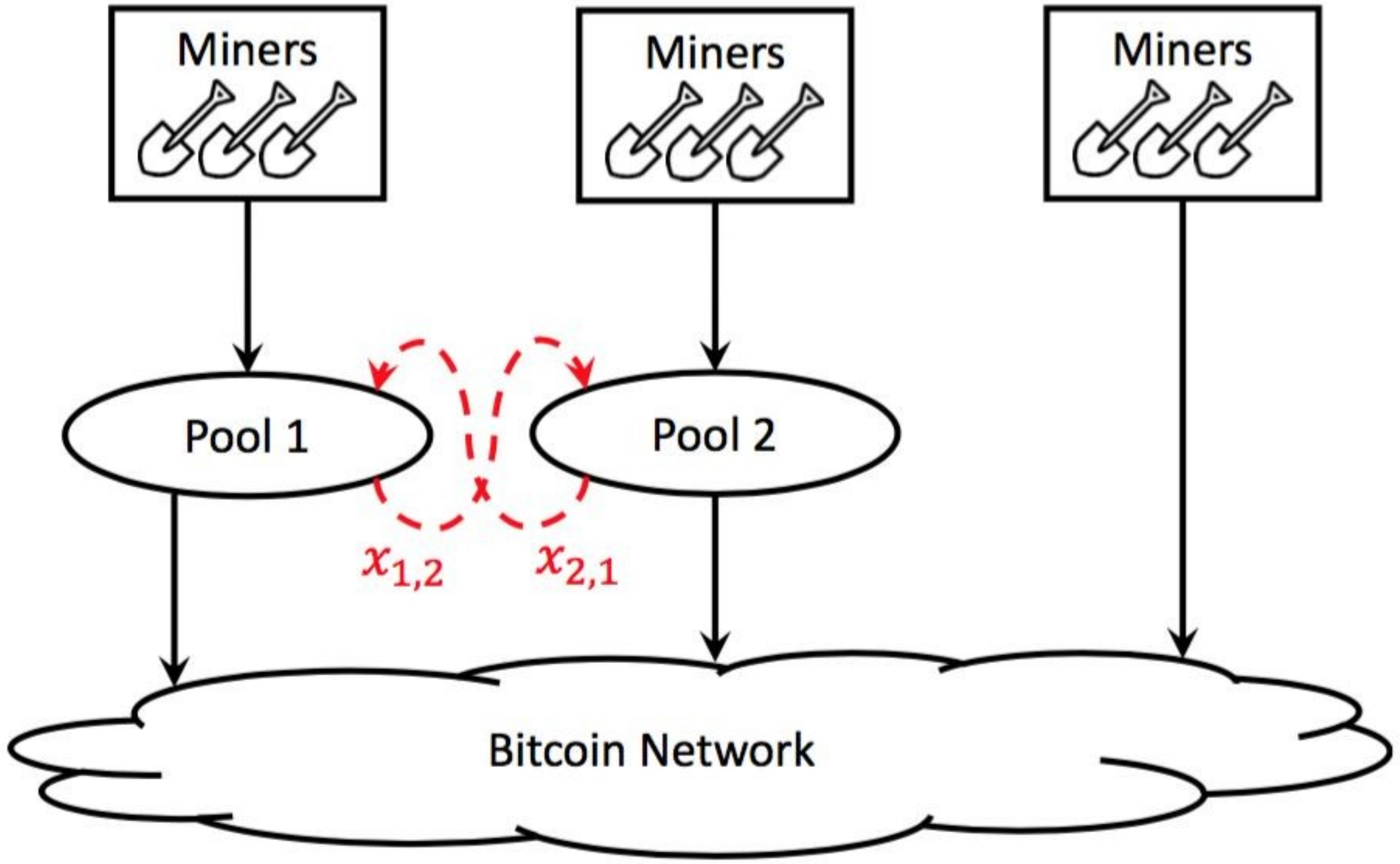


Fig. 7. Two pools attacking each other.

2

FORKING & DOUBLE SPENDS

DOUBLE SPENDING

THE CLASSIC ATTACK

Double Spend: Successfully spending the <u>same</u> value more than once.

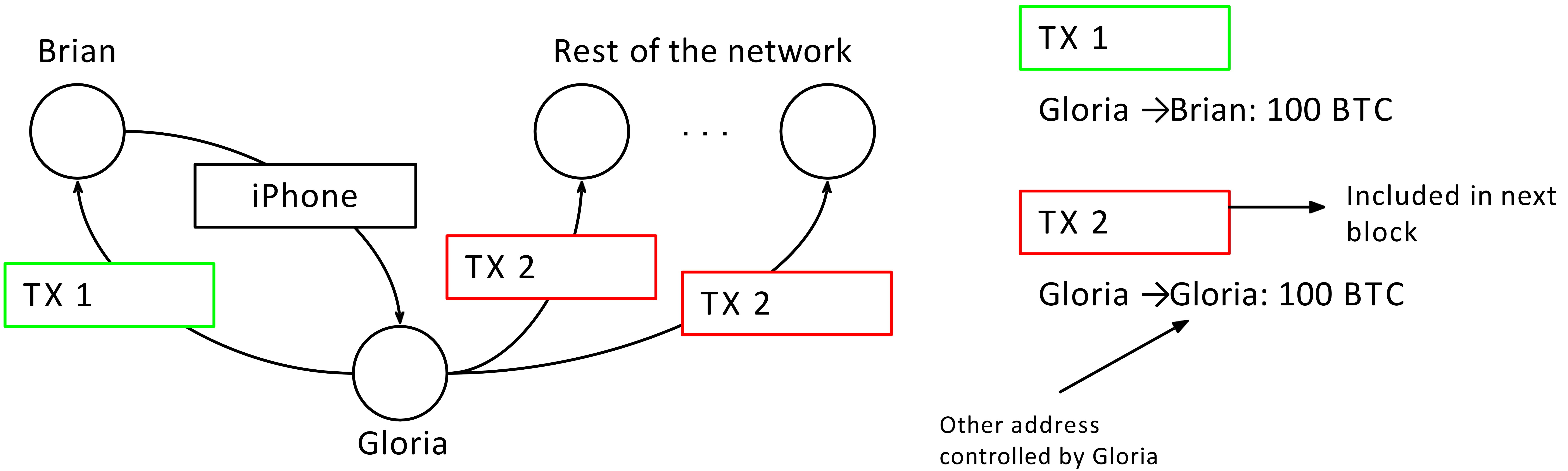
- In the year 2099, Gloria wants to buy an iPhone 92XCS from Brian on the black market for 100 BTC but doesn’t want to give up her bitcoins. #HODL
    - How can Gloria double spend on Brian?

DOUBLE SPENDING

RACE ATTACK

Suppose Brian simply checks that the transaction he sees is valid and **immediately** sends Gloria the iPhone. Brian is vulnerable to a **Race Attack**!

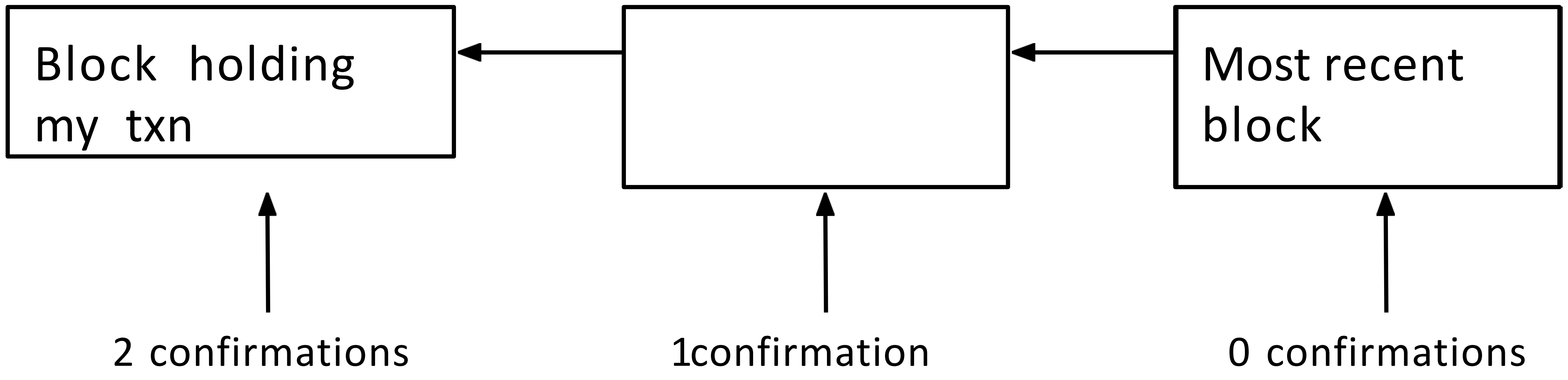
How can we stop this?



DOUBLE SPENDING

CONFIRMATIONS

Confirmations: The number of blocks created on top of the block a txn is in.



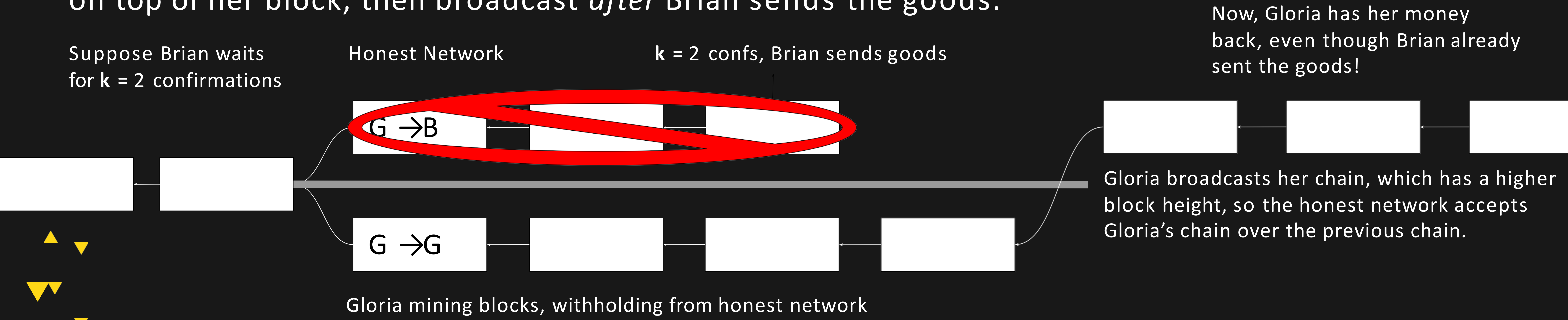
DOUBLE SPENDING

k CONFIRMATIONS

Clearly not secure if Brian doesn't wait for any confirmations…

What if Brian waits for **k** confirmations?

[G →B] transaction needs **k** confirmations before Brian sends the goods. In order to double spend on Brian, Gloria needs to start a private chain containing her malicious transaction, mine **k** blocks on top of her block, then broadcast *after* Brian sends the goods.



"CLICKER" QUESTION:

CONFIRMATIONS

How many confirmations should I wait for to be sure my transaction is "valid"?

A) 0

B) 1

C) 2

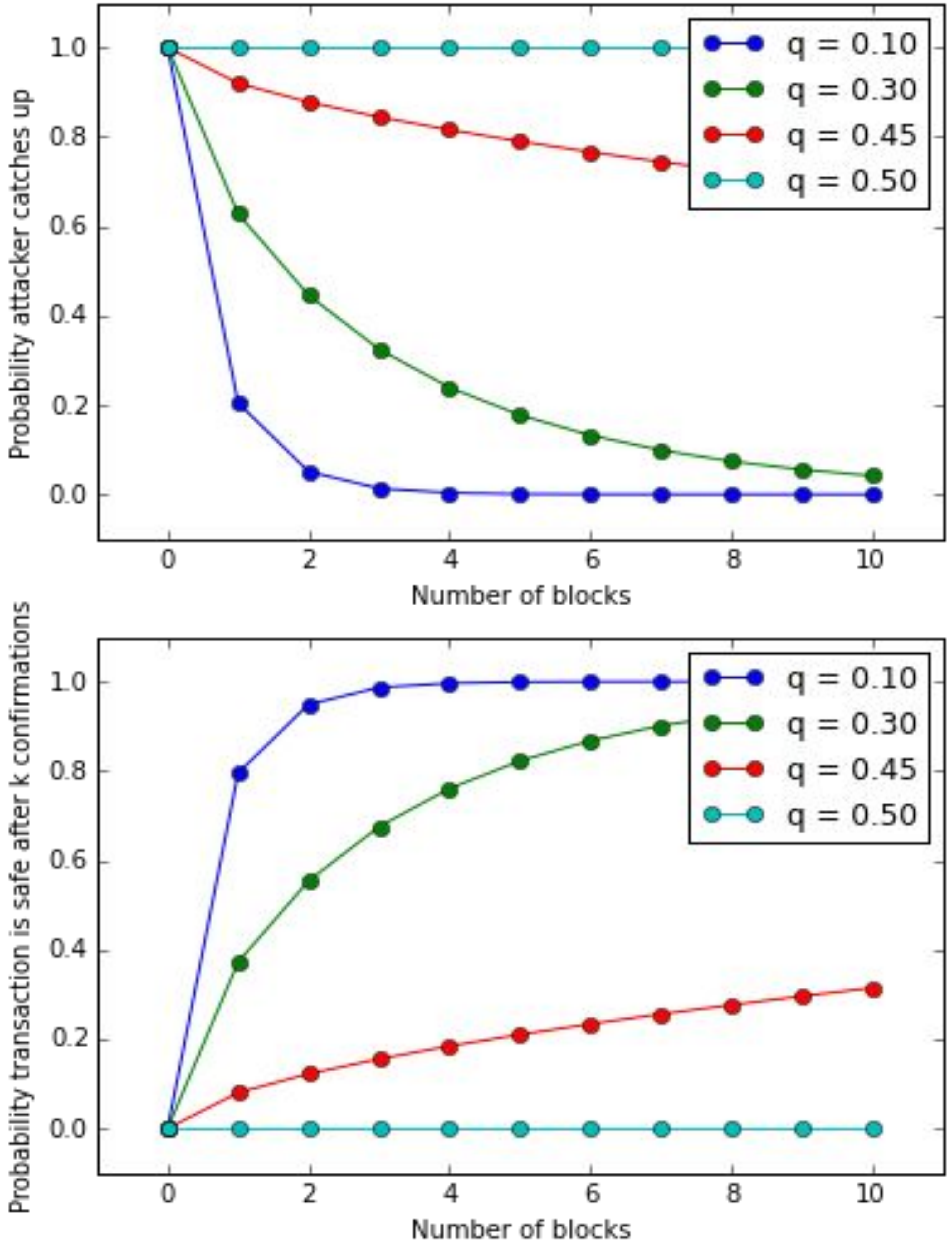
D) none of these

DOUBLE SPENDING

PROBABILITIES

Probabilities of success for the attacker

(Inverse represents bounds of the probability of safety for the vendor given assumptions of the attacker’s hashpower)



DOUBLE SPENDING

51% ATTACK

What if Gloria controls more than 50% of the total network hash power?

Whenever Gloria's chain is behind the honest network's chain, she will *always* (in expectation) be able to catch up and out-produce the honest miners.

Therefore, the probability that Gloria can successfully **double spend** with >50% hash power reaches 100%!

DOUBLE SPENDING

INCENTIVES

Why would Gloria not want to double spend?

If the rest of the network detects the double spend, it is assumed that confidence in the cryptocurrency and exchange rate would *plummet*.

Bribing Miners:

Gloria might not physically control the mining hardware necessary to perform a double spend.

Instead, Gloria can bribe miners or even entire pools to mine on her withheld chain.

DOUBLE SPENDING

What if Gloria is a **hostile government /adversarial altcoin /large finance institution** with *significant capital* available?

Gloria can acquire enough mining ASICs or bribe enough miners / pools to achieve >50% effective hash power.

Gloria can perform a so-called **“Goldfinger” attack** with the objective of *destroying* the target cryptocurrency, either by destroying confidence in the currency with a double spend or spamming the network with empty blocks. If Gloria isn’t staked in Bitcoin she can *short* the currency to profit after her attempted double spend.

Ex: Eligius pool kills CoiledCoin altcoin

DOUBLE SPENDING

CONFIRMATIONS

If I'm buying a coffee with bitcoin, should I have to wait 6 confirmations (approx 1hr) before I can get my coffee? What if I'm buying a house?

3 CENSORSHIP

BLACKLISTING

FREEZE PEACH IS DEAD

Say Gloria is a government, or has control over a government, that has jurisdiction over mining pools. (The Glorian nation) In addition, Gloria's mining pools have **over 51% of the network's hashpower.**

Objective: Censor the Bitcoin addresses owned by certain people, say <u>Brian</u>, and prevent them from spending any of their Bitcoin

Block containing transactions from Brian

Normal block

Block mined by Gloria

BLACKLISTING

NAIVE CENSORSHIP STRATEGY

First strategy:
Gloria tells her mining pools not to include Brian's transactions **(blacklisting)**

Does this strategy actually work?

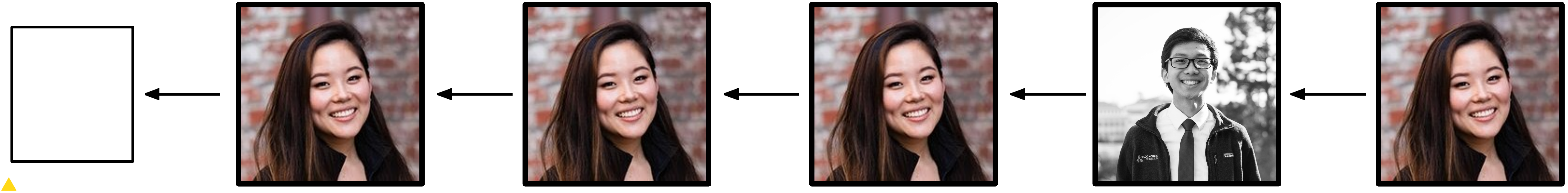
BLACKLISTING

NAIVE CENSORSHIP STRATEGY

First strategy:

Gloria tells her mining pools not to include Brian’s transactions **(blacklisting)**

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

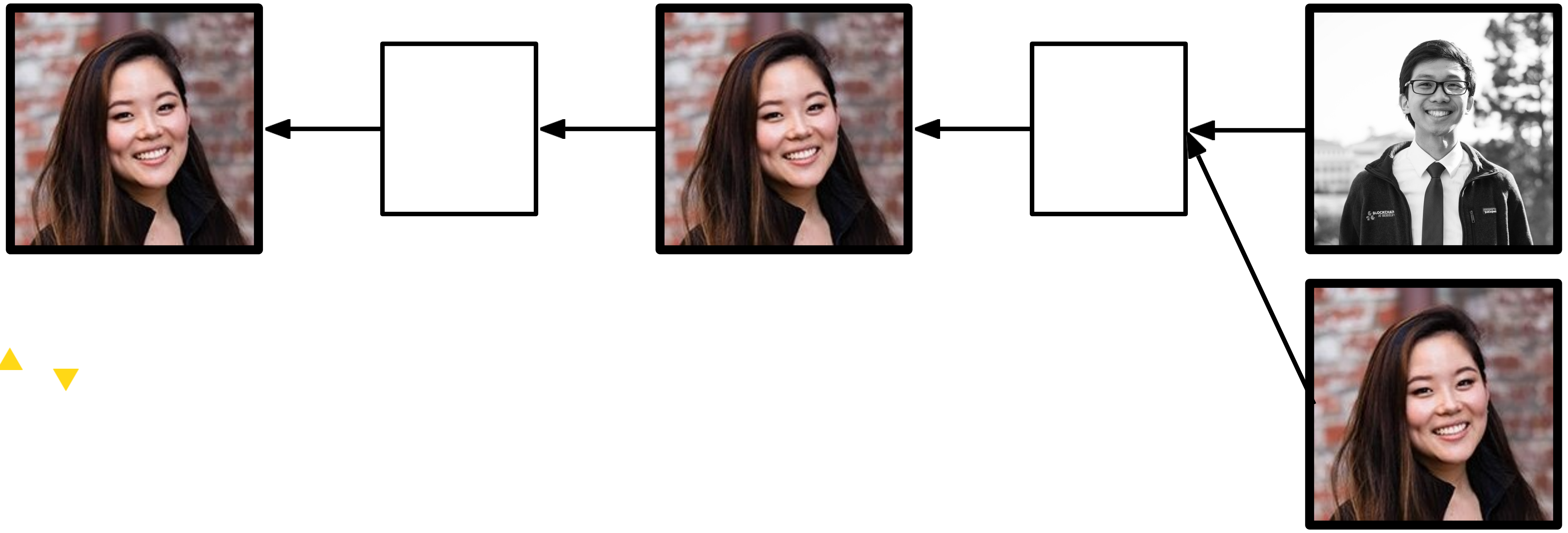


BLACKLISTING

PUNITIVE FORKING

Second strategy:

- Remember you are Gloria: **you have >51% of the network hashrate**
- Mandate that Glorian pools will refuse to work on a chain containing transactions spending from Brian's address
- Announce this to the world



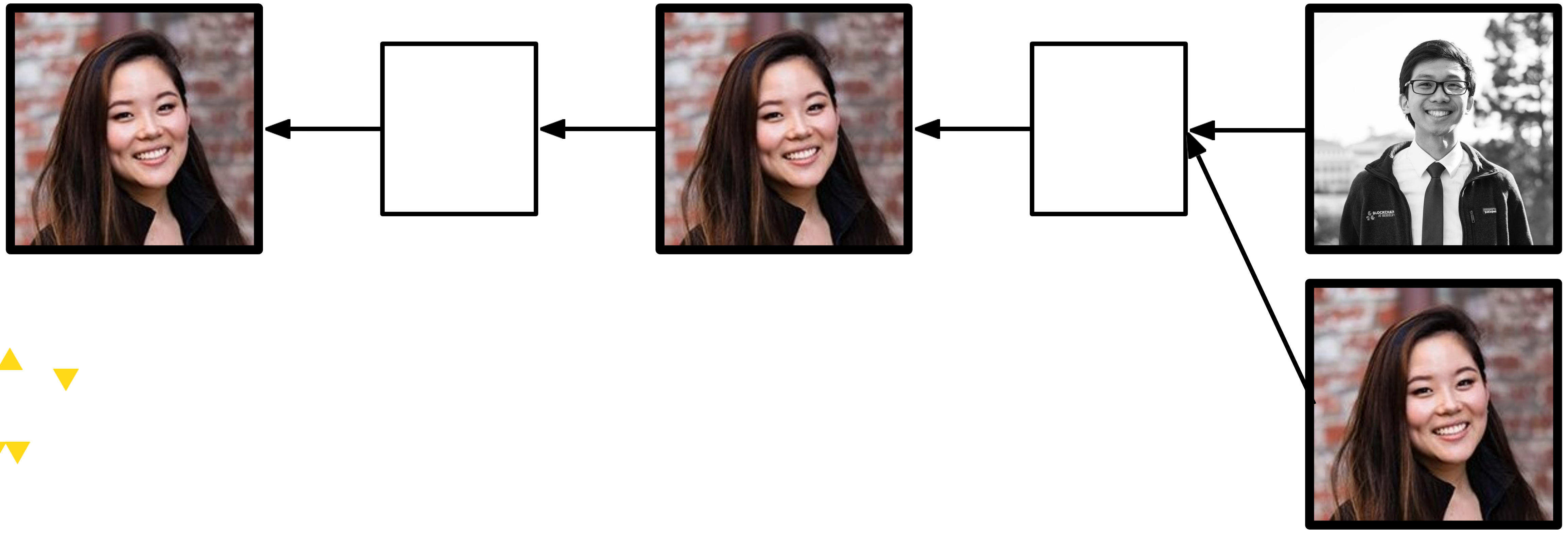
Does this strategy actually work?

BLACKLISTING

PUNITIVE FORKING

Second strategy:

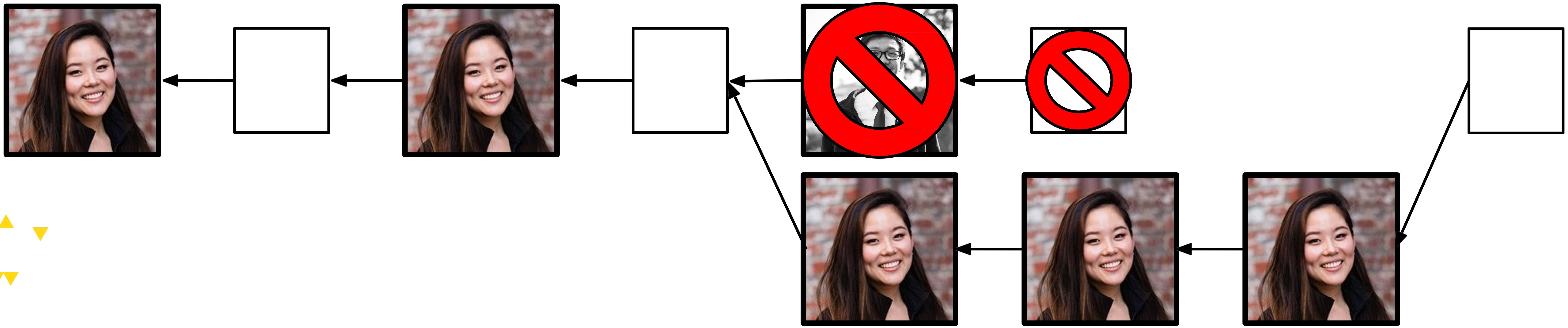
- Remember you are Gloria: **you have >51% of the network hashrate**
- Mandate that Glorian pools will refuse to work on a chain containing transactions spending from Brian's address
- Announce this to the world



BLACKLISTING

PUNITIVE FORKING

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

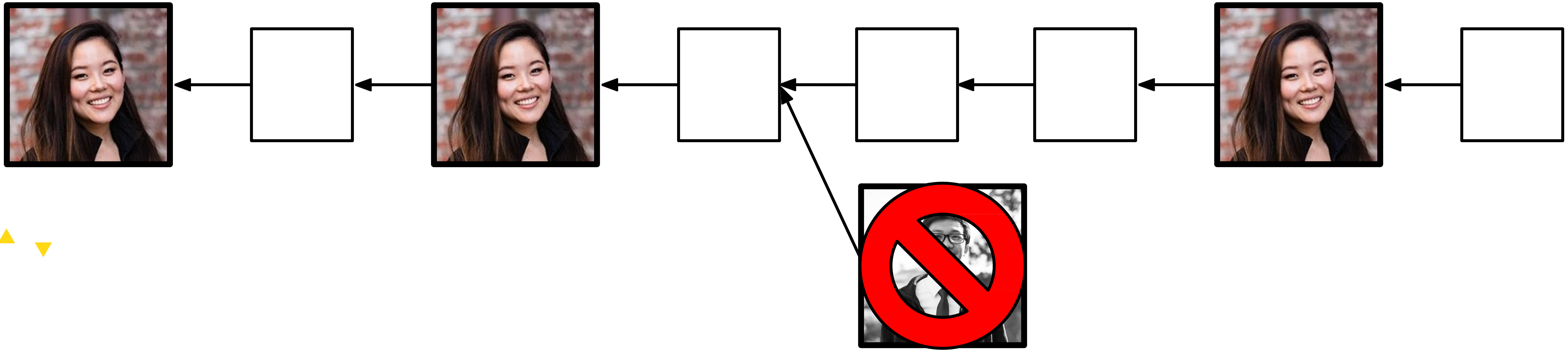


BLACKLISTING

PUNITIVE FORKING

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

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

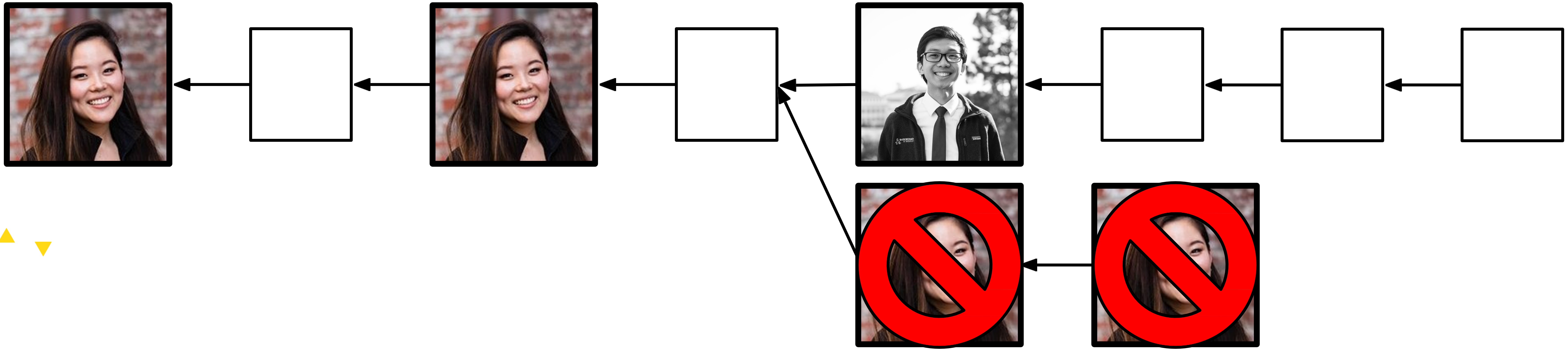


BLACKLISTING

FEATHER FORKING

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

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



BLACKLISTING

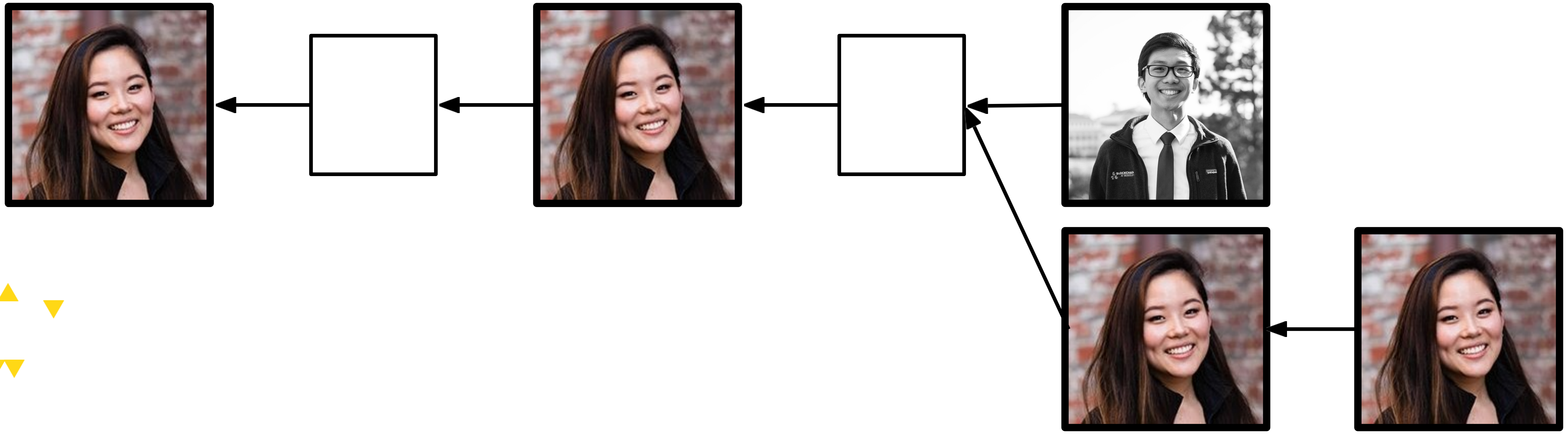
FEATHER FORKING

Let **q** equal the proportion of mining power Gloria have, 0 < **q** < 1

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

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

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



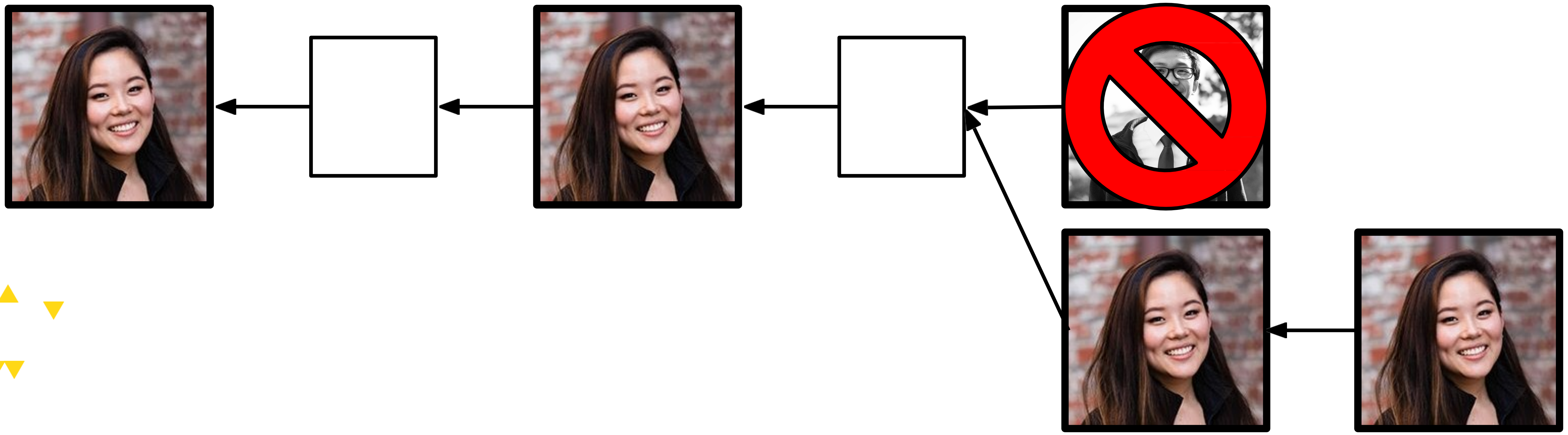
BLACKLISTING

FEATHER FORKING

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 Brian's tx in their block

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

EV(don't include) = BlockReward



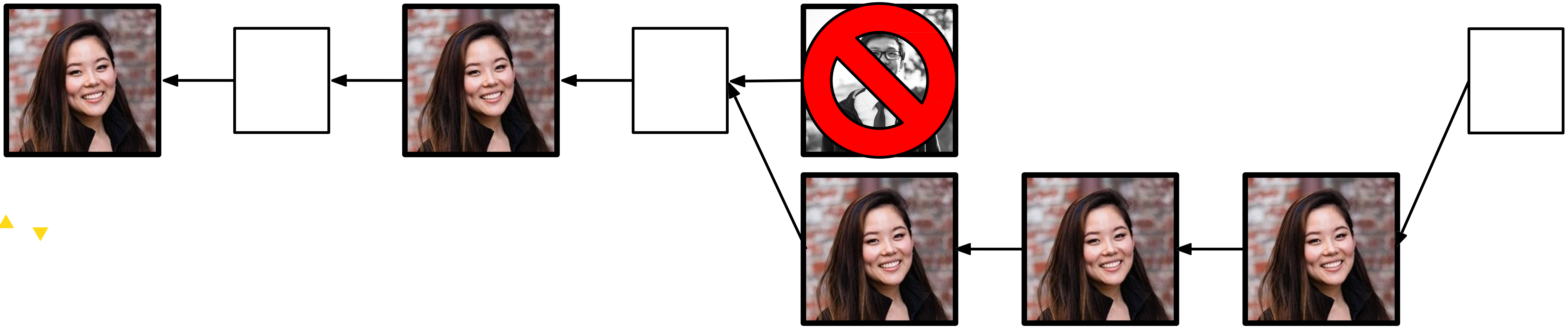
BLACKLISTING

FEATHER FORKING

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

Therefore, unless Brian 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 = Brian must pay **~$5000** minimum/transaction



4

SELFISH MINING

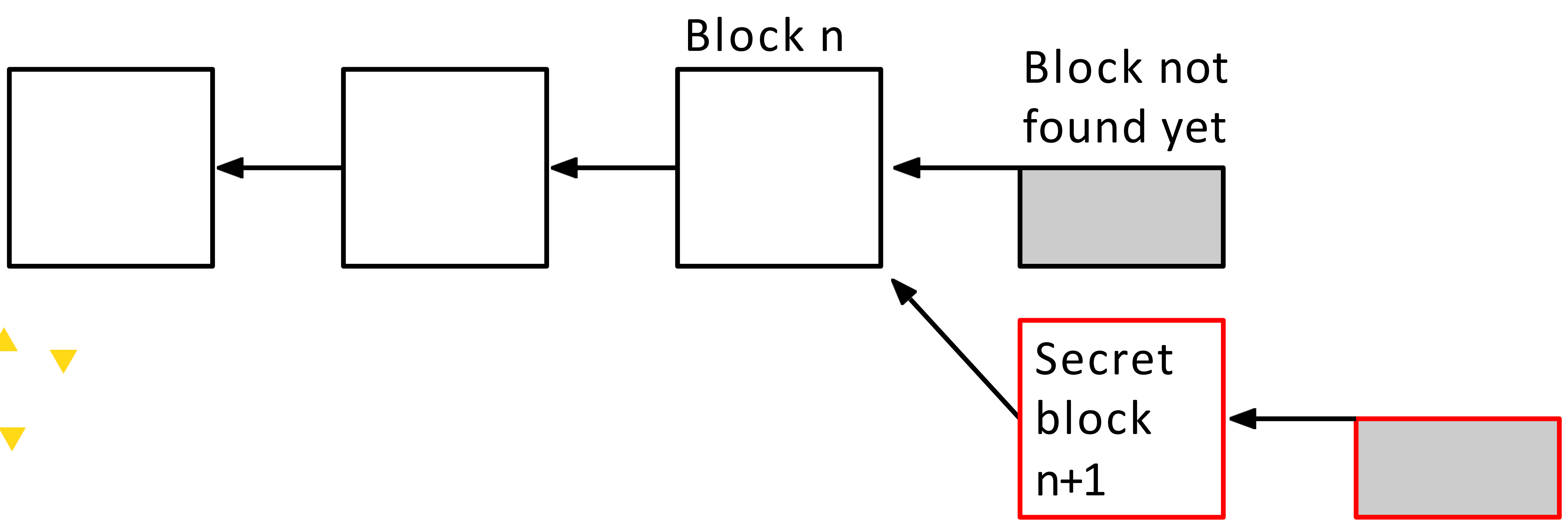
SELFISH MINING

BLOCK WITHHOLDING

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**



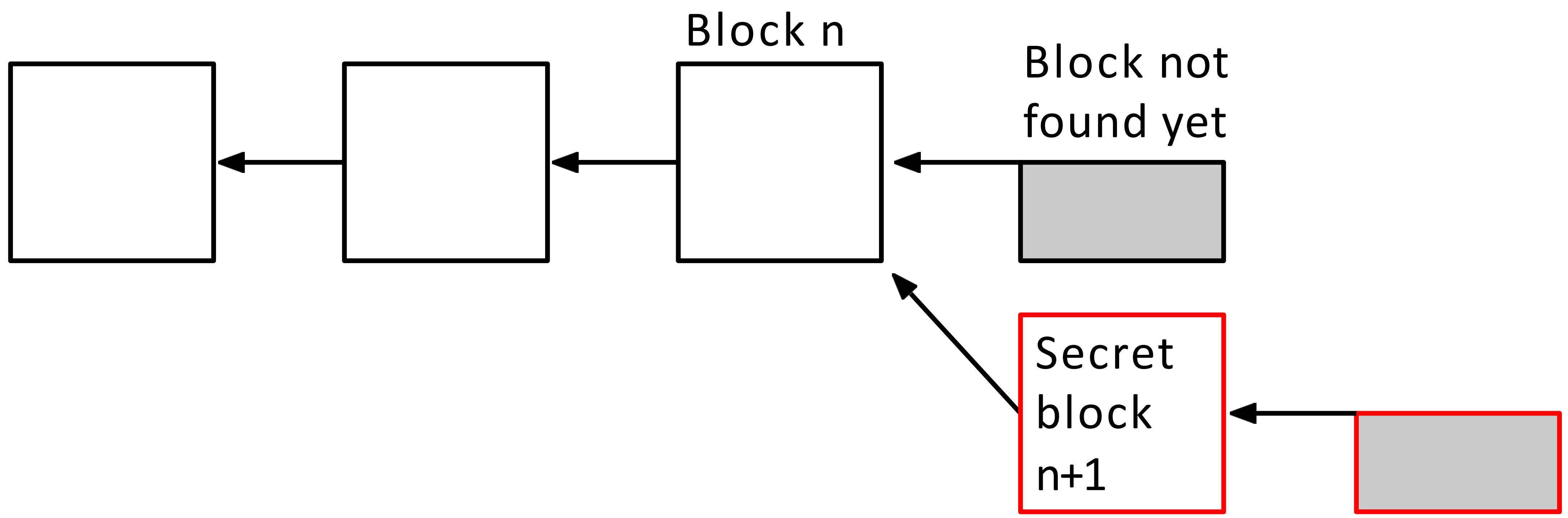
Note: "block-withholding" is also sometimes used in the context of mining pools - submitting shares but withholding valid blocks

SELFISH MINING

BLOCK WITHHOLDING

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

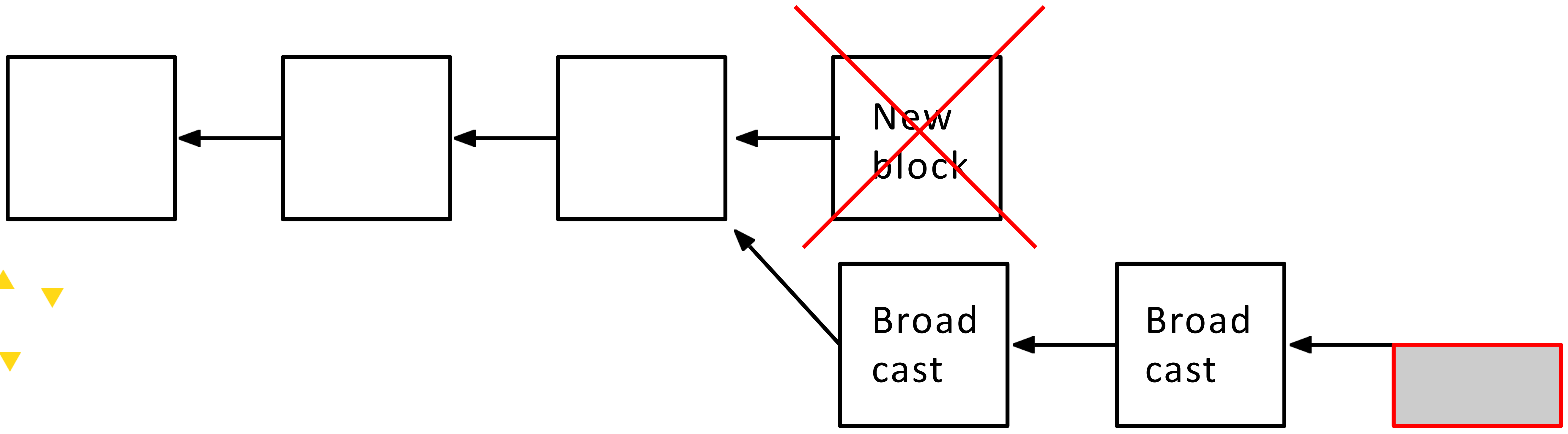


SELFISH MINING

BLOCK WITHHOLDING

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!**

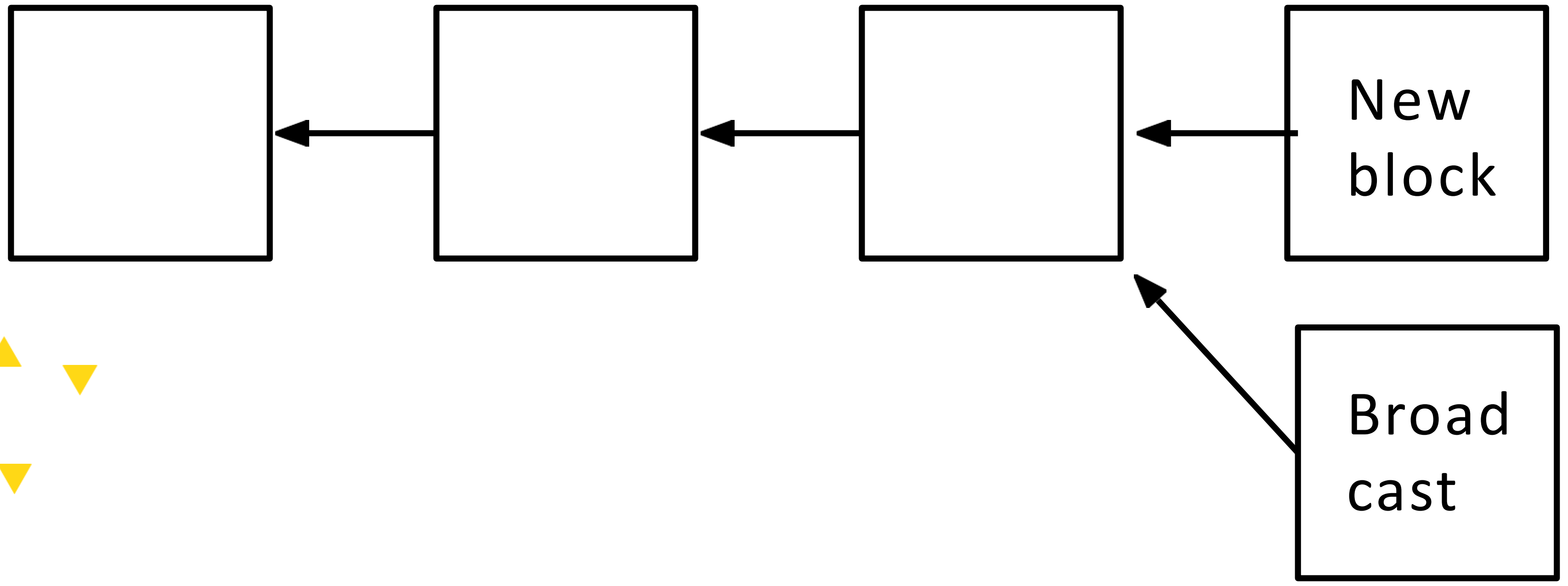


SELFISH MINING

BLOCK WITHHOLDING

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)



5 DEFENSE DISCUSSION

PROBLEM STATEMENT

USE SIGNATURES DUMMY

How do we prevent selfish mining?

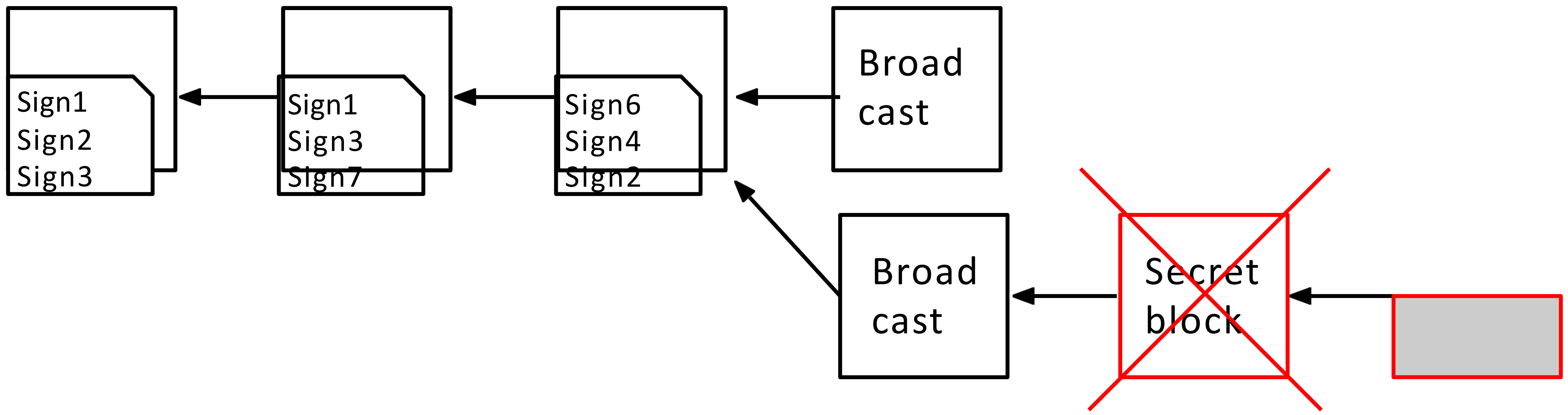
Hint: Use signatures

BLOCK VALIDATION

DUMMY BLOCK SIGNATURES

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

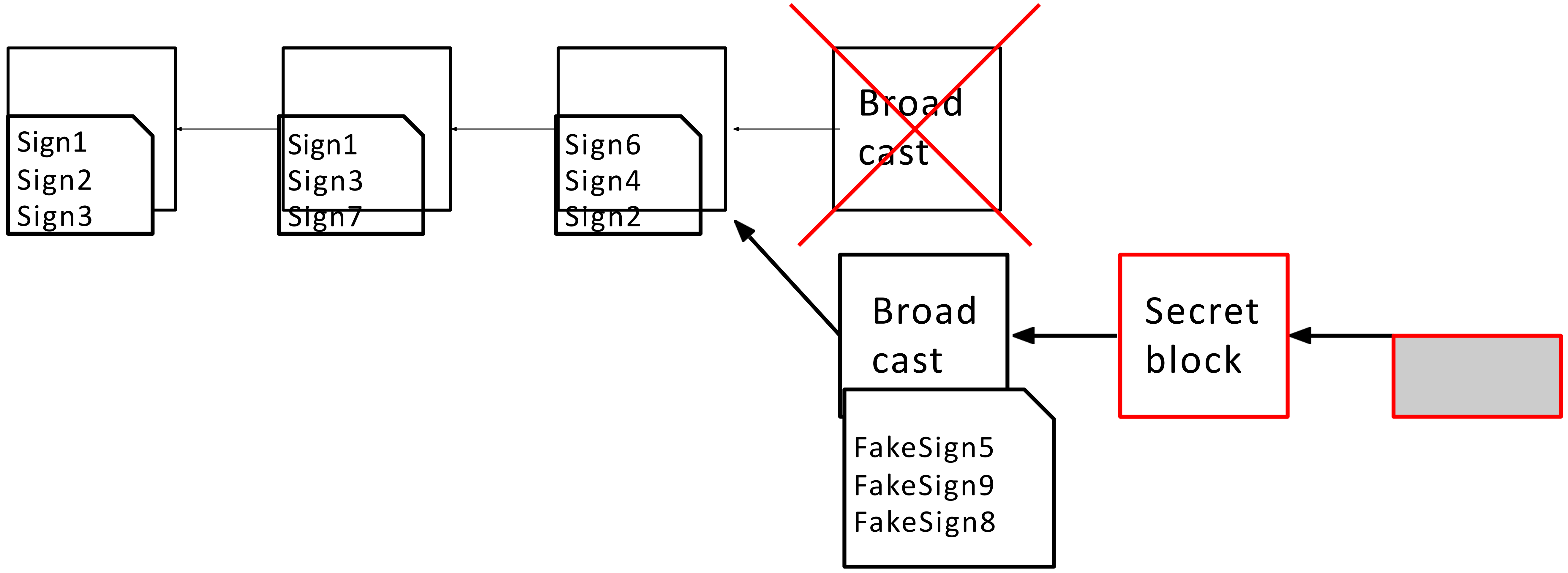
- Accompany solved blocks with signatures on dummy blocks
- Proves that the block is witnessed by the network
    - Proves that a competing block is absent before miners are able to work on it



PROBLEM STATEMENT

THE PROBLEM WITH DUMMY BLOCKS

- Doesn't provide a mechanism to evaluate whether the **number of proofs** is adequate
- Does not discuss how to prevent **Sybil attacks** on signatures
  - Selfish miner generates many signatures on the dummy block
- This defense requires **fundamental changes** to the block validity rules



PROBLEM STATEMENT

How do we prevent selfish mining?

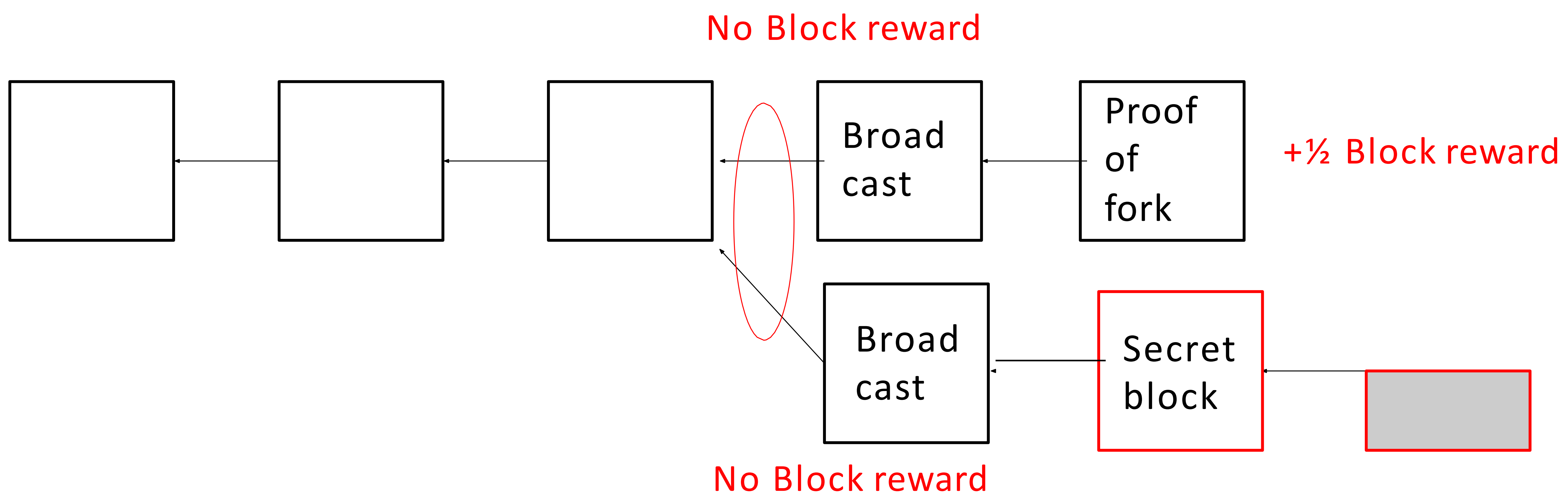
Hint: Use block rewards

FORK-PUNISHMENT

FORK-PUNISHMENT RULE

Proposed by Lear Bahack (2013)

- Competing blocks receive no block reward
- The first miner who incorporates a proof of the block fork in the blockchain gets half of the forfeited rewards



FORK-PUNISHMENT

FORK-PUNISHMENT RULE

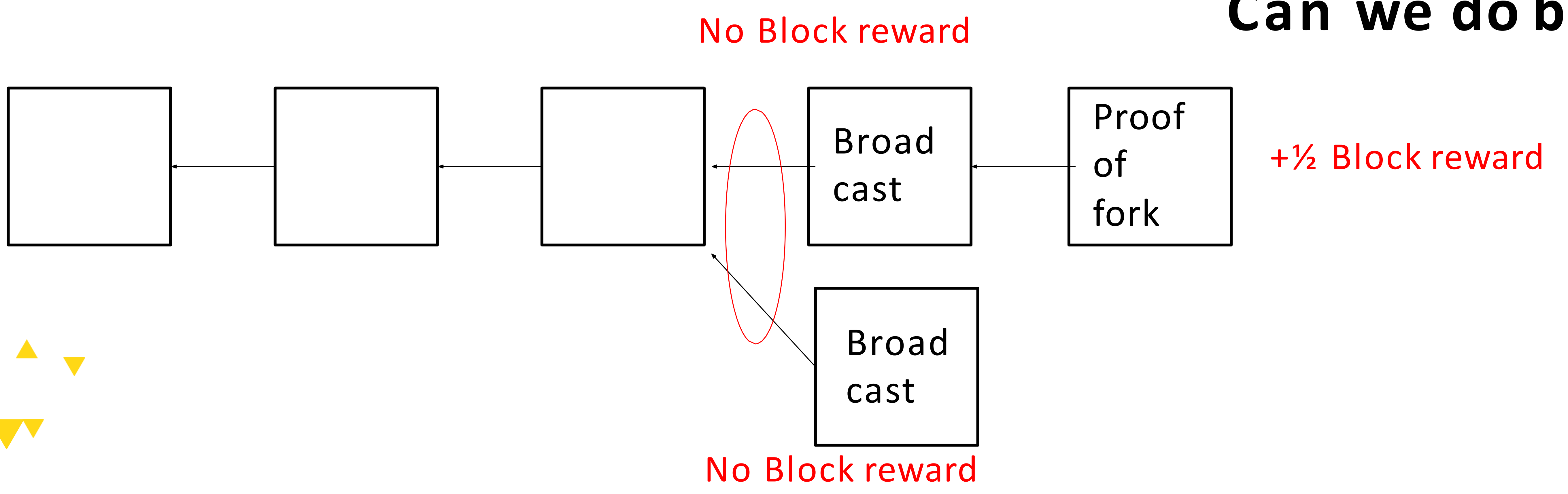
Drawbacks

- Honest miners suffer **collateral damage** of this defense
  - And lead to more attacks...

This defense requires **fundamental changes** to the reward distribution rules

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

Can we do better?



THE BITCOIN NETWORK

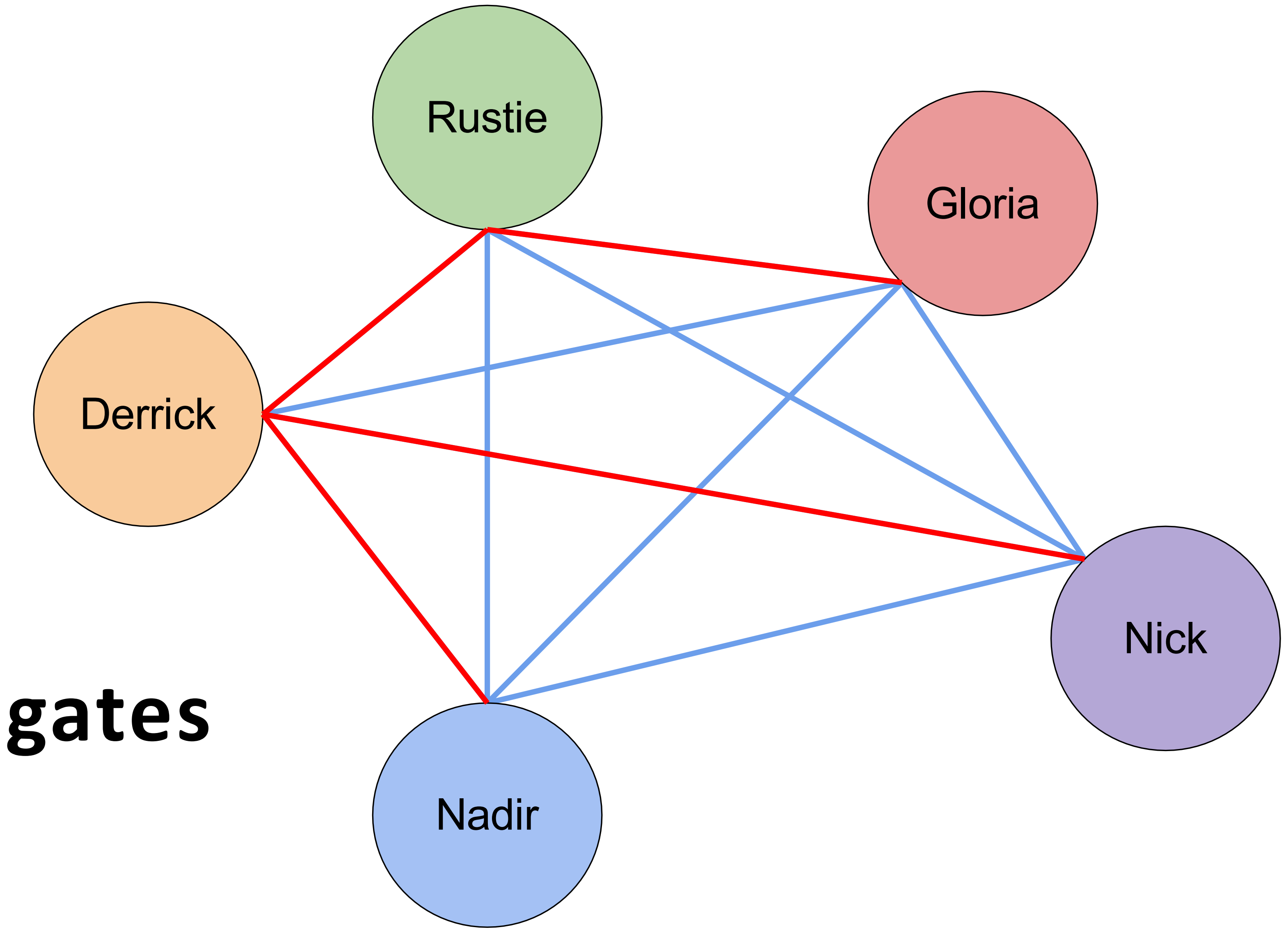
A CLOSER LOOK

What do we mean by p2p?

- Gossip protocol (flooding)
- 8 outbound connections, 117 inbound (pseudorandom) connections

Need to understand how information propagates

- Topology
- Network latency

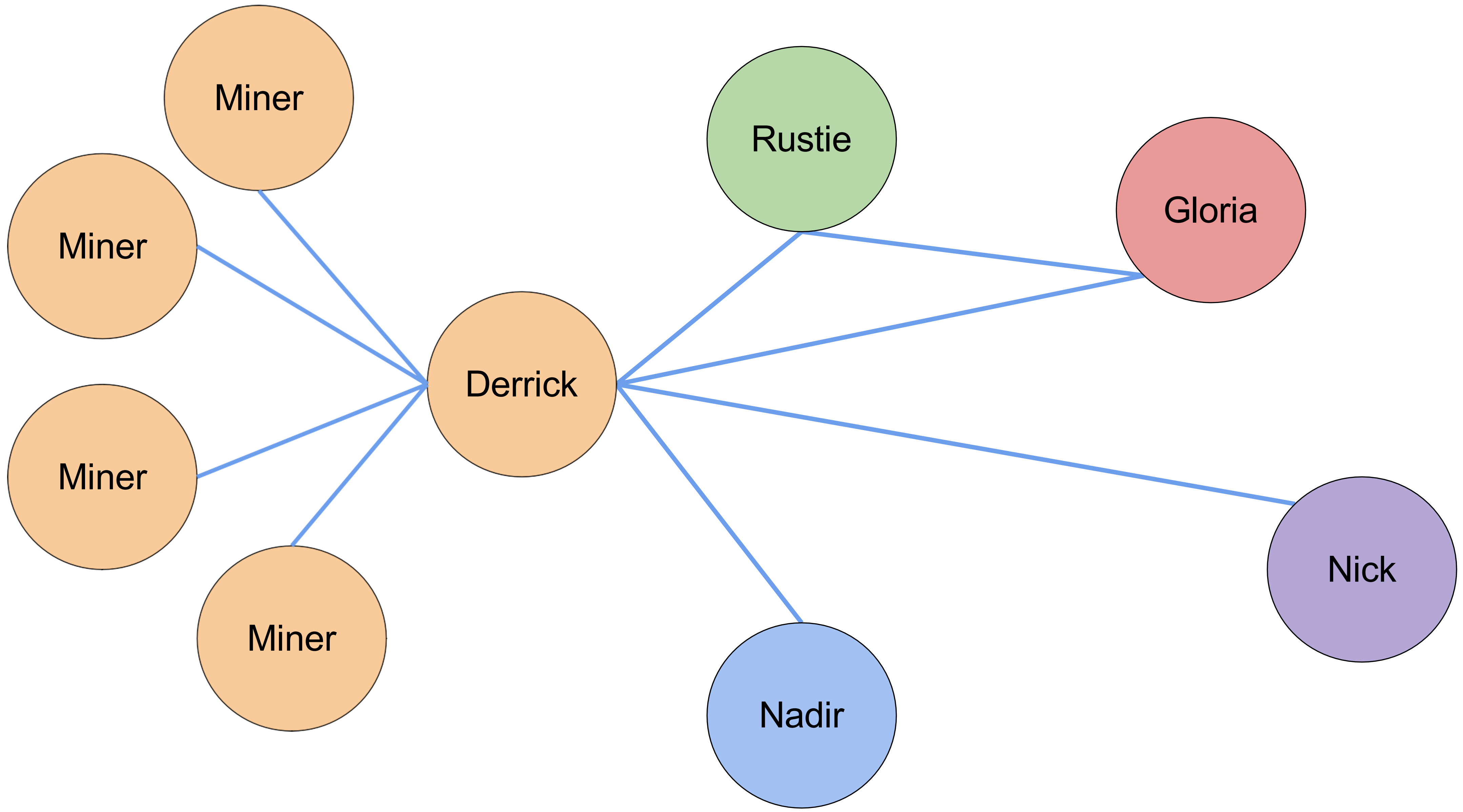


THE BITCOIN NETWORK

AN UNEVEN TOPOLOGY

Some nodes are more influential than others

- Uneven distribution of hash power
- Hard to distinguish network topology

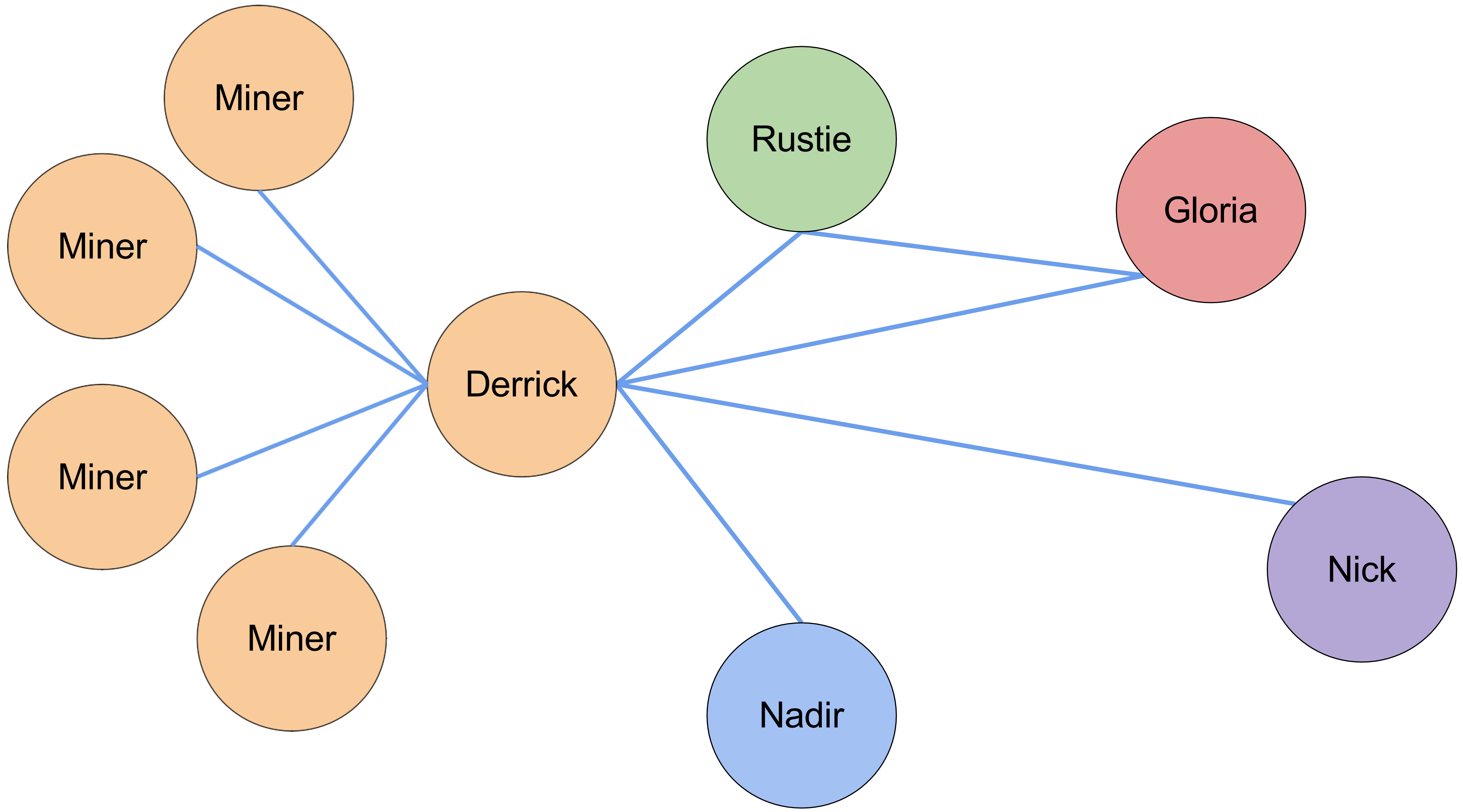


THE BITCOIN NETWORK

AN UNEVEN TOPOLOGY

Bitcoin safe as long as >50% network is honest?

- Incentive alignment
- See selfish mining
- 33%, 25%, 23.2%, 32%, …

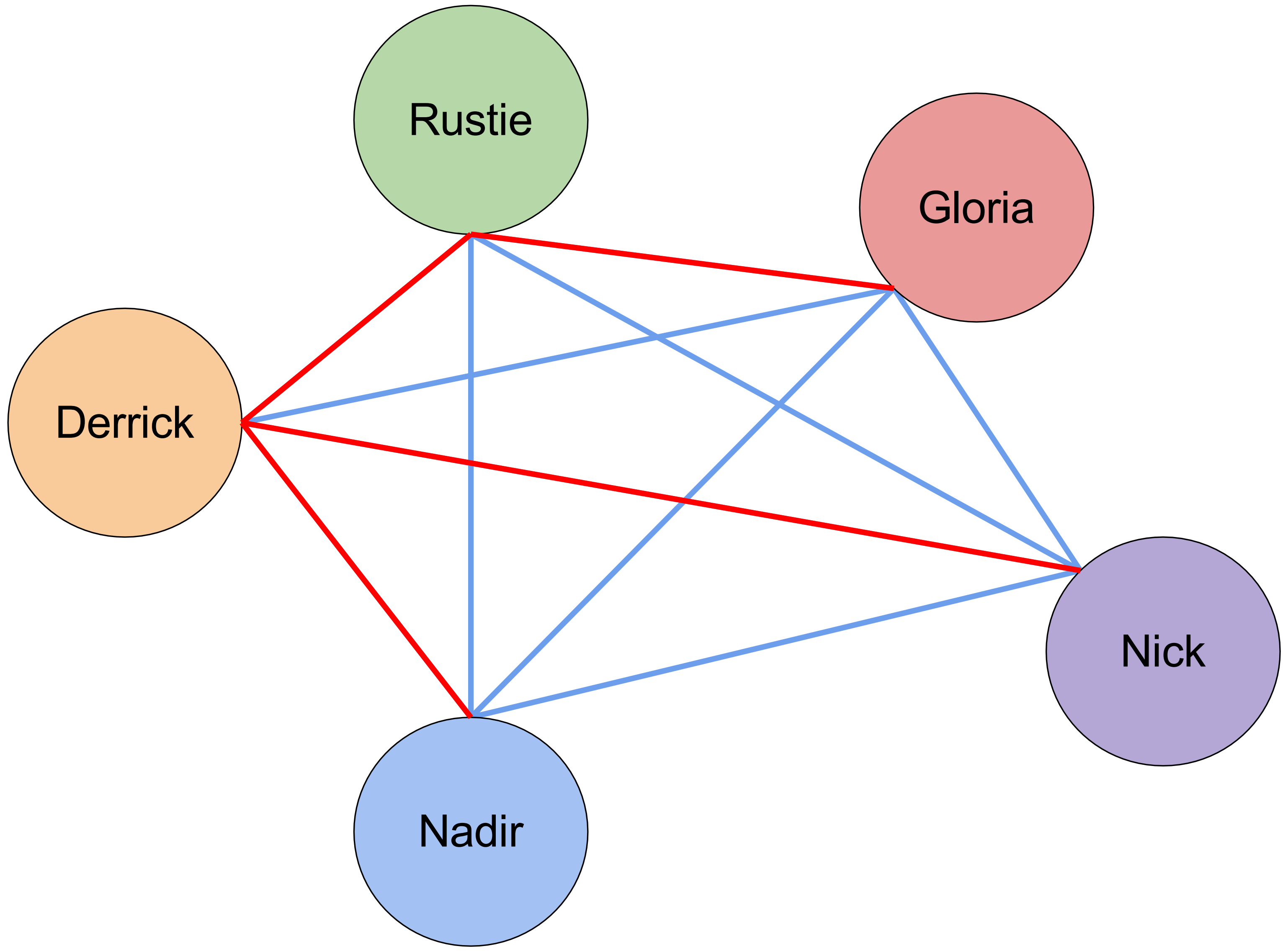


THE BITCOIN NETWORK

NETWORK LATENCY

Different propagation times

- Getting data items out to the network quickly
- Block propagation races
- Leads to disproportionate profits

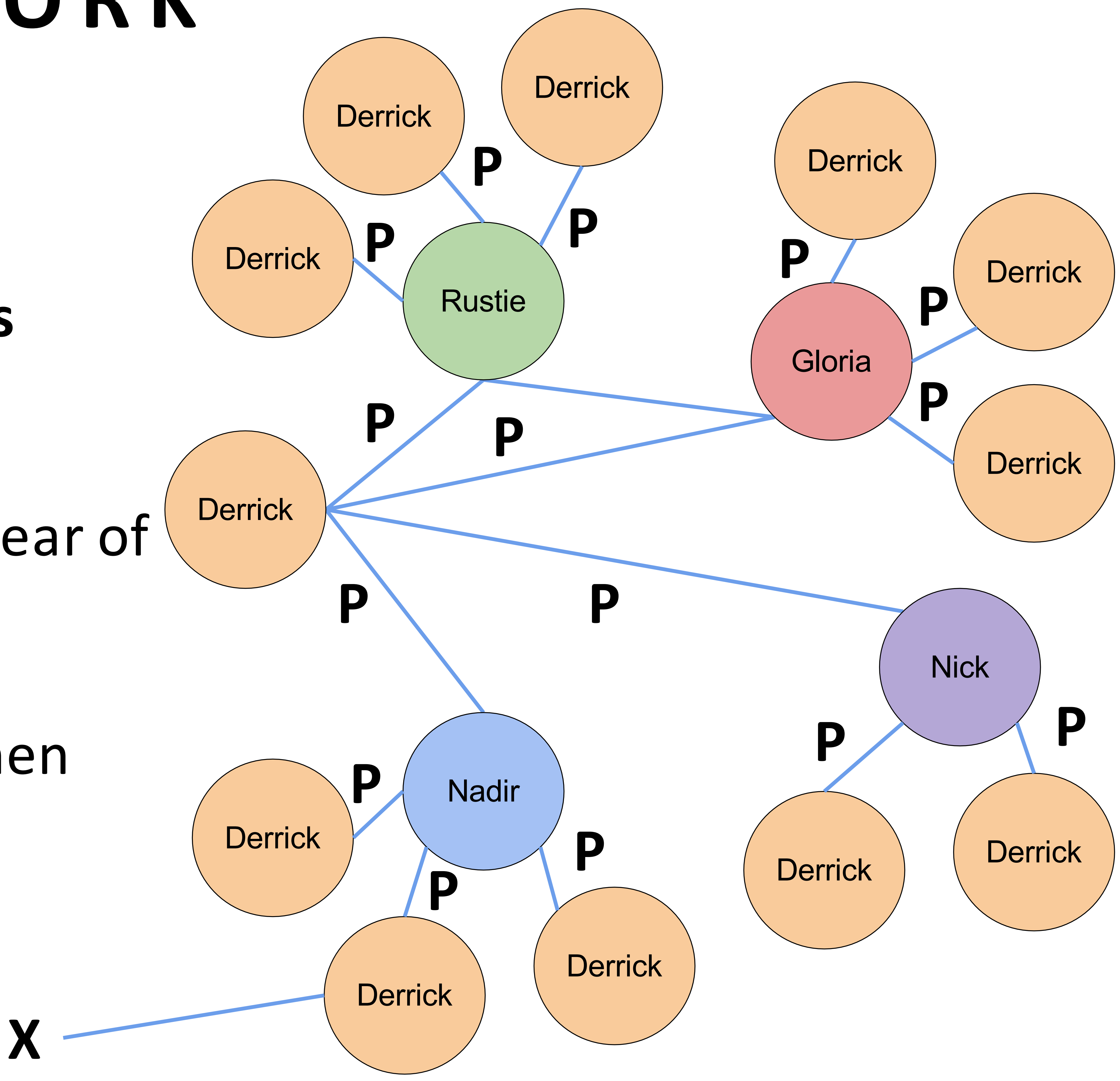


THE BITCOIN NETWORK

SYBIL ATTACK REVISITED

Flood the network with 0-power nodes

- Sybil attack on honest miners
- 0-power nodes as sensors
  - Publish secret block P when hear of competing block X
  - Ignore X, replay P
  - If P reaches miner before X, then mine on P



PROBLEM STATEMENT

How do we prevent selfish mining?

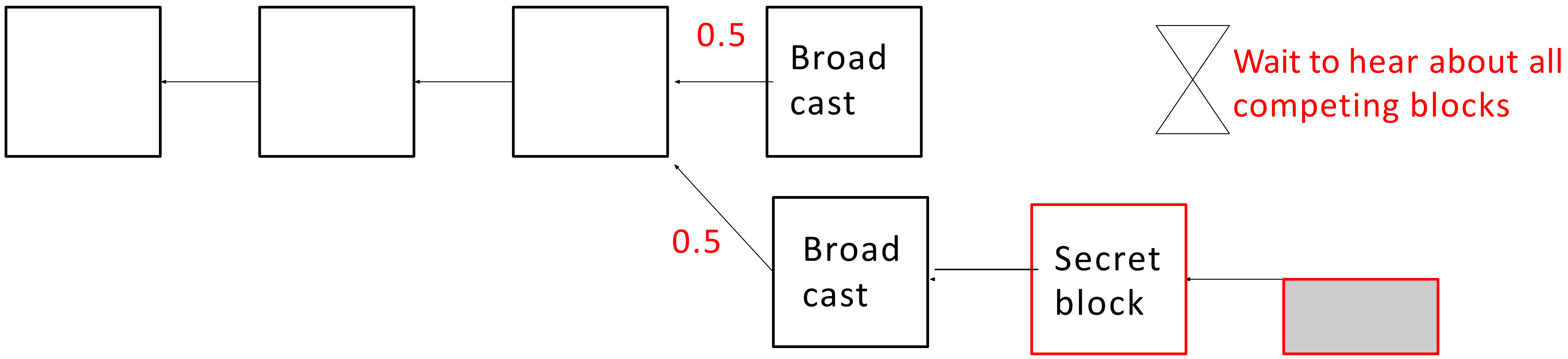
Claim: The problem is that we reward the longest chain.

TIE-BREAKING

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%**

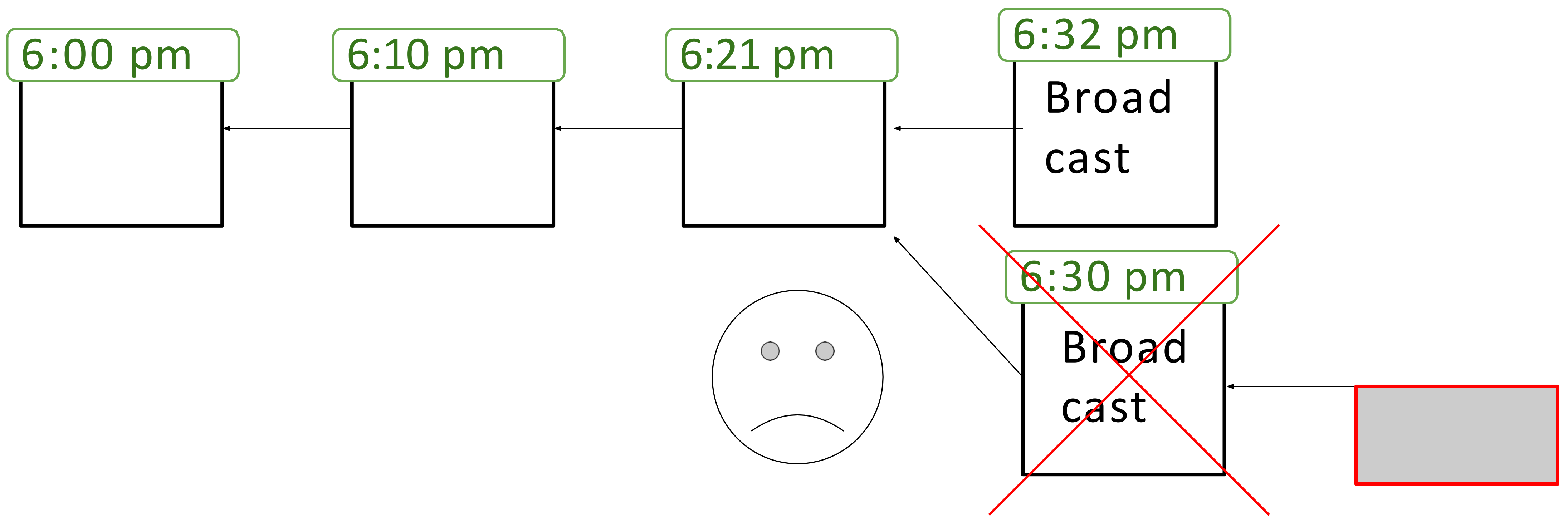


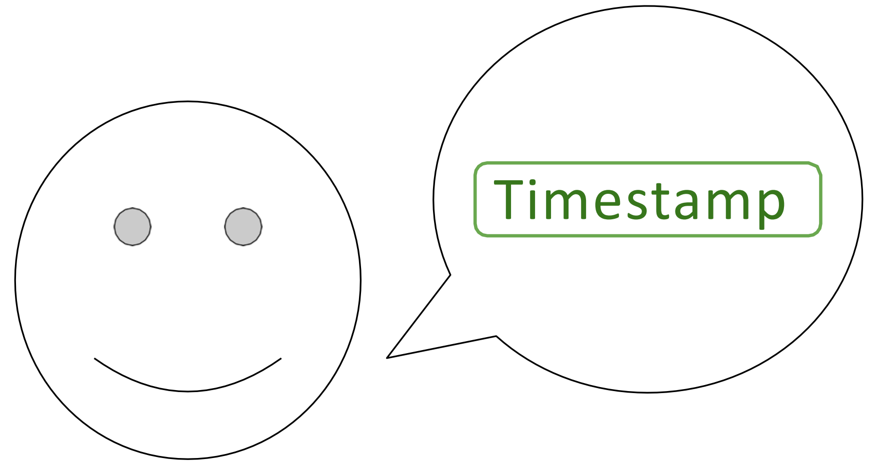
TIE-BREAKING

UNFORGETTABLE TIMESTAMPS

Proposed by Ethan Heilman (2014)

- Each miner incorporates the **latest unforgettable 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%



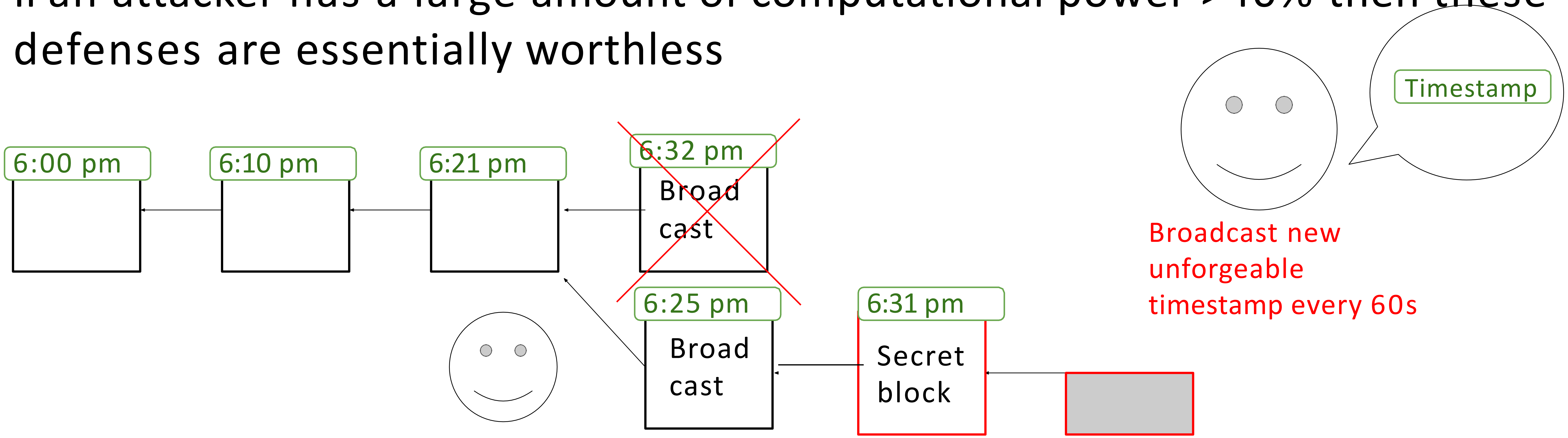


TIE-BREAKING

UNFORGETTABLE TIMESTAMPS

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



TIE-BREAKING

UNFORGETTABLE TIMESTAMPS

Drawbacks

- Introducing trusted third party contradicts the Bitcoin philosophy

+ **CENTRALIZATION**

PROBLEM STATEMENT

How do we disincentivize selfish mining even when the selfish miner has a longer chain?

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 the selfish miner has a longer chain

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

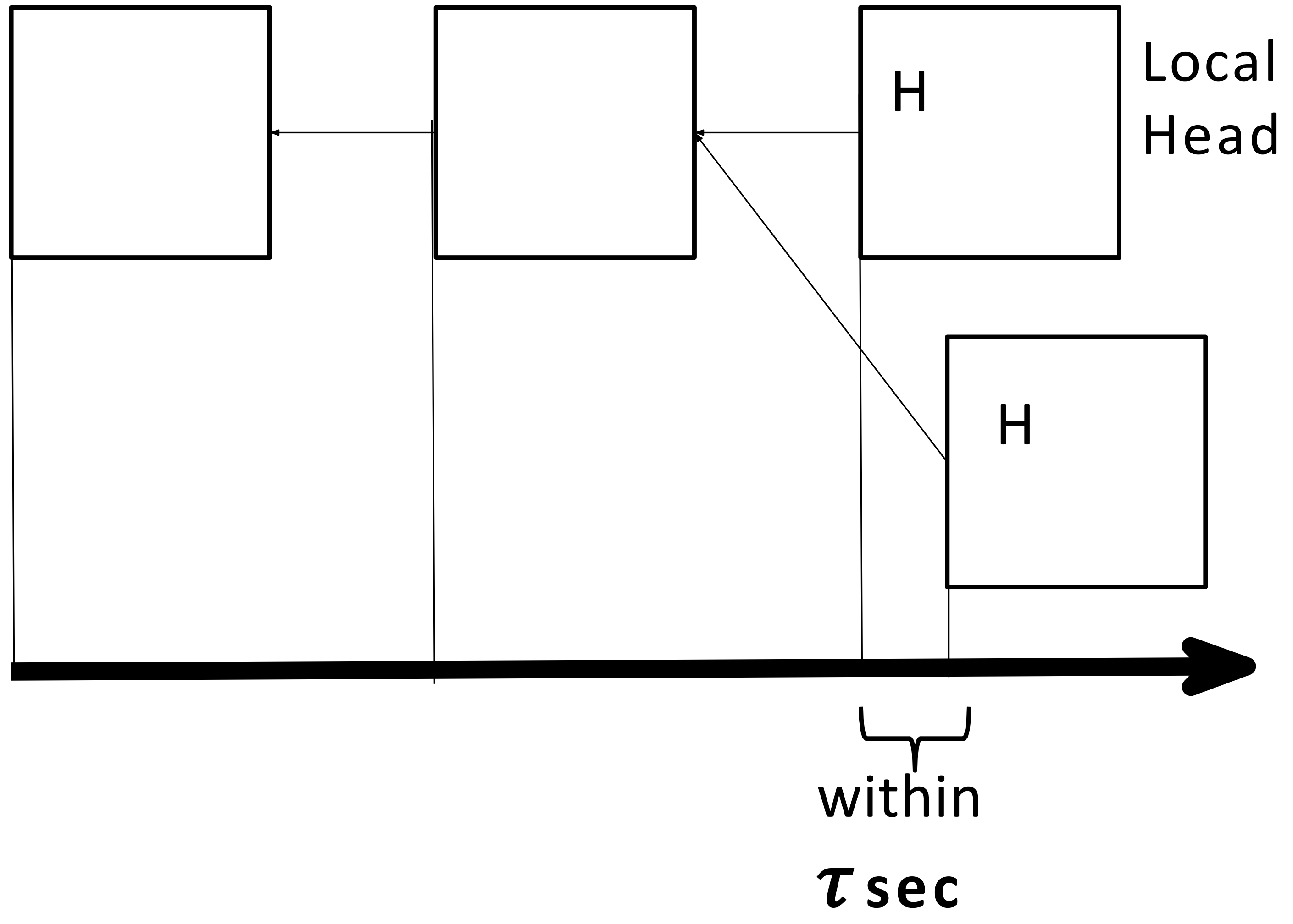
- Replace the original Bitcoin FRP (length FRP), with a **weighted FRP**
  - Embed in the working block the hashes of all its uncle blocks
- Note that selfish mining is premised on the idea of first building a secret block
- Idea: Make sure this secret block does not help the selfish miner win the block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

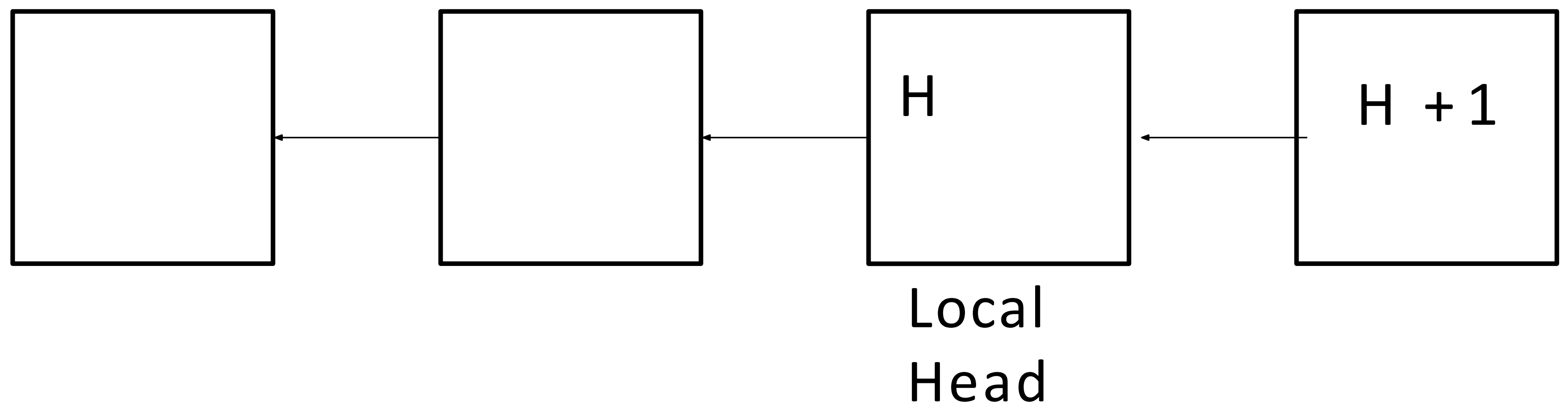
PUBLISH OR PERISH

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

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





PUBLISH OR PERISH

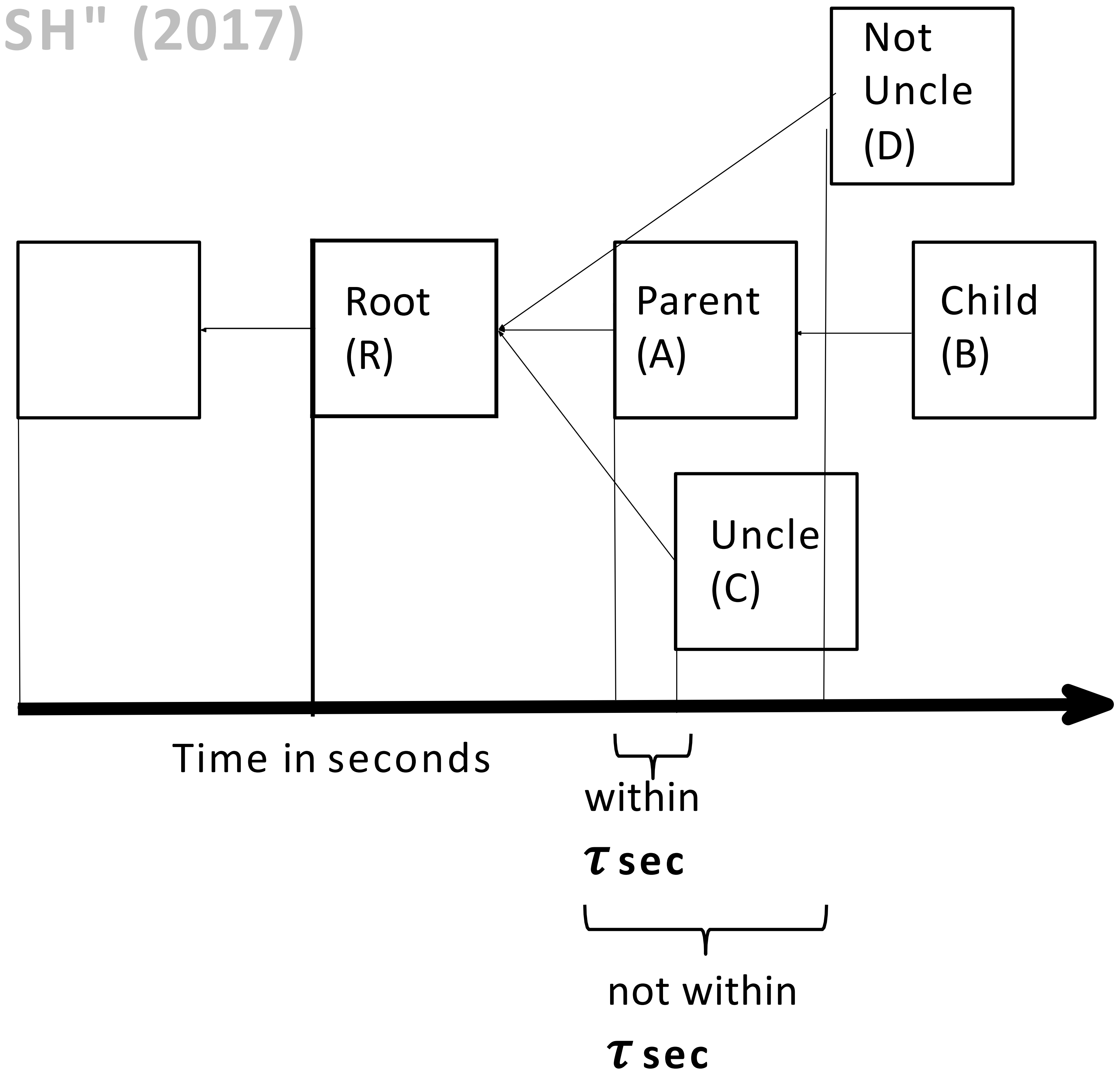
ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

Definitions

- **Uncle**.

  1. The uncle of a block B is one less the height of B
  2. The uncle has to be in time with B's parent

- **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)

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

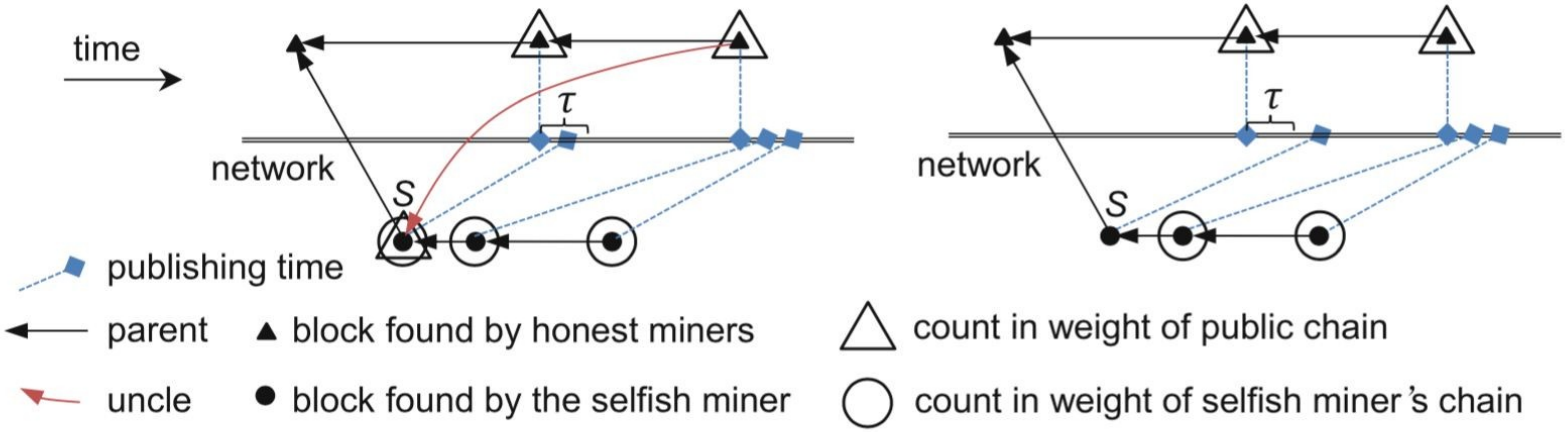
Aside:* ***k* *is a "fail-safe parameter" that gauges the allowed amount of network partition. Note that when* ***k*** *= ∞ the first rule never applies.*

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

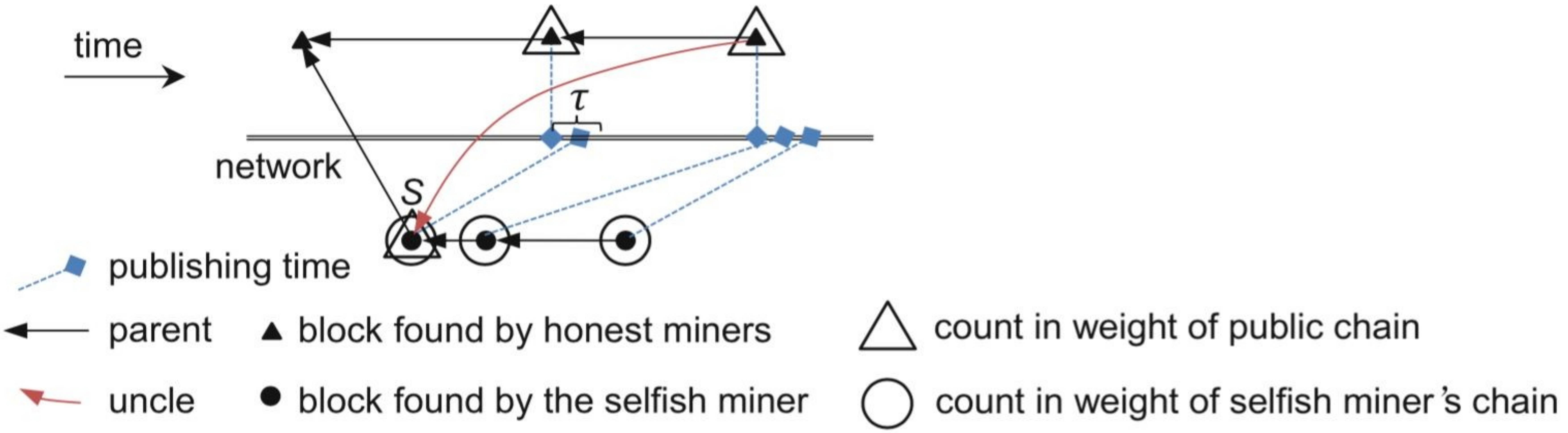
ZHANG AND PRENEEL, "PUBLISH OR PERISH" (2017)

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

Option 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

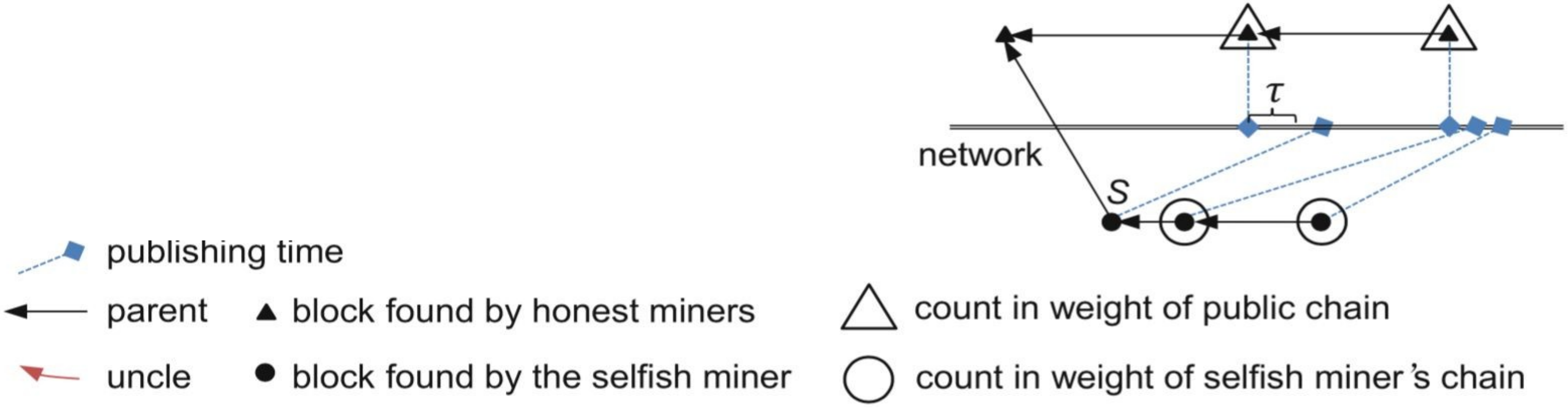
PUBLISH OR PERISH

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

Option 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



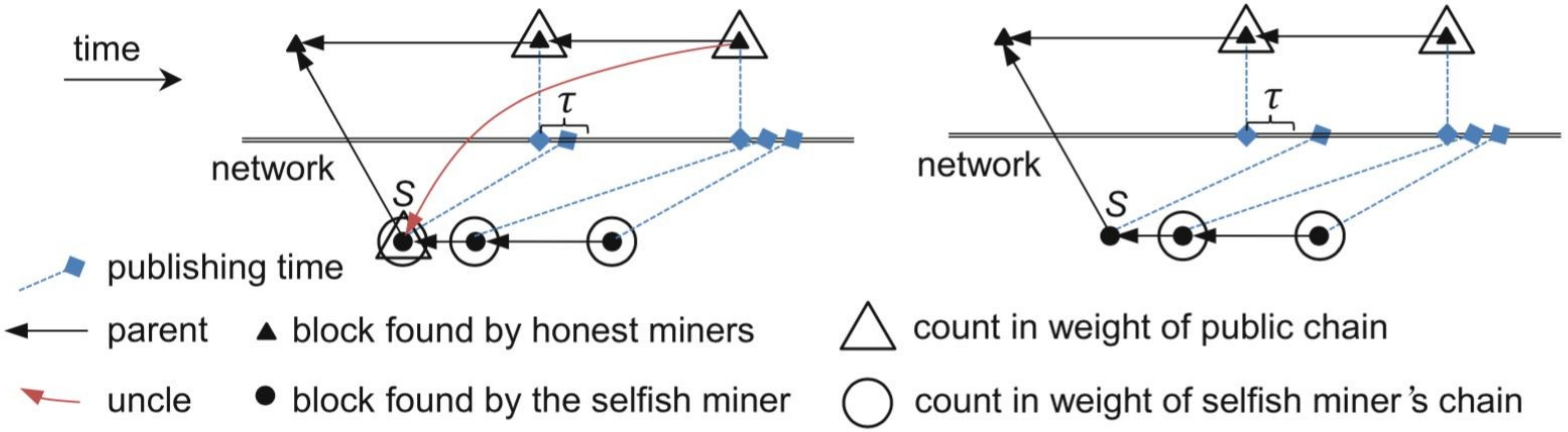
Choice 2: Don't publish

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)

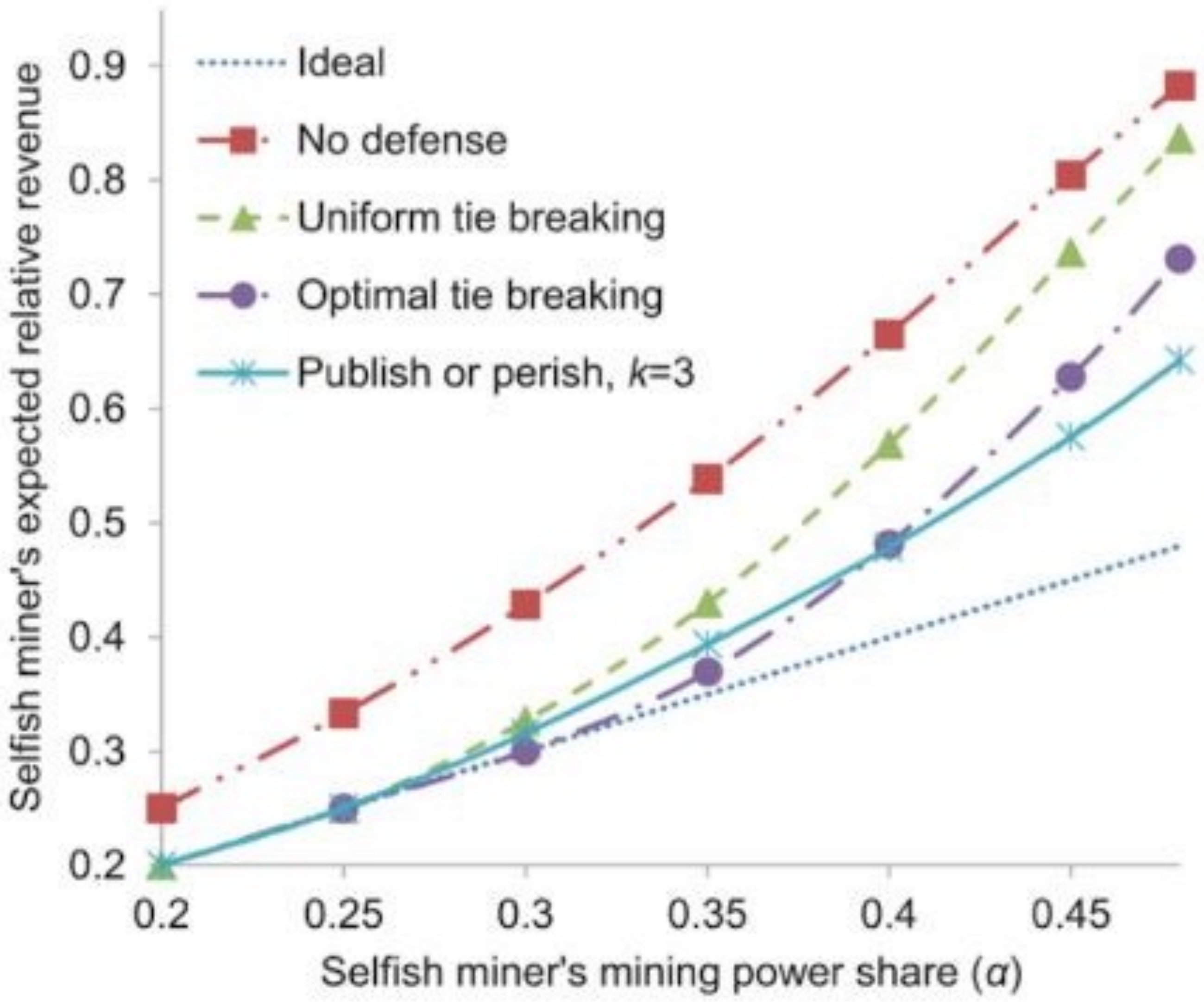


Fig. 5. Comparison with other defenses

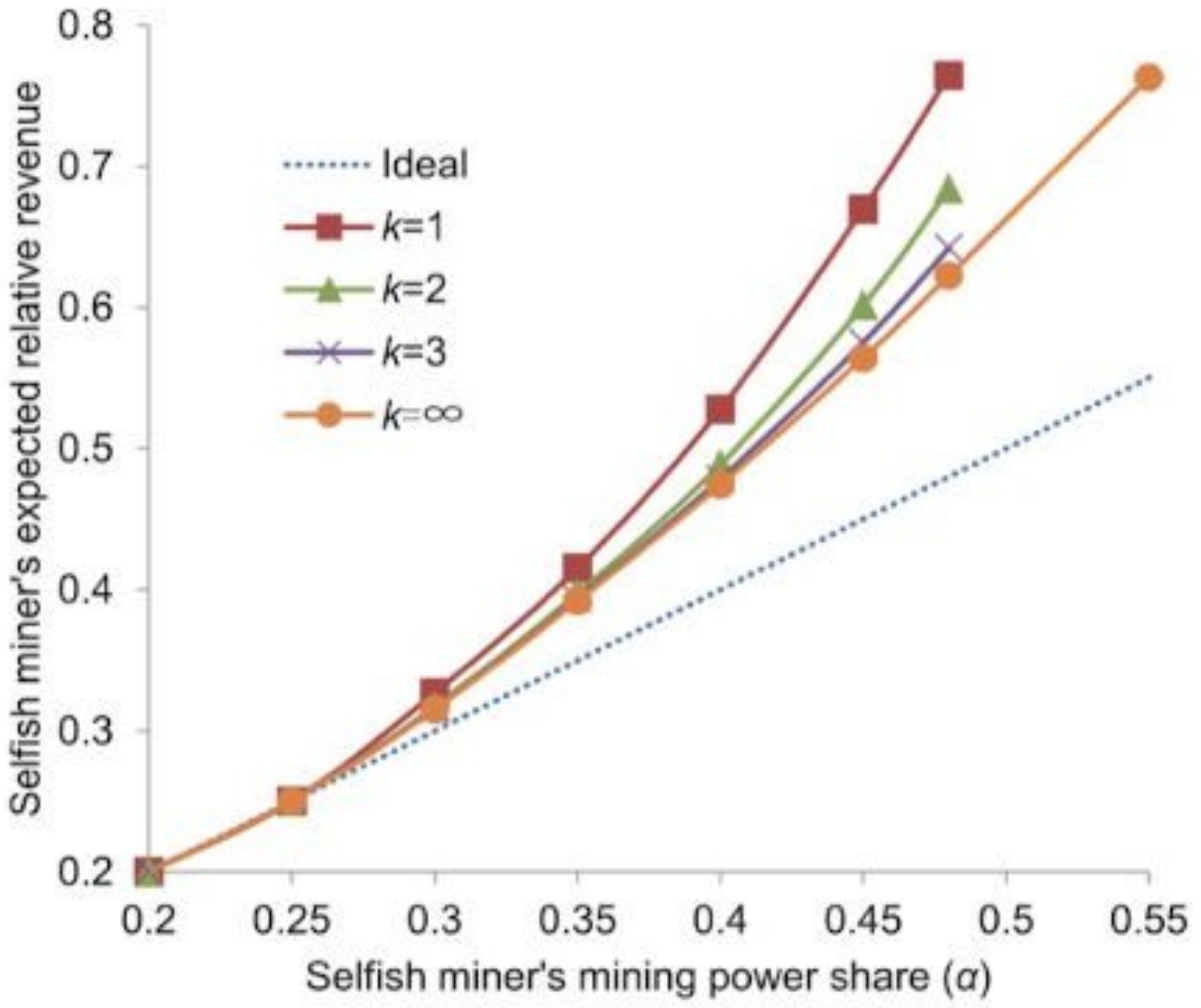


Fig. 4. Relative revenue of the selfish miner within our defense

PUBLISH OR PERISH

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

READINGS

- https://www.coindesk.com/short-guide-blockchain-consensus-protocols
- CAP Theorem - https://www.youtube.com/watch?v=Jw1iFr4v58M

References

Slides mainly adopted from

- Blockchain @ Berkeley : https://blockchain.berkeley.edu/
- Blockchain @ Princeton : http://bitcoinbook.cs.princeton.edu/