Lecture 8

Bitcoin mining

Mining Bitcoins in 6 easy steps

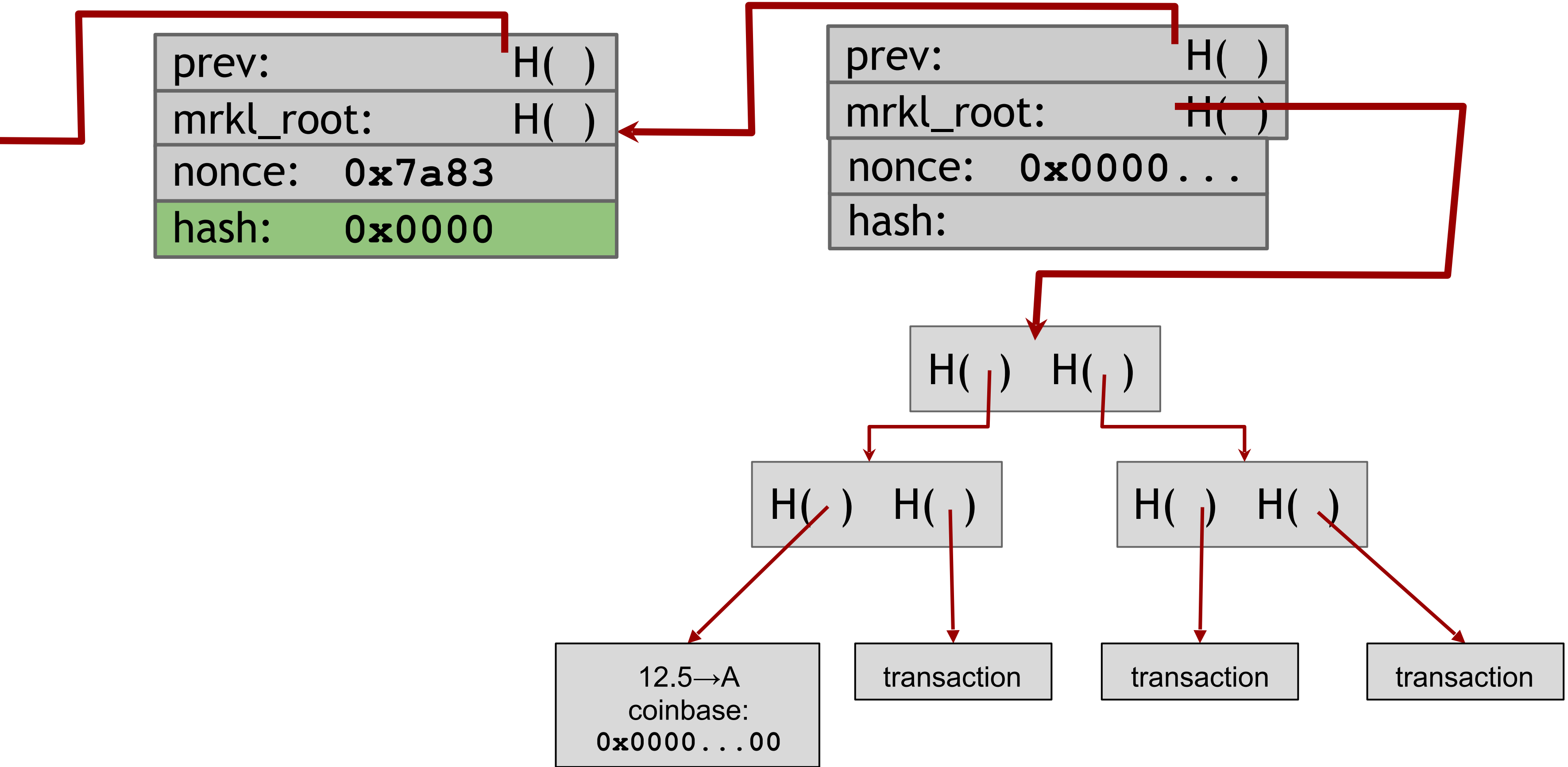
1. Join the network, listen for transactions
    a. Validate all proposed transactions
2. Listen for new blocks, maintain block chain
    a. When a new block is proposed, validate it
3. Assemble a new valid block
4. Find the nonce to make your block valid
5. Hope everybody accepts your new block
6. Money!

Mining Bitcoins in 6 easy steps

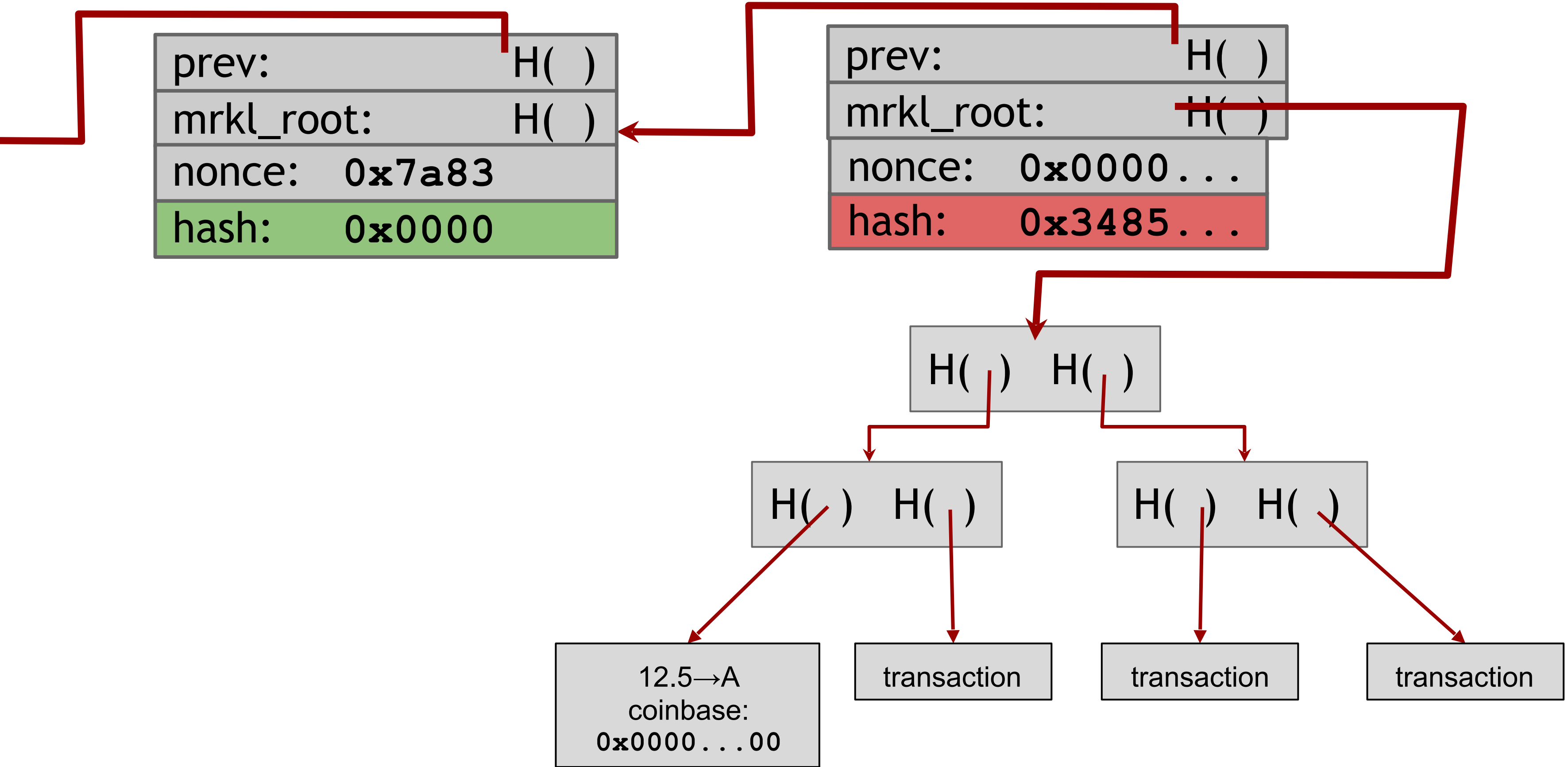
1. Join the network, listen for transactions
    a. Validate all proposed transactions
2. Listen for new blocks, maintain block chain
    a. When a new block is proposed, validate it
3. Assemble a new valid block
4. Find the nonce to make your block valid
5. Hope everybody accepts your new block
6. Money!

Useful to Bitcoin network

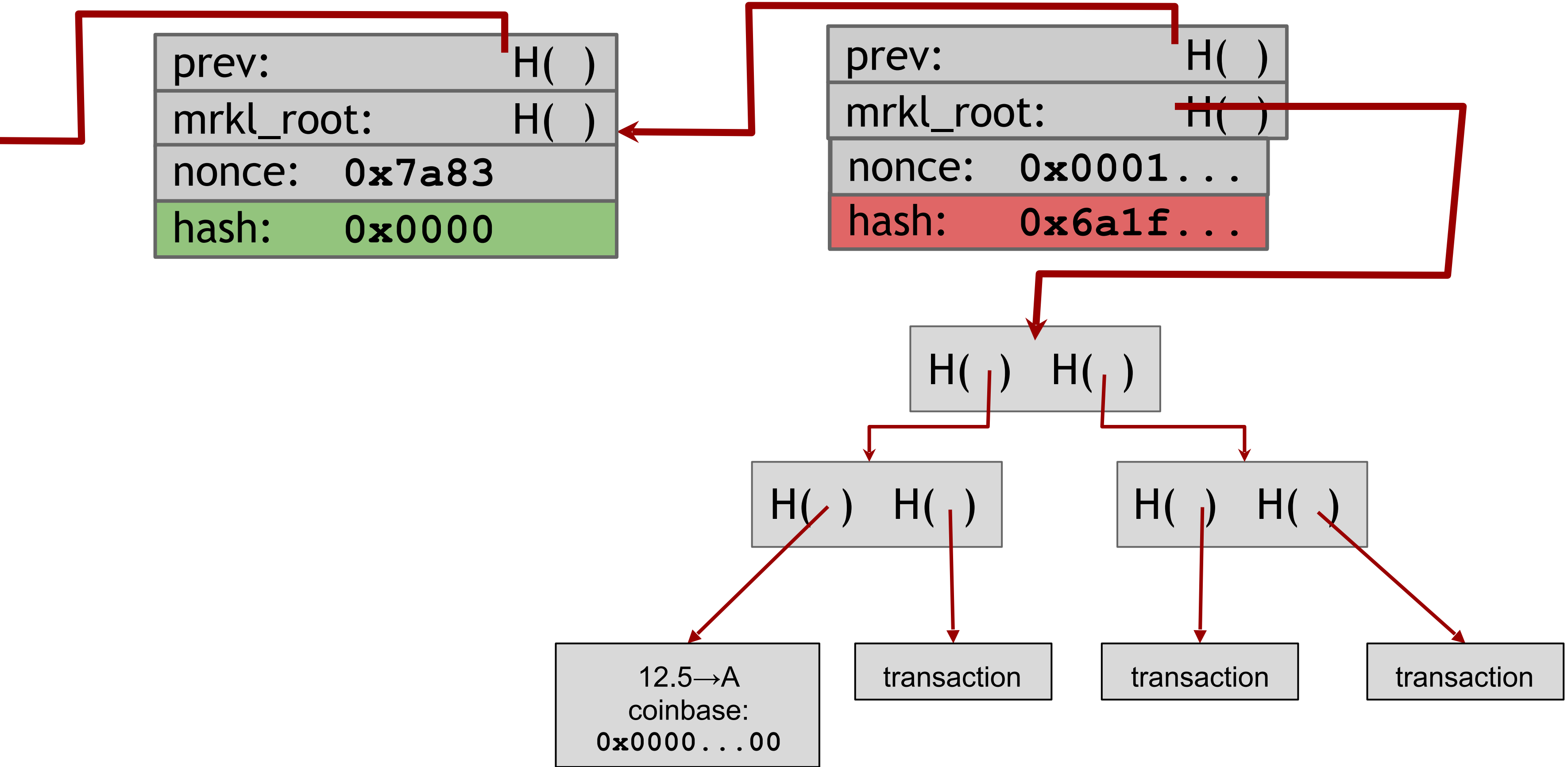
Finding a valid block



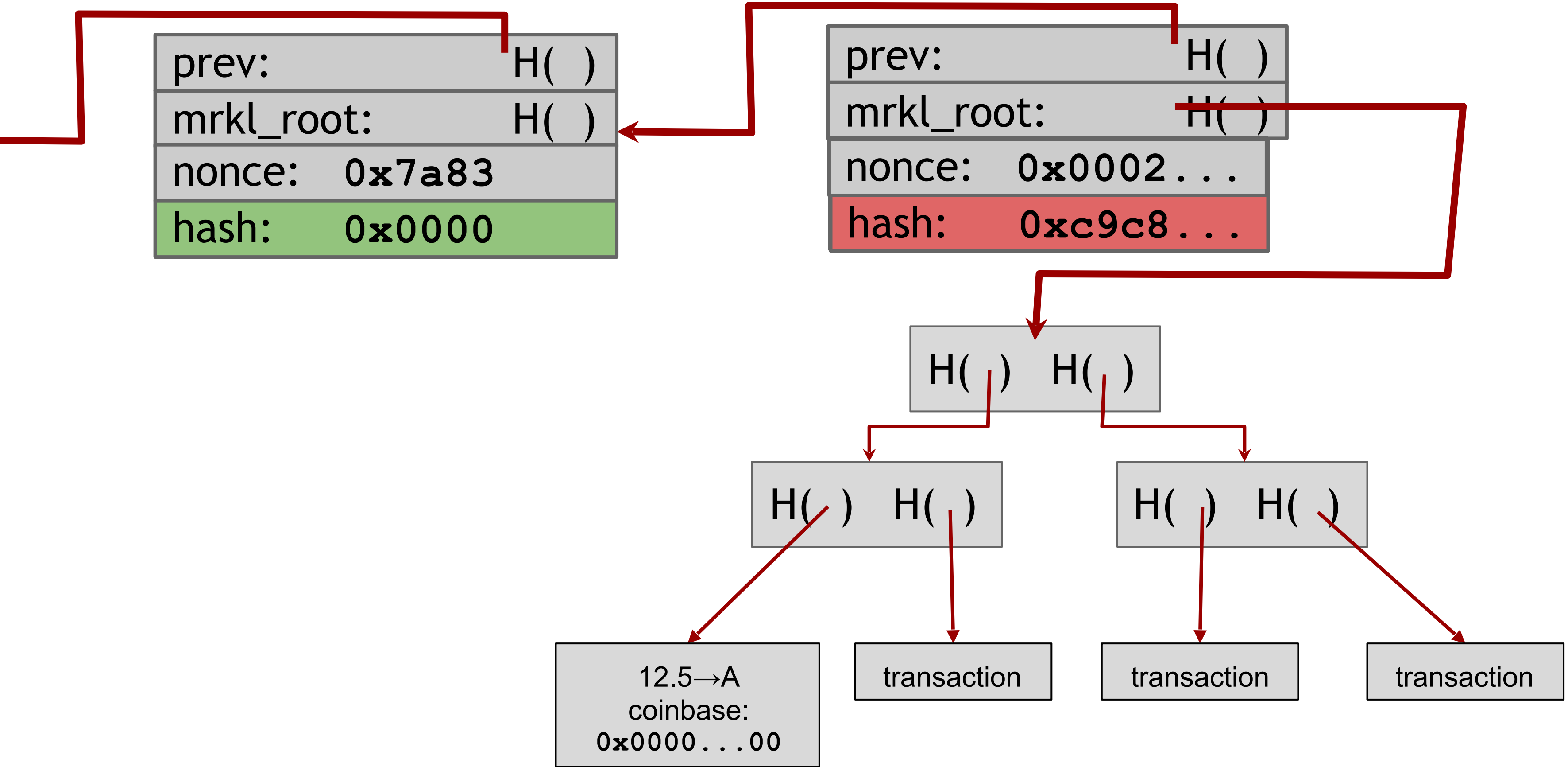
Finding a valid block



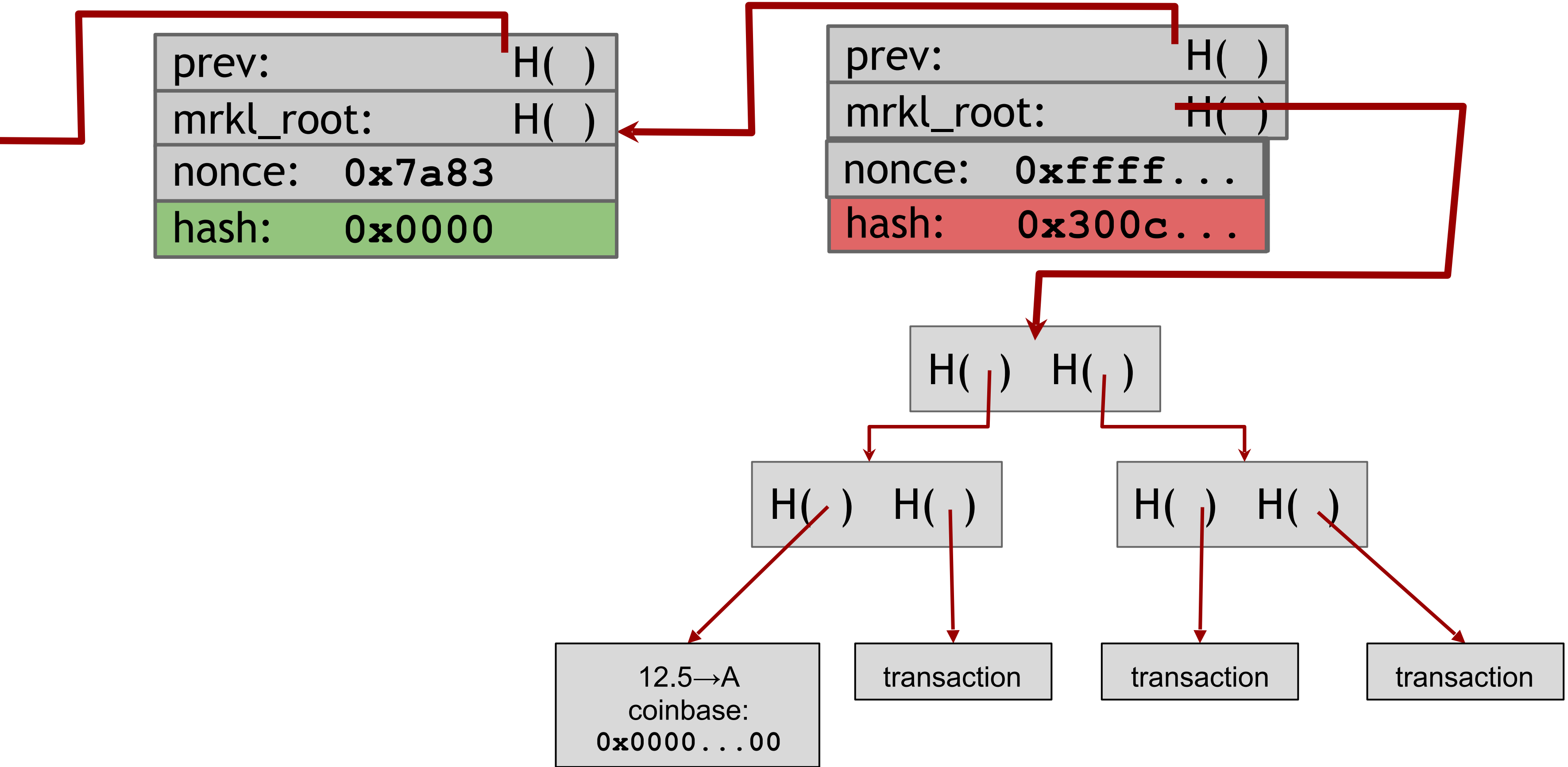
Finding a valid block



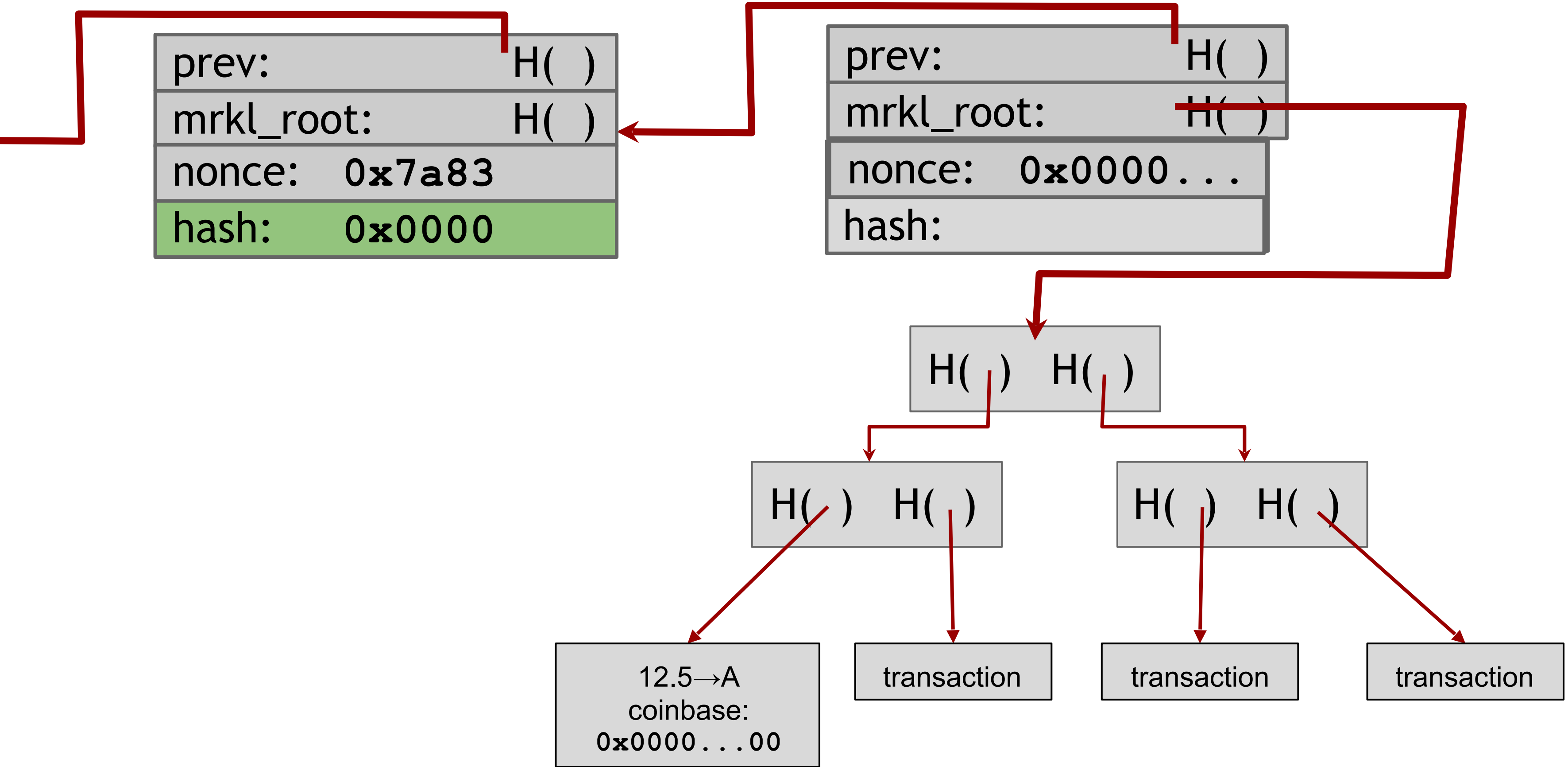
Finding a valid block



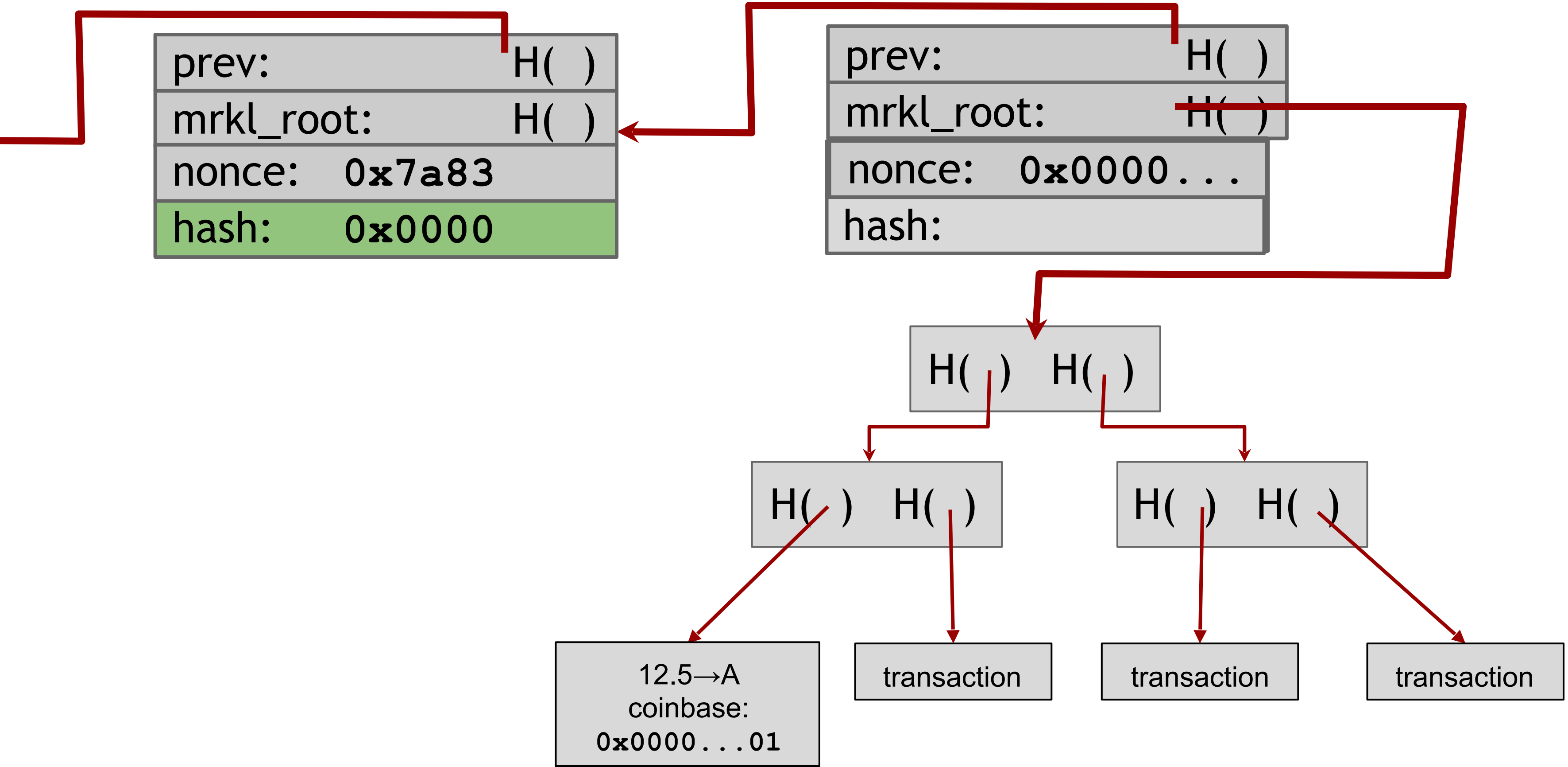
Finding a valid block



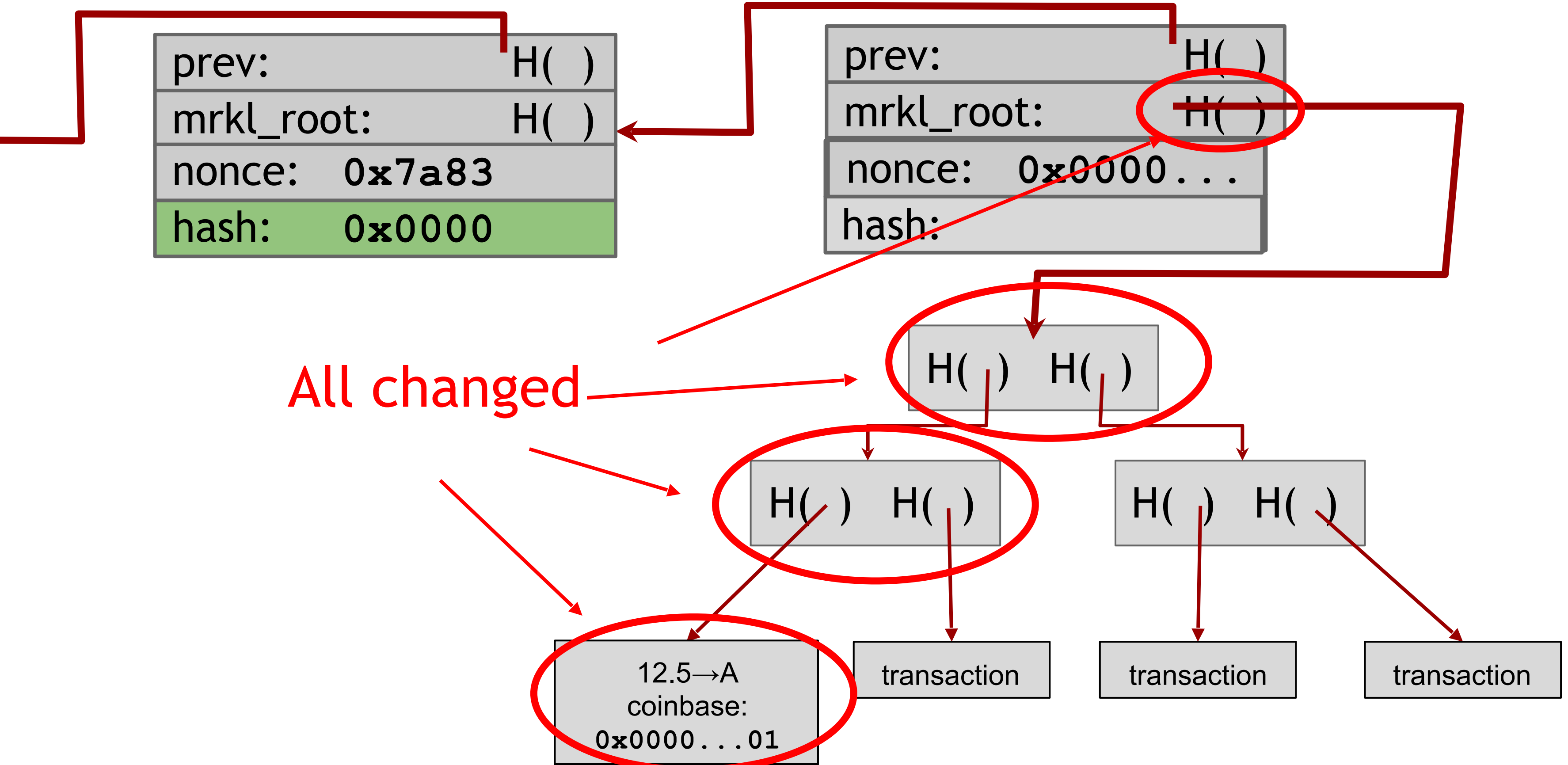
Finding a valid block



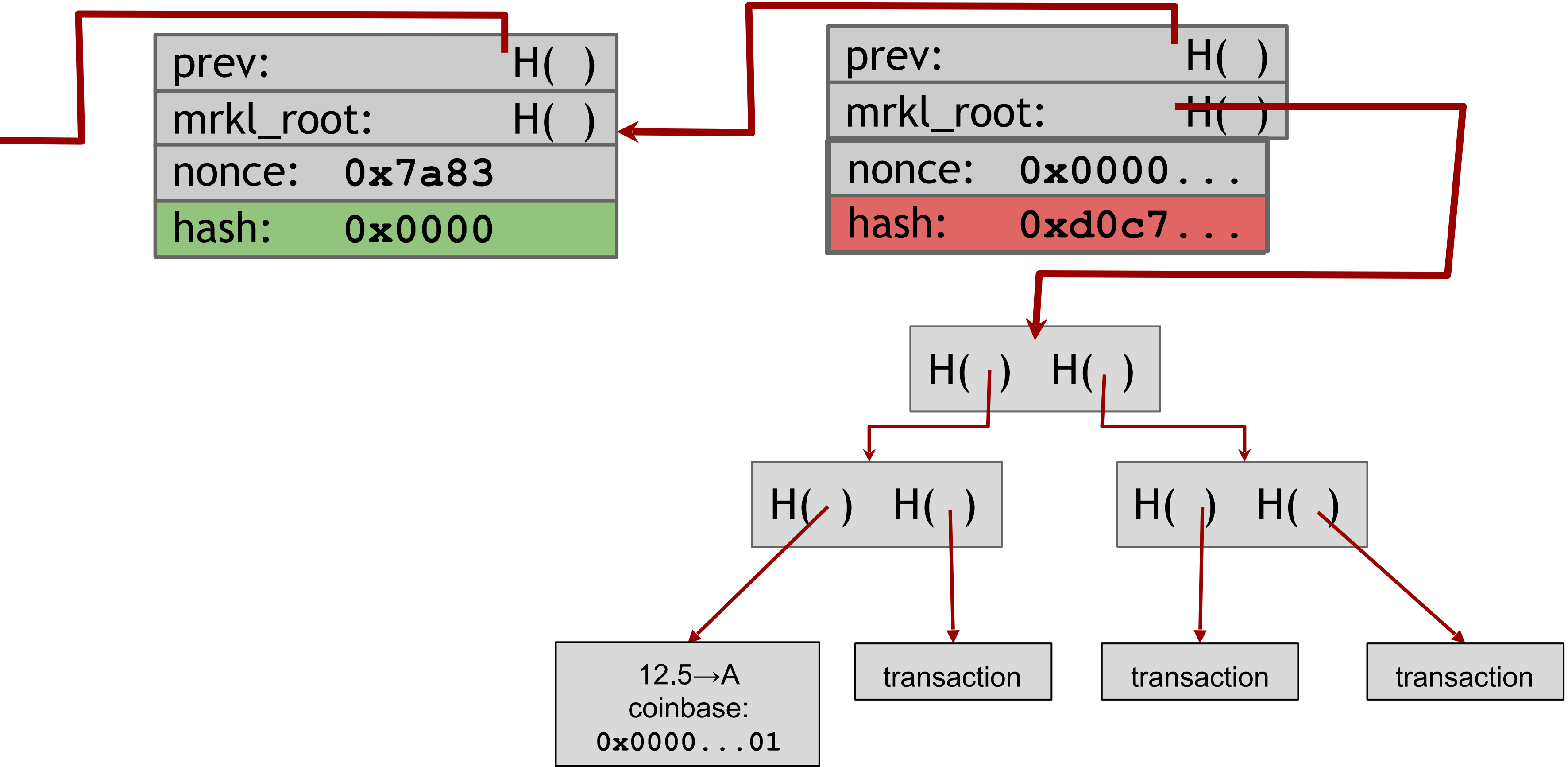
Finding a valid block



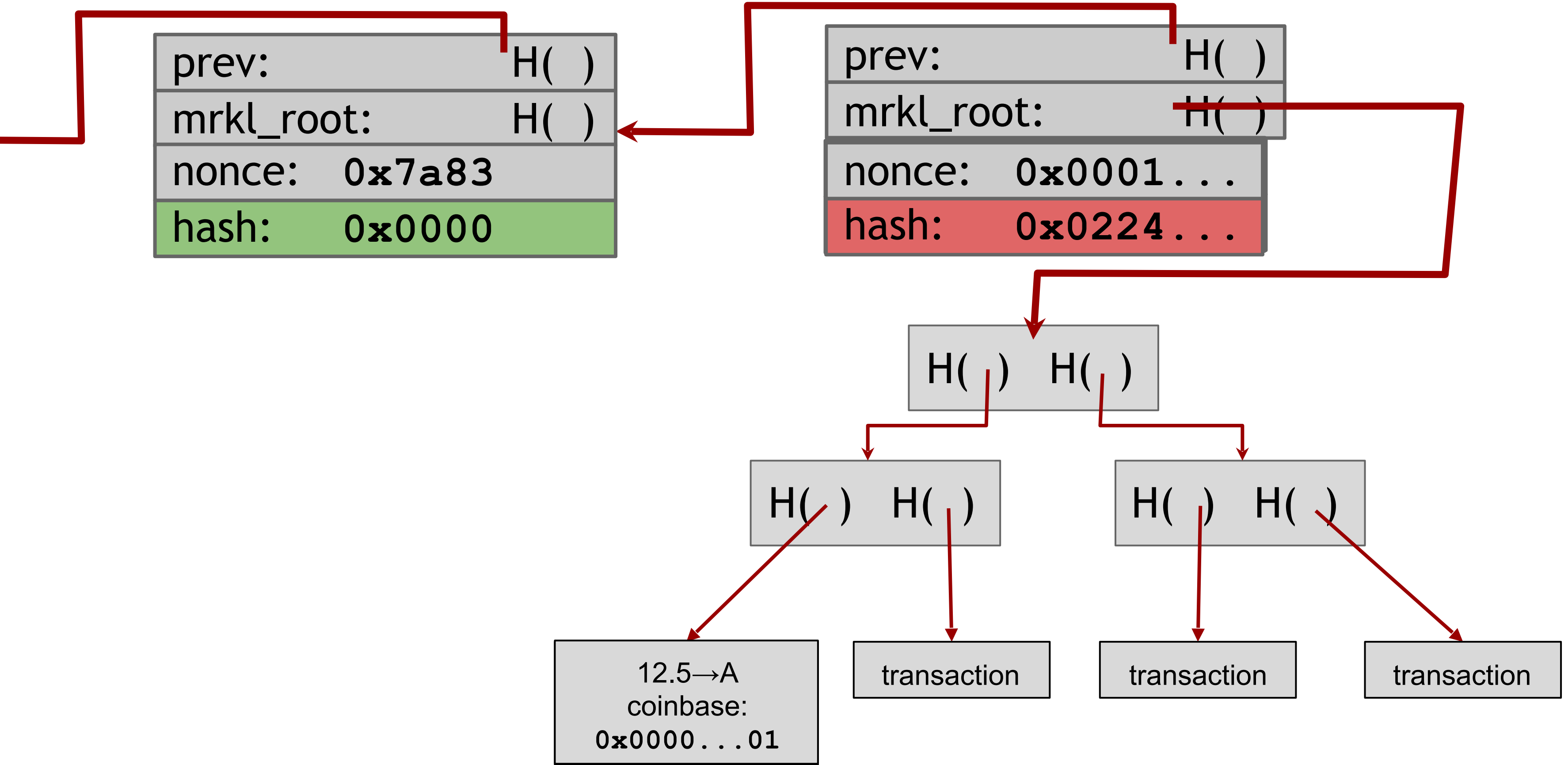
Finding a valid block



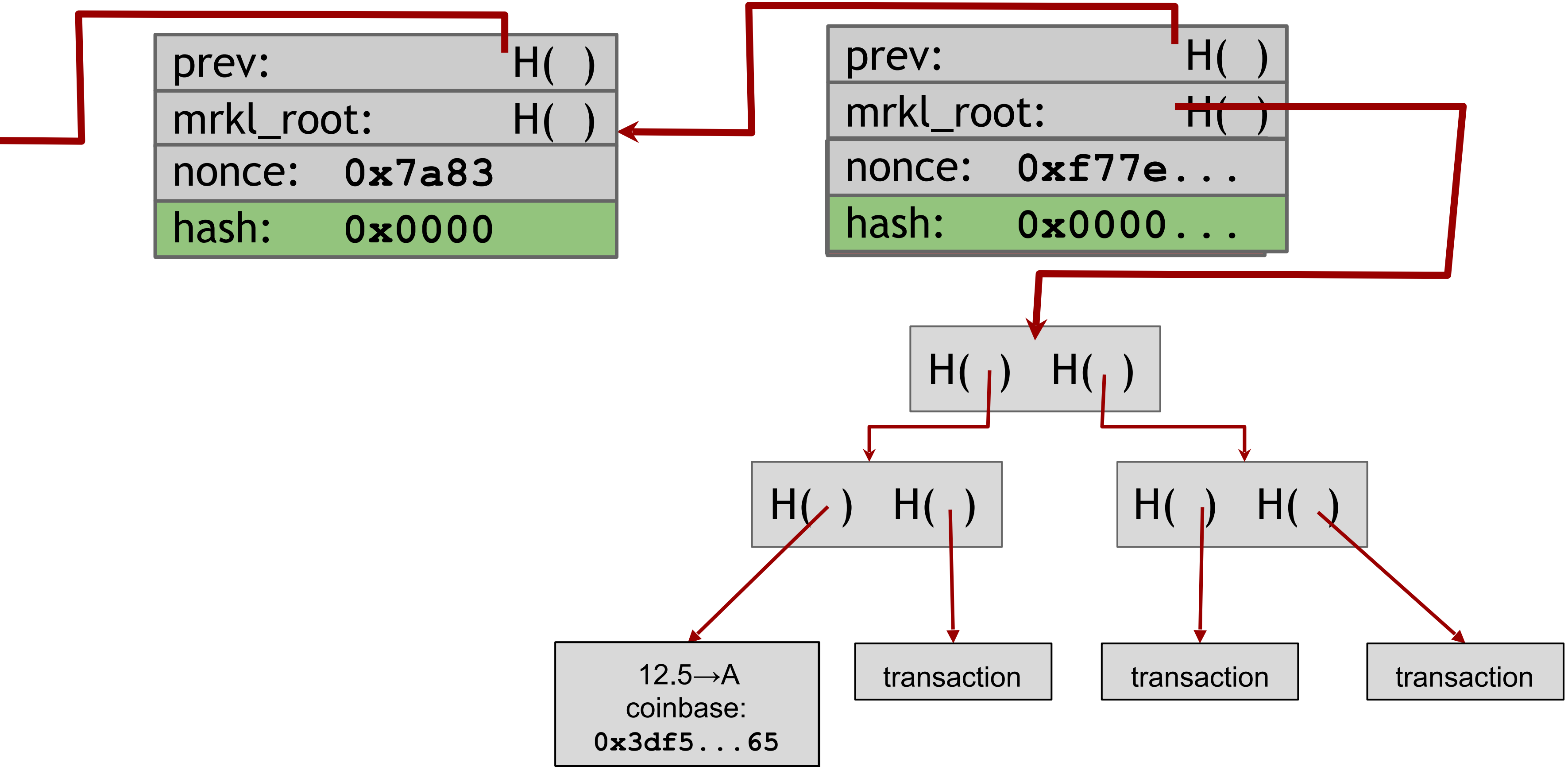
Finding a valid block



Finding a valid block



Finding a valid block



Mining difficulty “target”

000000000000000000**3AAEA2**0000000000000000000000000000000000000000

Mining difficulty “target”

256 bit hash output

$\underbrace{\texttt{0000000000000000003AAEA20000000000000000000000000000000000000000}}$

0000000000000000003AAEA20000000000000000000000000000000000000000

Mining difficulty “target”

256 bit hash output

0000000000000000003AAEA20000000000000000000000000000000000000000

72+ leading zeroes required

Mining difficulty "target"

256 bit hash output

`0000000000000000003AAEA20000000000000000000000000000000000000000`

72+ leading zeroes required

Current difficulty ~ 2^{72}

Setting the mining difficulty

Every two weeks, compute:

```
next_difficulty = previous_difficulty *
                  (2 weeks)/(time to mine last 2016 blocks)
```

Setting the mining difficulty

Every two weeks, compute:

```
next_difficulty = previous_difficulty *
                 (2 weeks)/(time to mine last 2016 blocks)
```

Expected number of blocks in 2 weeks at 10 minutes/block

CPU mining (numbers from 2014)

```
while (1){
    HDR[kNoncePos]++;
    IF (SHA256(SHA256(HDR)) < (65535 << 208)/ DIFFICULTY)
        return;
}
```

CPU mining (numbers from 2014)

```
while (1){
    HDR[kNoncePos]++;
    IF (SHA256(SHA256(HDR)) < (65535 << 208)/ DIFFICULTY)
        return;
}
```

Throughput on a high-end PC = 10-20 MHz ≈ 2^{24}

CPU mining (numbers from 2014)

```
while (1){
    HDR[kNoncePos]++;
    IF (SHA256(SHA256(HDR)) < (65535 << 208)/ DIFFICULTY)
        return;
}
```

Throughput on a high-end PC = 10-**20 MHz** ≈ 2^{24}

139,461 years to find a block!

Evolution of mining

CPU

GPU

FPGA

ASIC

Evolution of mining

CPU

GPU

FPGA

ASIC

Huge energy consumption (in 2017, annual rate nearly as high as Denmark)!

The future

- Can small miners stay in the game?
- Would we be better off without ASICs?
- Should we implement consensus without proofs of work?

The future

- Can small miners stay in the game?
- Would we be better off without ASICs?
- Should we implement consensus without proofs of work?

Motivation for Altcoins

Mining pools

Economics of being a small miner

- In 2014, expected revenue: ≈$1,000/month
- High probability (~50%) of not mining a block within a year

Mining pools

- **Goal:** pool participants all attempt to mine a block with the same coinbase recipient
  - send money to key owned by pool manager
- Distribute revenues to members based on how much work they have performed
  - minus a cut for pool manager

Mining pools

- **Goal:** pool participants all attempt to mine a block with the same coinbase recipient
  - send money to key owned by pool manager
- Distribute revenues to members based on how much work they have performed
  - minus a cut for pool manager

How do we know how much work members perform?

Mining shares

Idea: prove work with "near-valid blocks" (shares)

```
4AA087F0A52ED2093FA816E53B9B6317F9B8C1227A61F9481AFED67301F2E3FB
D3E51477DCAB108750A5BC9093F6510759CC880BB171A5B77FB4A34ACA27DEDD
0000000008534FF68B98935D090DF5669E3403BD16F1CDFD41CF17D6B474255
BB34ECA3DBB52EFF4B104EBBC0974841EF2F3A59EBBC4474A12F9F595EB81F4B
0000000002F891C1E232F687E41515637F7699EA0F462C2564233FE082BB0AF
009048813377 9E7E98177AF1C765CF02D01AB4848DF555533B6C4CFCA201CBA1
460BEFA43B7083E502D36D9D08D64AFB99A100B3B80D4EA4F7B38E18174A0BFB
000000000000000078FB7E1F7E2E4854B8BC71412197EB1448911FA77BAE808A
652F374601D149AC47E01E77761384561 81FA4F9D0EEDD8C4FDE3BEF6B1B7ECE
78552640214 3A291CFD60DA09CC80DD066BC723FD5FD20F9B50D614313529AF3
00000000041EE59343468 6000AF77F54CDE839A6CE30957B14EDEC10B15C9E5
9C20B06B01A0136F192BD48E0F372A4B9E6BA6ABC36F02FCED22FD9780026A8F
```

Mining shares

Idea: prove work with "near-valid blocks" (shares)

```
4AA087F0A52ED2093FA816E53B9B6317F9B8C1227A61F9481AFED67301F2E3FB
D3E51477DCAB108750A5BC9093F6510759CC880BB171A5B77FB4A34ACA27DEDD
0000000008534FF68B98935D090DF5669E3403BD16F1CDFD41CF17D6B474255
BB34ECA3DBB52EFF4B104EBBC0974841EF2F3A59EBBC4474A12F9F595EB81F4B
0000000002F891C1E232F687E41515637F7699EA0F462C2564233FE082BB0AF
009048813377 9E7E98177AF1C765CF02D01AB4848DF555533B6C4CFCA201CBA1
460BEFA43B7083E502D36D9D08D64AFB99A100B3B80D4EA4F7B38E18174A0BFB
00000000000000078FB7E1F7E2E4854B8BC71412197EB1448911FA77BAE808A
652F374601D149AC47E01E7776138456181FA4F9D0EEDD8C4FDE3BEF6B1B7ECE
78552640214 3A291CFD60DA09CC80DD066BC723FD5FD20F9B50D614313529AF3
00000000041EE59343468 6000AF77F54CDE839A6CE30957B14EDEC10B15C9E5
9C20B06B01A0136F192BD48E0F372A4B9E6BA6ABC36F02FCED22FD9780026A8F
```

Mining shares

Idea: prove work with “near-valid blocks” (shares)

```
4AA087F0A52ED2093FA816E53B9B6317F9B8C1227A61F9481AFED67301F2E3FB
D3E51477DCAB108750A5BC9093F6510759CC880BB171A5B77FB4A34ACA27DEDD
00000000008534FF68B98935D090DF5669E3403BD16F1CDFD41CF17D6B474255
BB34ECA3DBB52EFF4B104EBBC0974841EF2F3A59EBBC4474A12F9F595EB81F4B
00000000002F891C1E232F687E41515637F7699EA0F462C2564233FE082BB0AF
0090488133779E7E98177AF1C765CF02D01AB4848DF555533B6C4CFCA201CBA1
460BEFA43B7083E502D36D9D08D64AFB99A100B3B80D4EA4F7B38E18174A0BFB
000000000000000078FB7E1F7E2E4854B8BC71412197EB1448911FA77BAE808A
652F374601D149AC47E01E7776138456181FA4F9D0EEDD8C4FDE3BEF6B1B7ECE
78552640214 3A291CFD60DA09CC80DD066BC723FD5FD20F9B50D614313529AF3
000000000041EE59343468600 0AF77F54CDE839A6CE30957B14EDEC10B15C9E5
9C20B06B01A0136F192BD48E0F372A4B9E6BA6ABC36F02FCED22FD9780026A8F
```

Mining pools

Pool manager

Mining pools

Pool manager

Hey folks! Here's our next block to work on

Mining pools

Pool manager

prev:	H( )
mrkl_root:	H( )
nonce:	
hash:	

Mining pools

Pool manager

prev:	H( )
mrkl_root:	H( )
nonce:	
hash:	

coinbase:
12.5→pool

Mining pools

Pool manager

Mining pools

Pool manager

0x00000000000a877902e...

0x000000000001e8709ce...

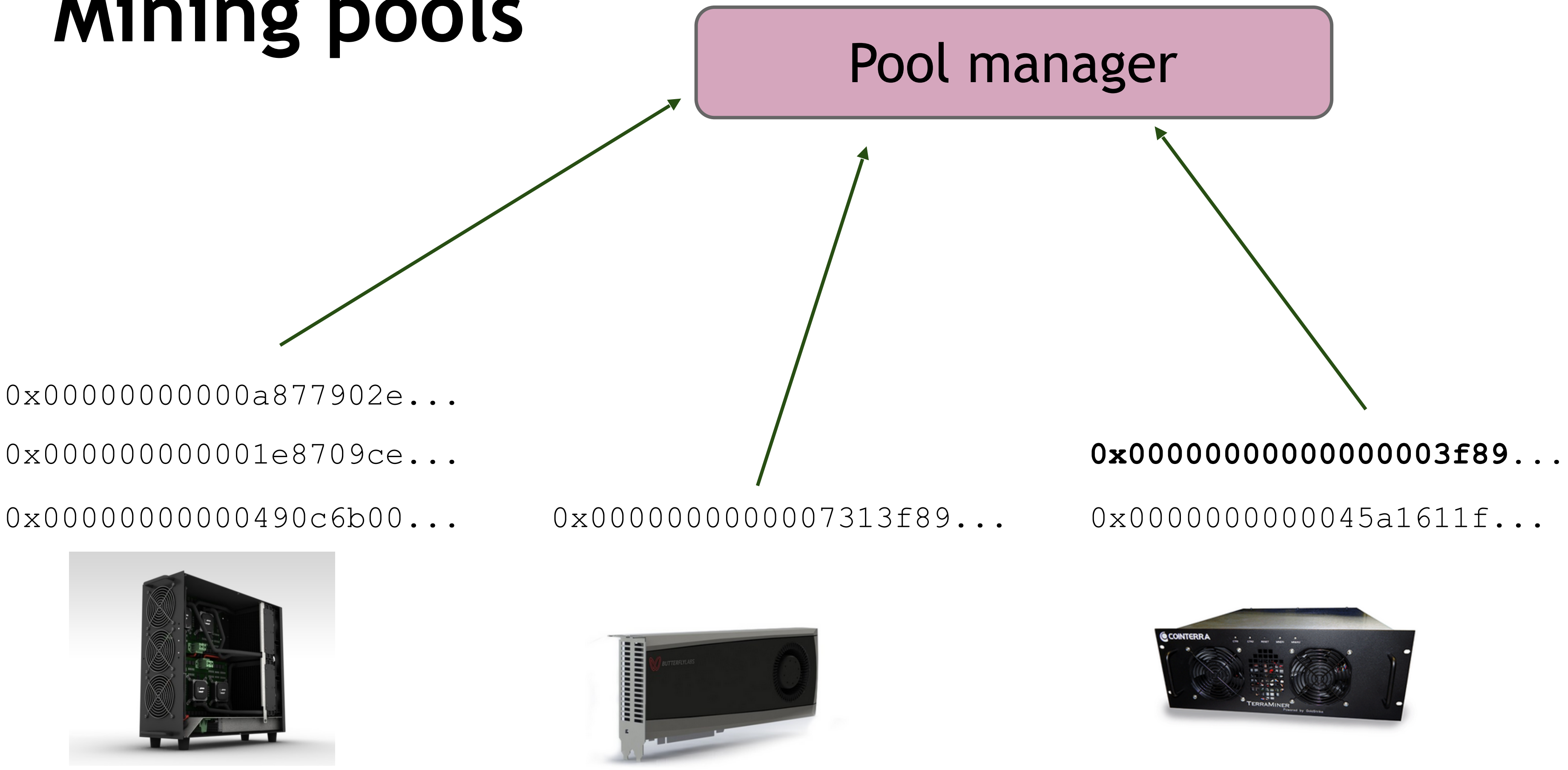
0x00000000000490c6b00...

0x0000000000007313f89...

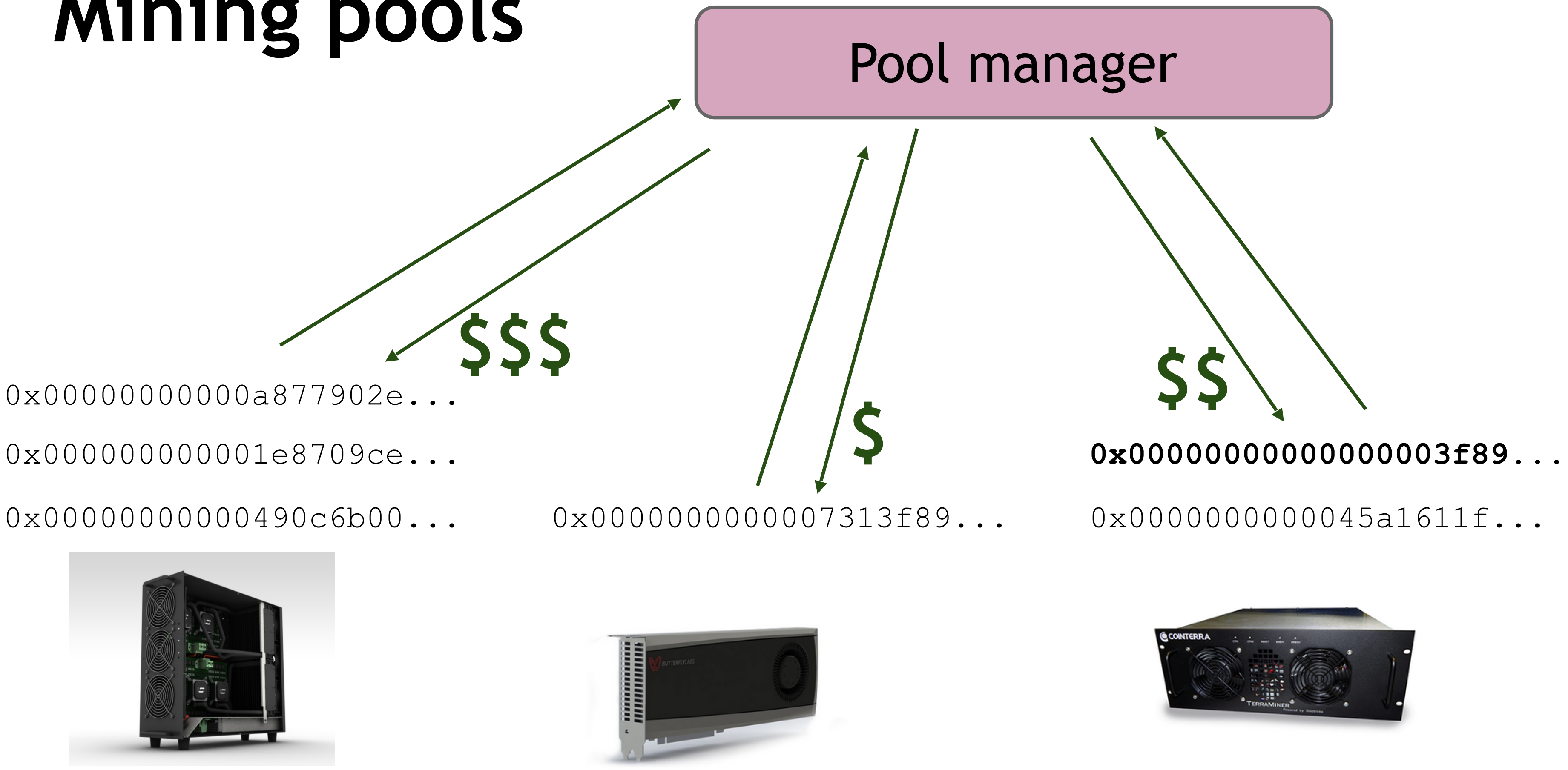
0x00000000000000003f89...

0x000000000045a1611f...

Mining pools



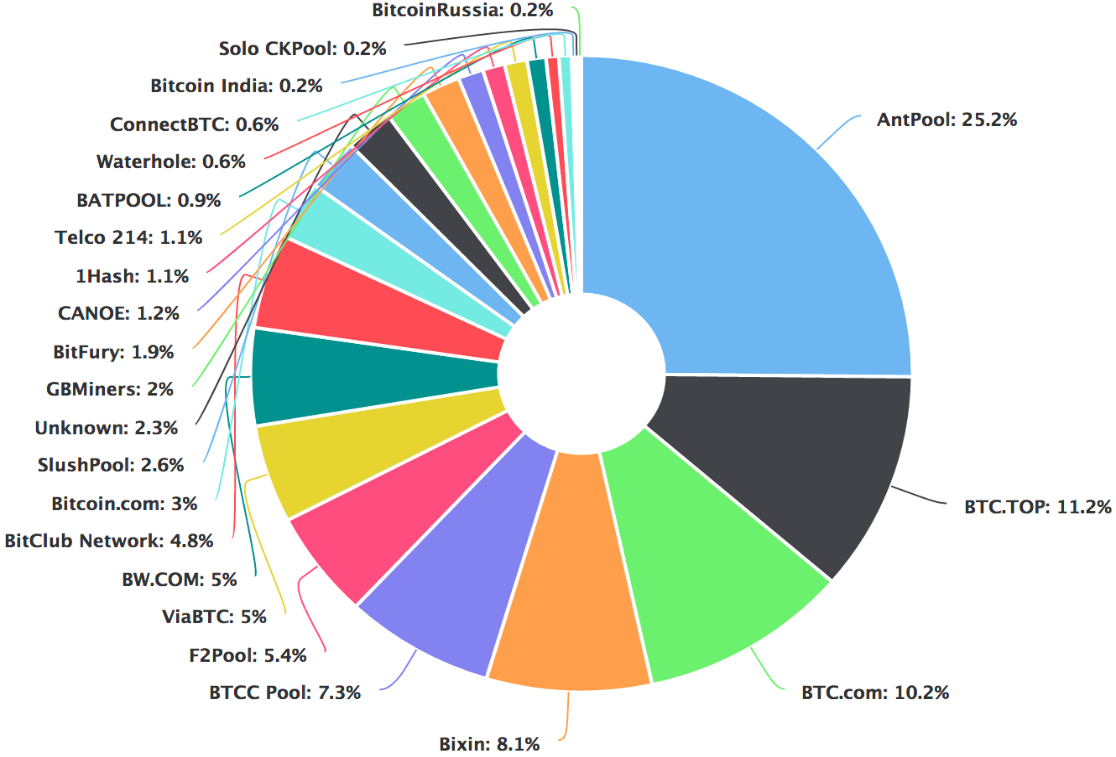
Mining pools



Mining pool history

- First pools appear in late-2010
  - Back in the GPU era!
- By 2014: around 90% of mining pool-based
- June 2014: GHash.io exceeds 50%

Mining pools



Are mining pools a good thing?

- Pros
  - Make mining more predictable
  - Allow small miners to participate
  - More miners using updated validation software
- Cons
  - Lead to centralization
  - Discourage miners from running full nodes

Question: Can we prevent pools?

Mining incentives and strategies

Game-theoretic analysis of mining

Several strategic decisions

- Which transactions to include in a block
  - Default: any above minimum transaction fee
- Which block to mine on top of
  - Default: longest valid chain
- How to choose between colliding blocks
  - Default: first block heard
- When to announce new blocks
  - Default: immediately after finding them

Game-theoretic analysis of mining

Assume you control $0 < \alpha < 1$ of mining power

Can you profit from a non-default strategy?

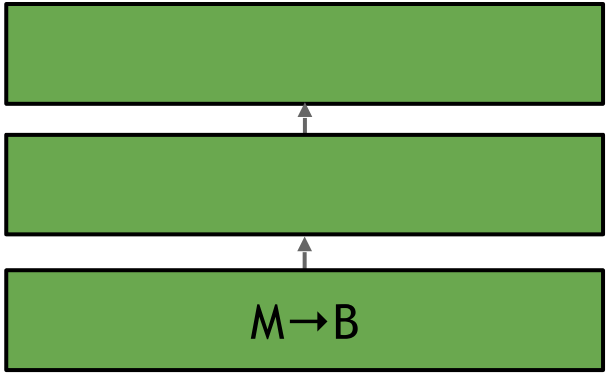
Game-theoretic analysis of mining

Assume you control $0 < \alpha < 1$ of mining power

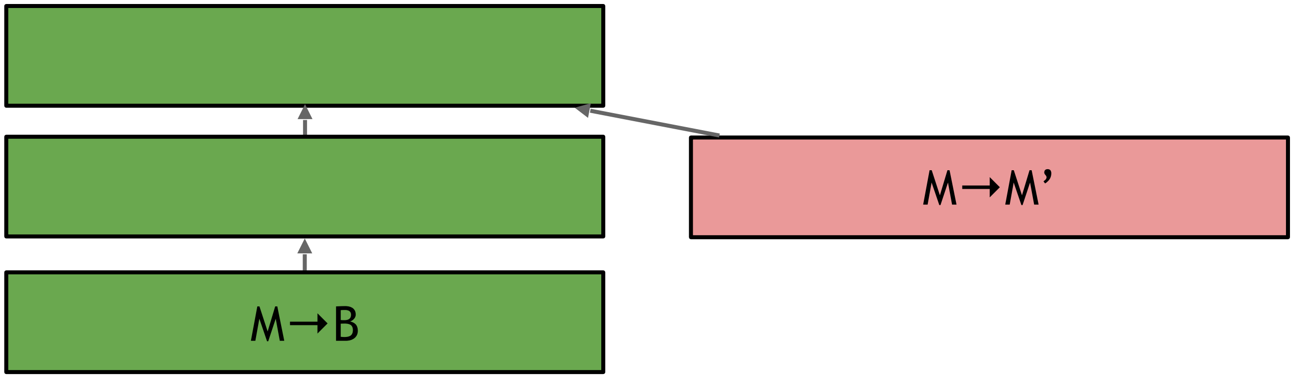
Can you profit from a non-default strategy?

For some α, **YES!**

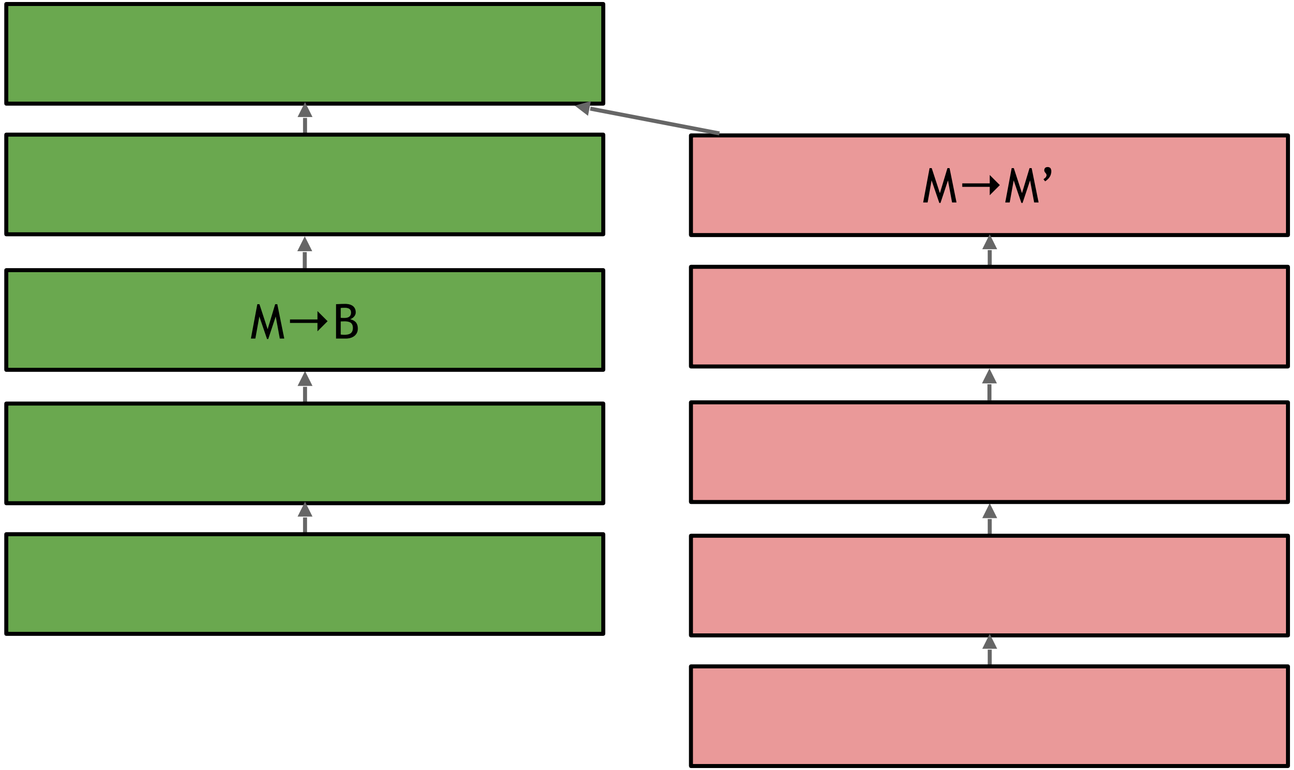
Forking attacks



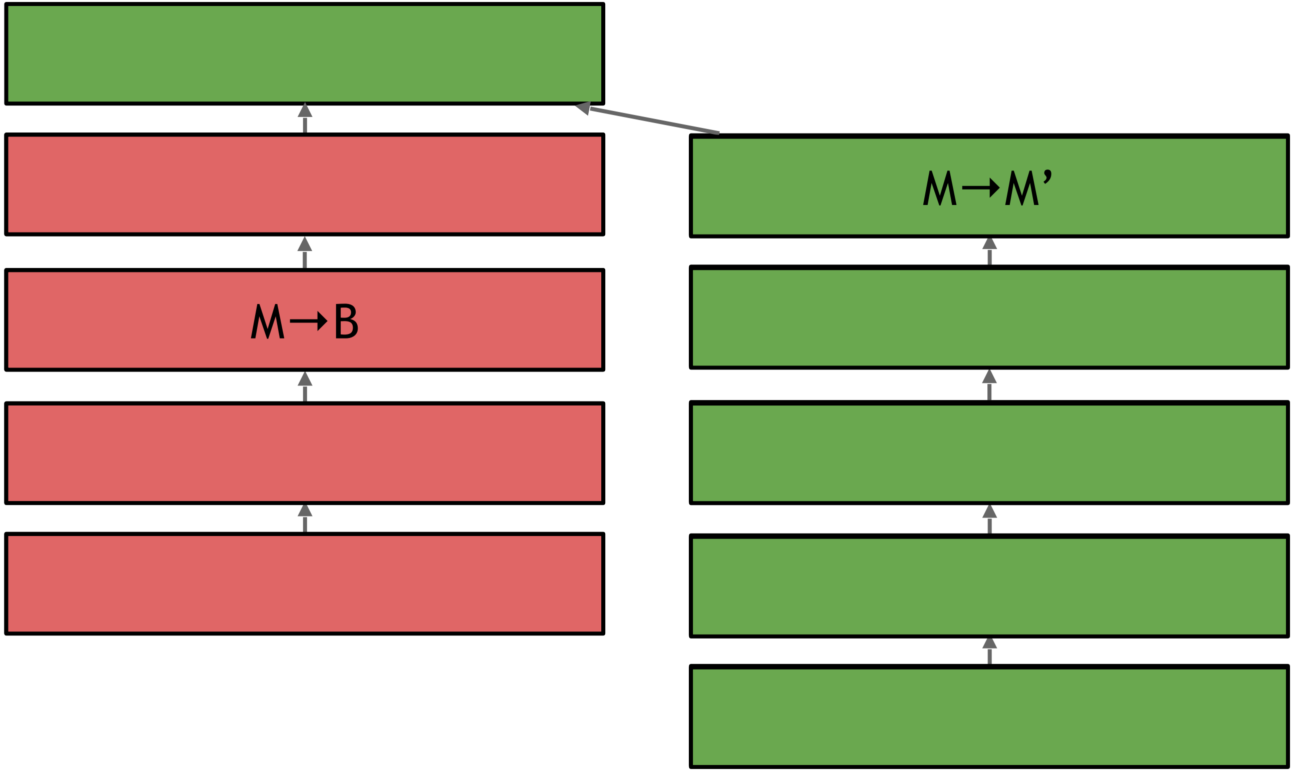
Forking attacks



Forking attacks



Forking attacks



Forking attacks

- Certainly possible if $\alpha > 0.5$
  - may be possible with less

Forking attacks

- Certainly possible if $\alpha > 0.5$
  - may be possible with less
- Attack is detectable

Forking attacks

- Certainly possible if $\alpha > 0.5$
    - may be possible with less
- Attack is detectable
- Might be reversed

Forking attacks

- Certainly possible if $\alpha > 0.5$
  - may be possible with less
- Attack is detectable
- Might be reversed
- Might crash exchange rate

Forking attacks

- Certainly possible if $\alpha > 0.5$
  - may be possible with less
- Attack is detectable
- Might be reversed
- Might crash exchange rate

Goldfinger Attack

Forking attacks

- Certainly possible if α >0.5
    - may be possible with less
- Attack is detectable
- Might be reversed
- Might crash exchange rate

Goldfinger Attack

Forking attacks via bribery

- **Idea:** building $\alpha > 0.5$ is expensive. Why not rent it instead?

- Payment techniques:
  - Out-of-band bribery
  - Run a mining pool at a loss
  - Insert large “tips” in the block chain

Selfish Mining

(a.k.a. Block-withholding attack)

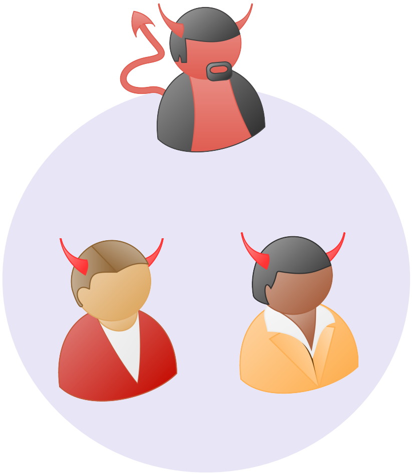
Majority is not Enough: Bitcoin Mining is Vulnerable*

Ittay Eyal and Emin Gün Sirer

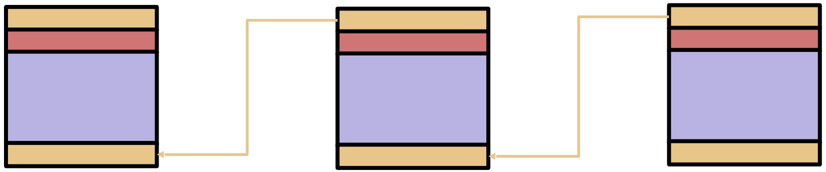
Department of Computer Science, Cornell University
ittay.eyal@cornell.edu, egs@systems.cs.cornell.edu

Selfish Mining: Strategy

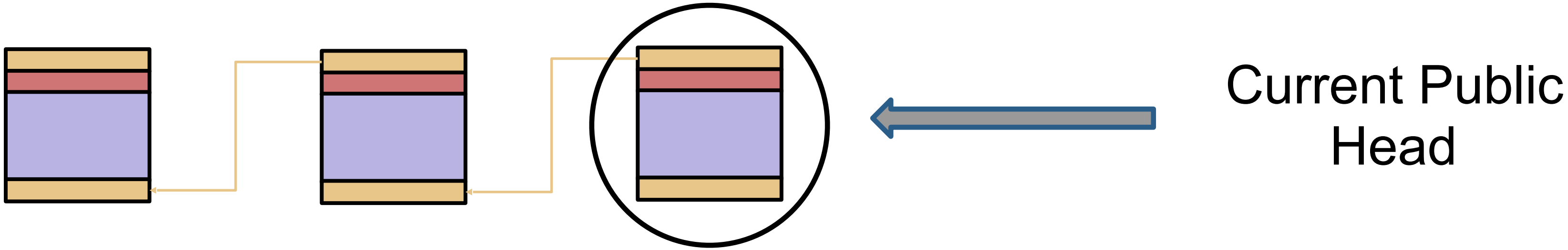
- Form a pool.
- Secretly fork the chain.
- Don't announce blocks right away. Try to get ahead!



Public Chain

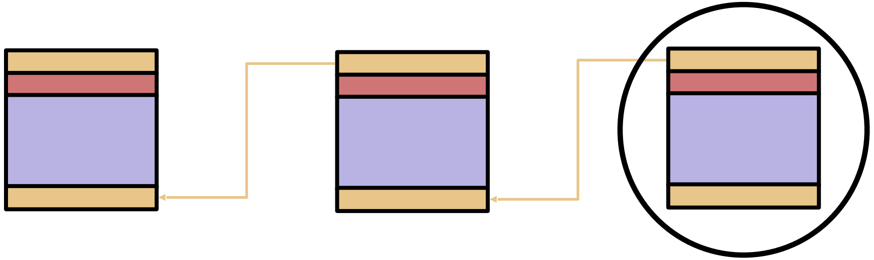


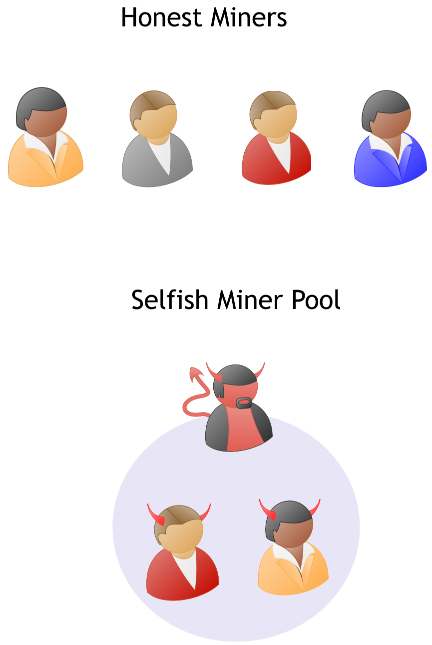
Public Chain

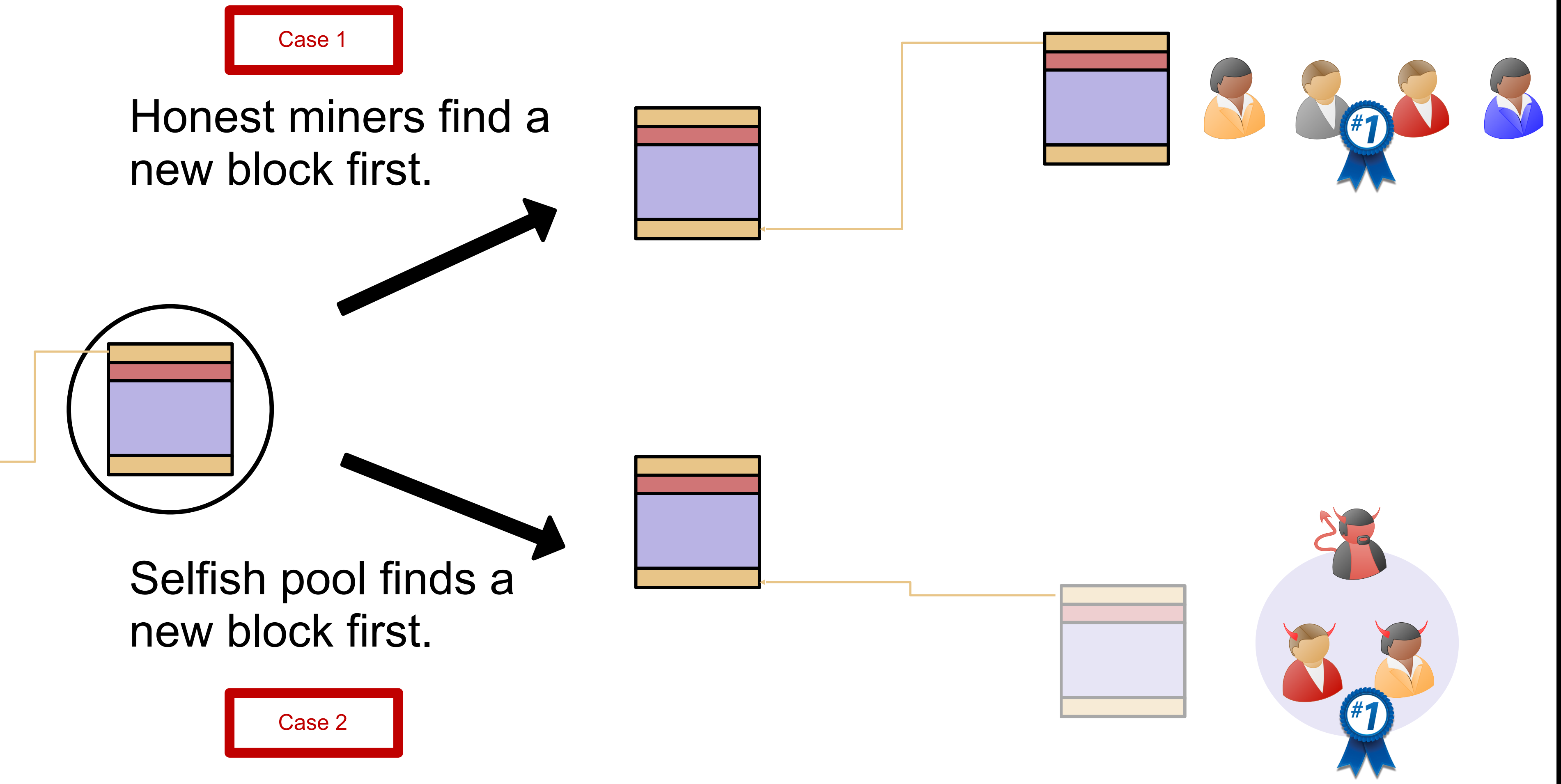


Public Chain

- The honest miners and the selfish miner pool start mining at the current public head.

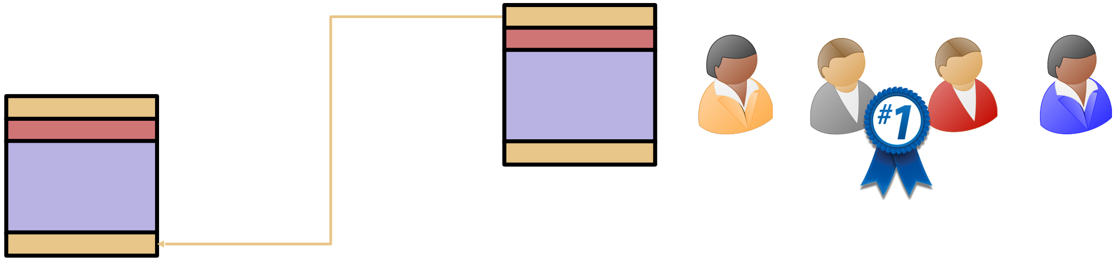


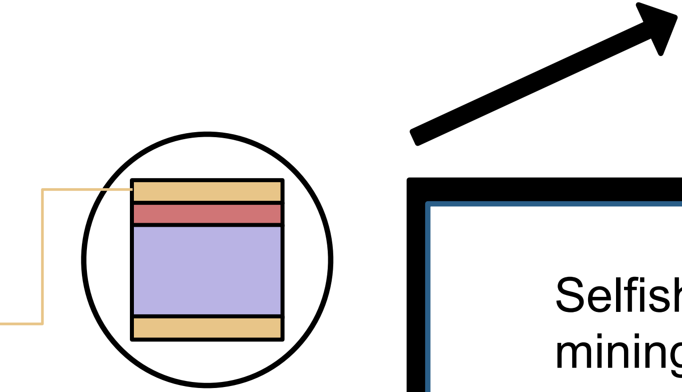



Case 1
Honest miners find a new block first.
Selfish pool finds a new block first.
Case 2
#1
#1

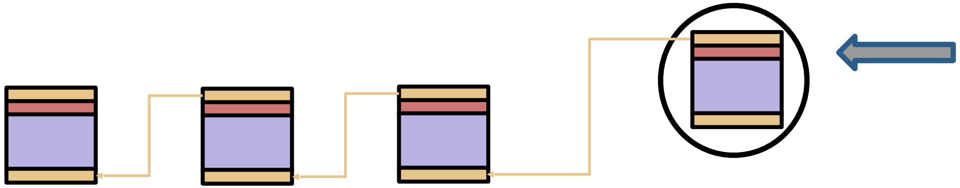
Case 1

Honest miners find a new block first.



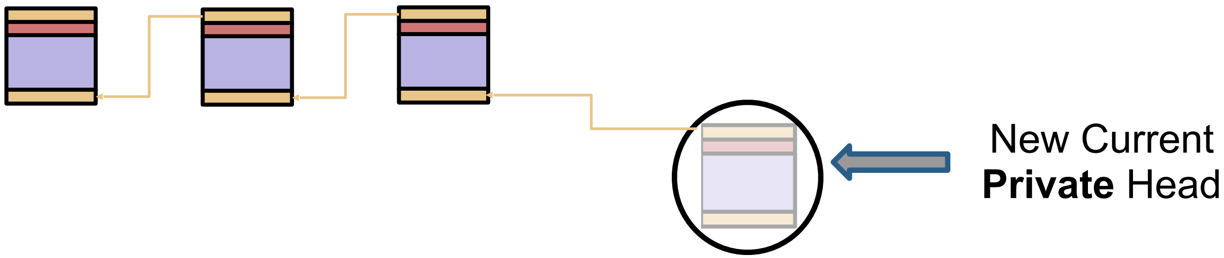


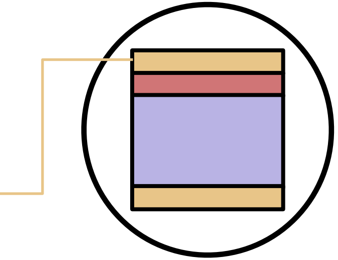
Selfish miner pool adopts the main branch and starts mining on the new current public head.

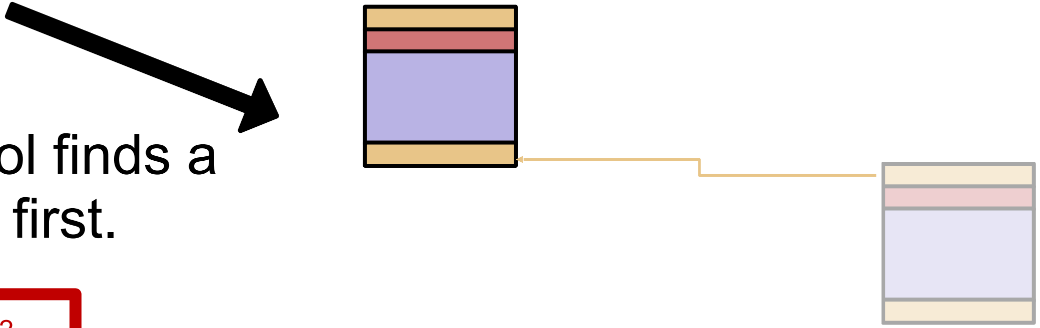


New Current Public Head

Selfish pool keeps this branch **private**, and starts mining on this private branch.

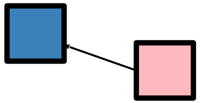






Selfish pool finds a new block first.

Case 2



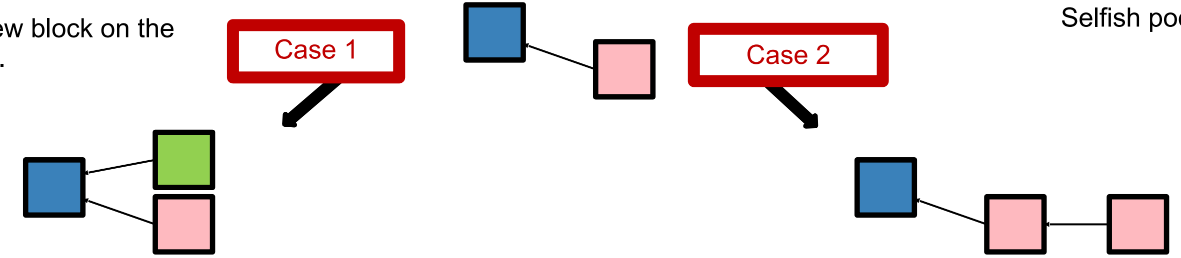
Public Head

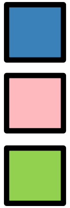
Block mined by selfish pool and kept private

Let’s focus on the case where selfish pool mines a block and keeps it private

Honest miners discover a new block on the public branch.

Selfish pool finds a second block.

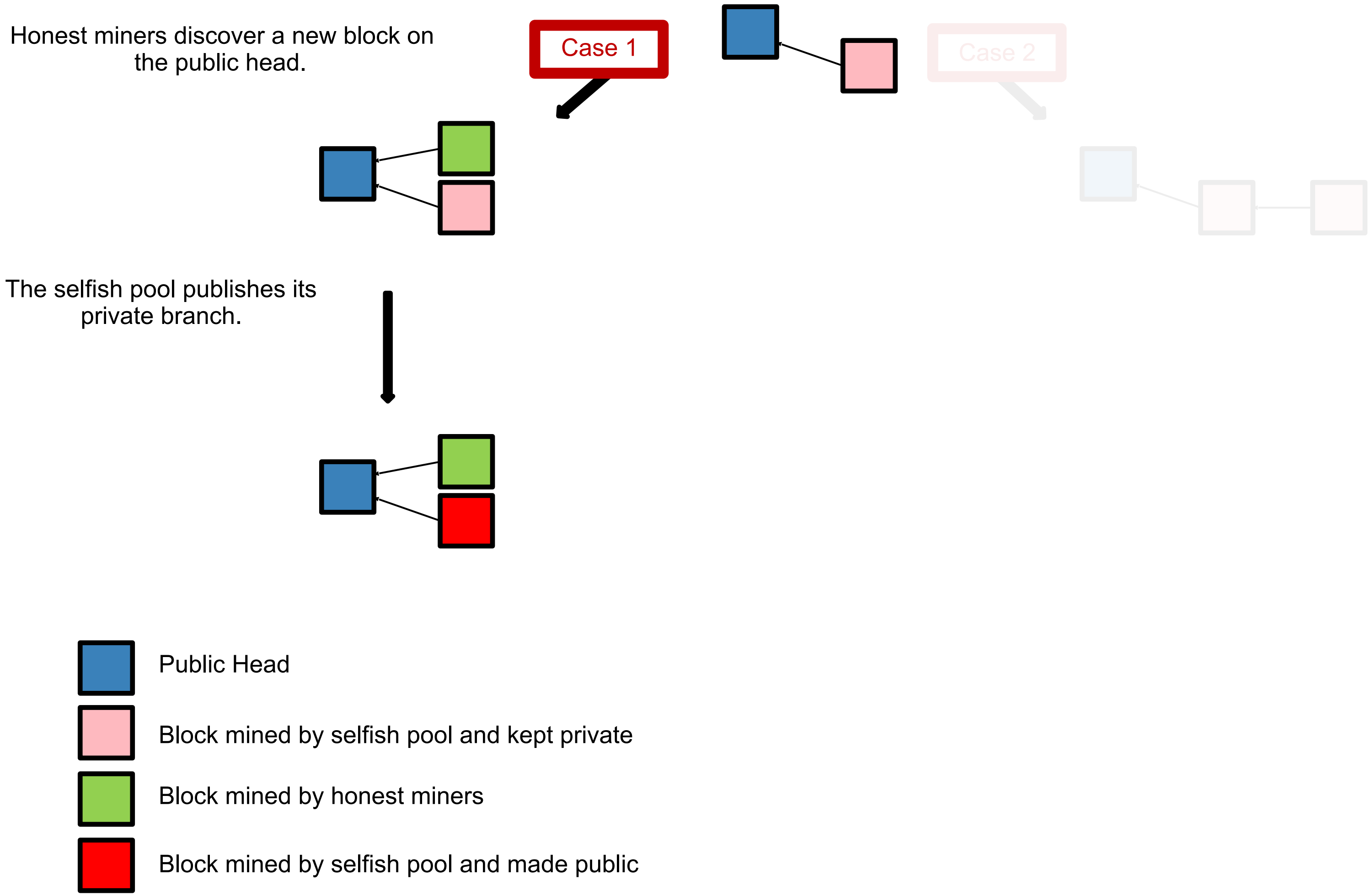




Public Head

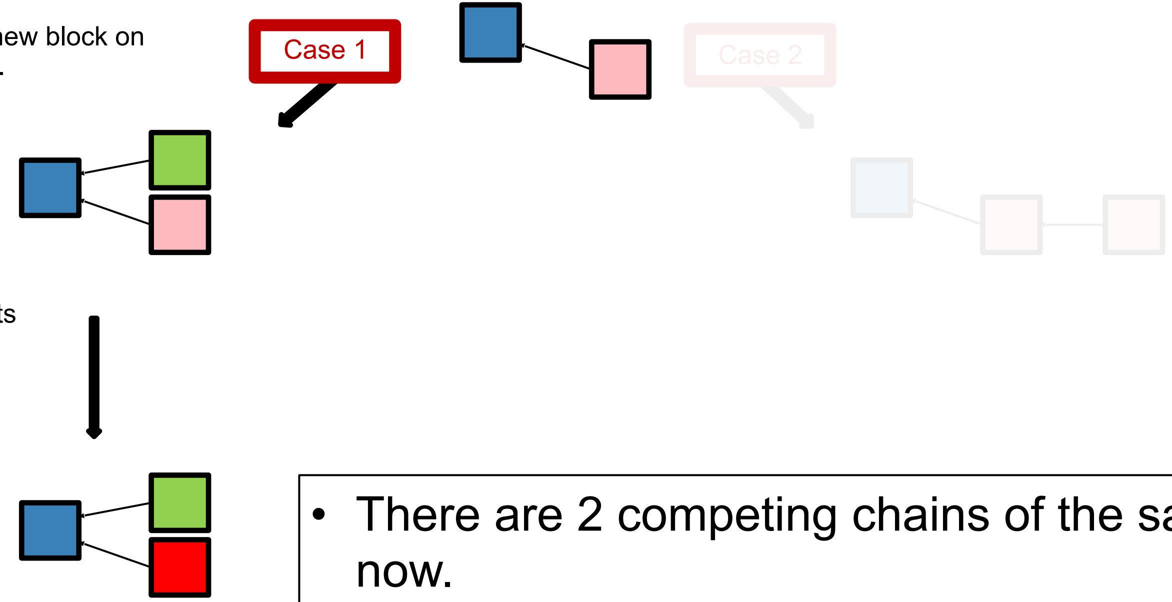
Block mined by selfish pool and kept private

Block mined by honest miners

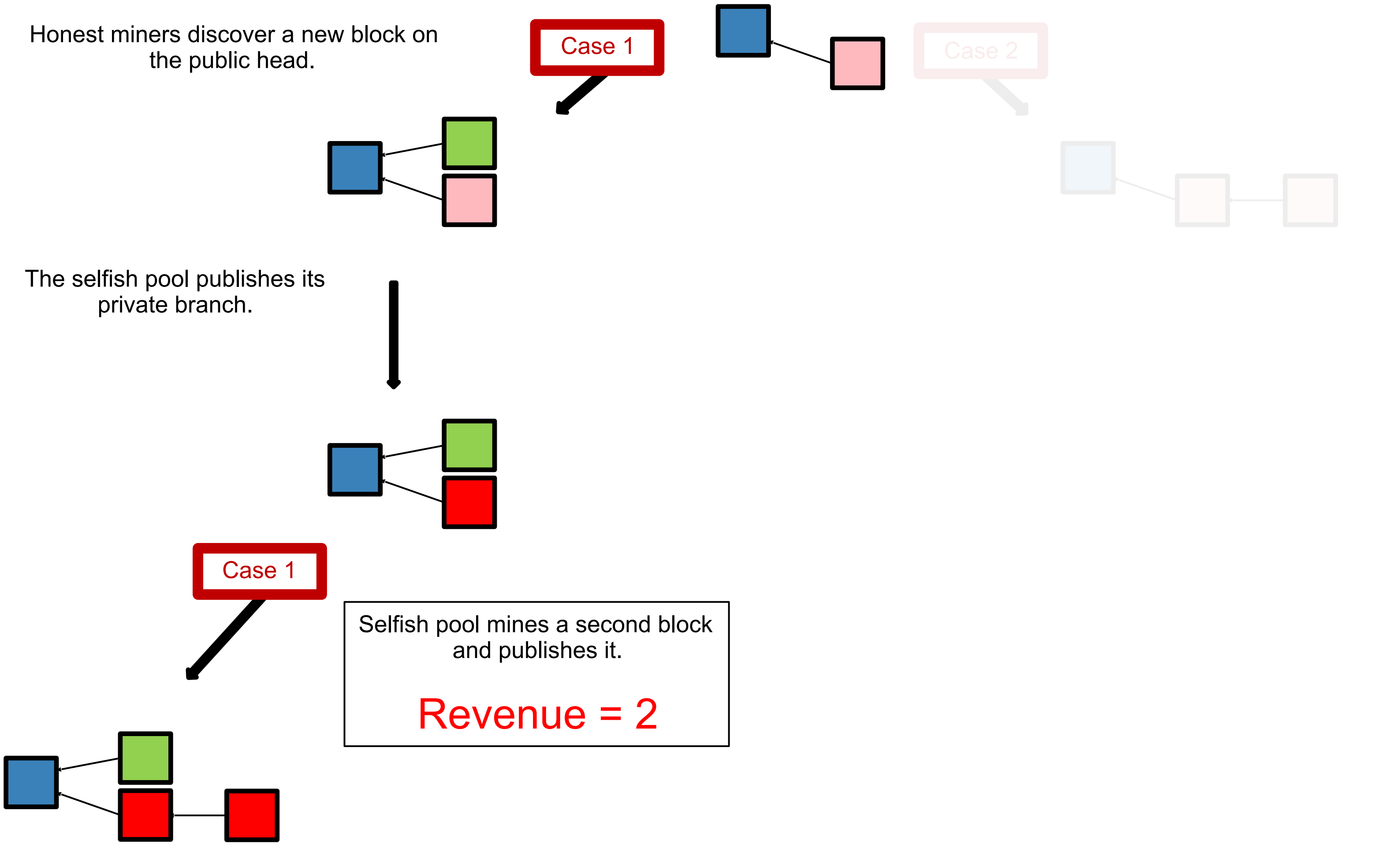

Honest miners discover a new block on the public head.
Case 1
Case 2
The selfish pool publishes its private branch.
Public Head
Block mined by selfish pool and kept private
Block mined by honest miners
Block mined by selfish pool and made public

Honest miners discover a new block on the public head.

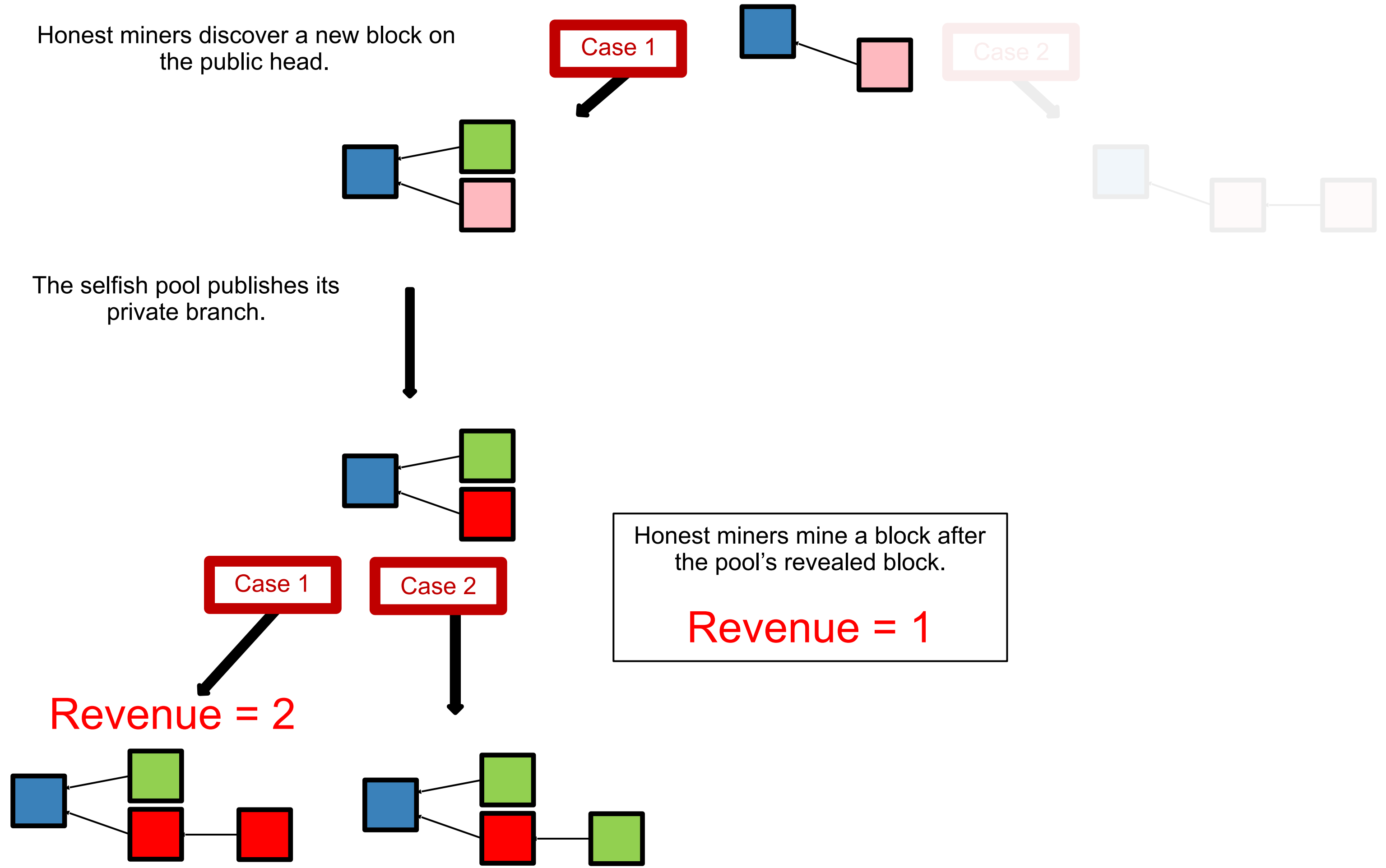
The selfish pool publishes its private branch.

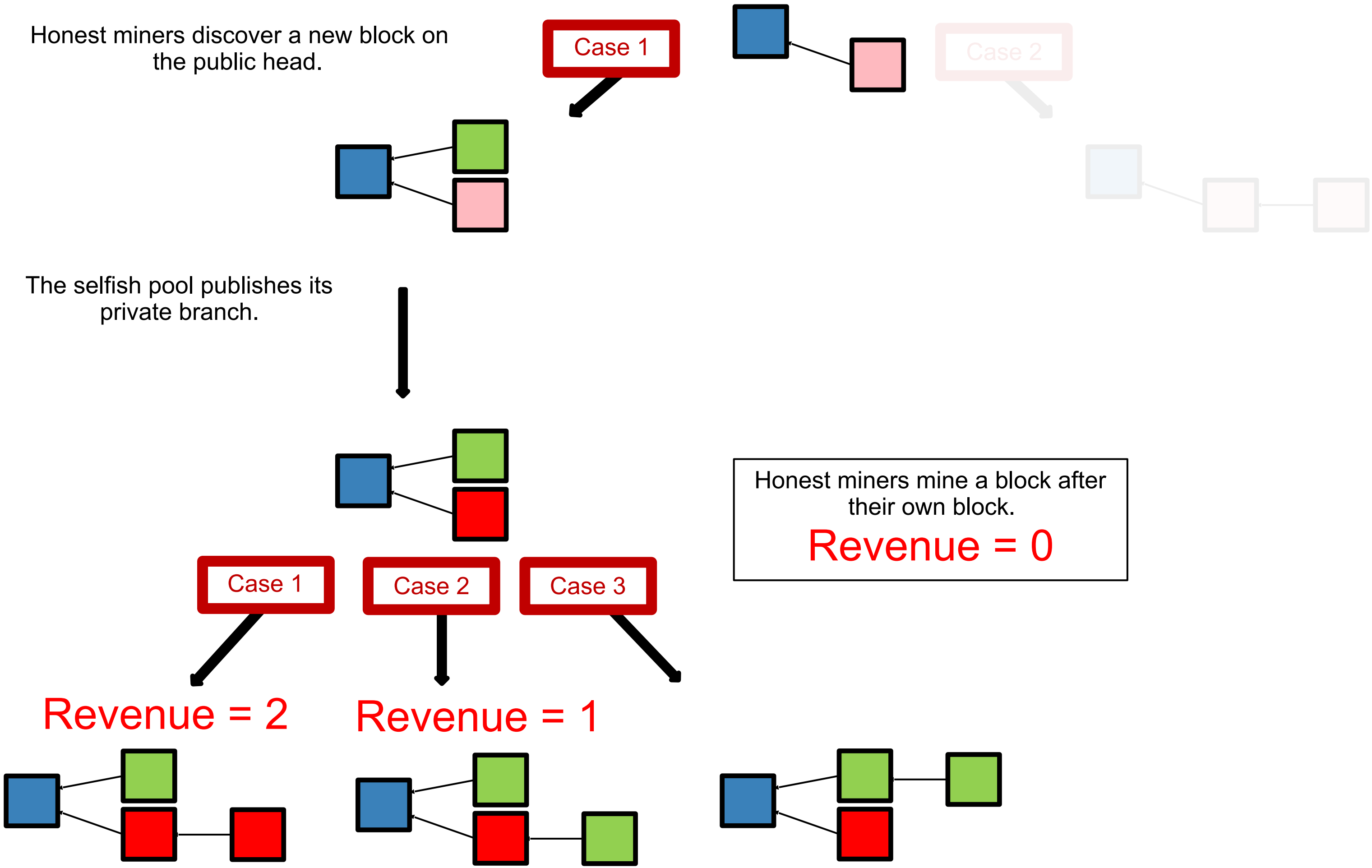


- There are 2 competing chains of the same length now.
- The selfish pool mines to extend its branch.
- Honest miners choose to mine on either branch.

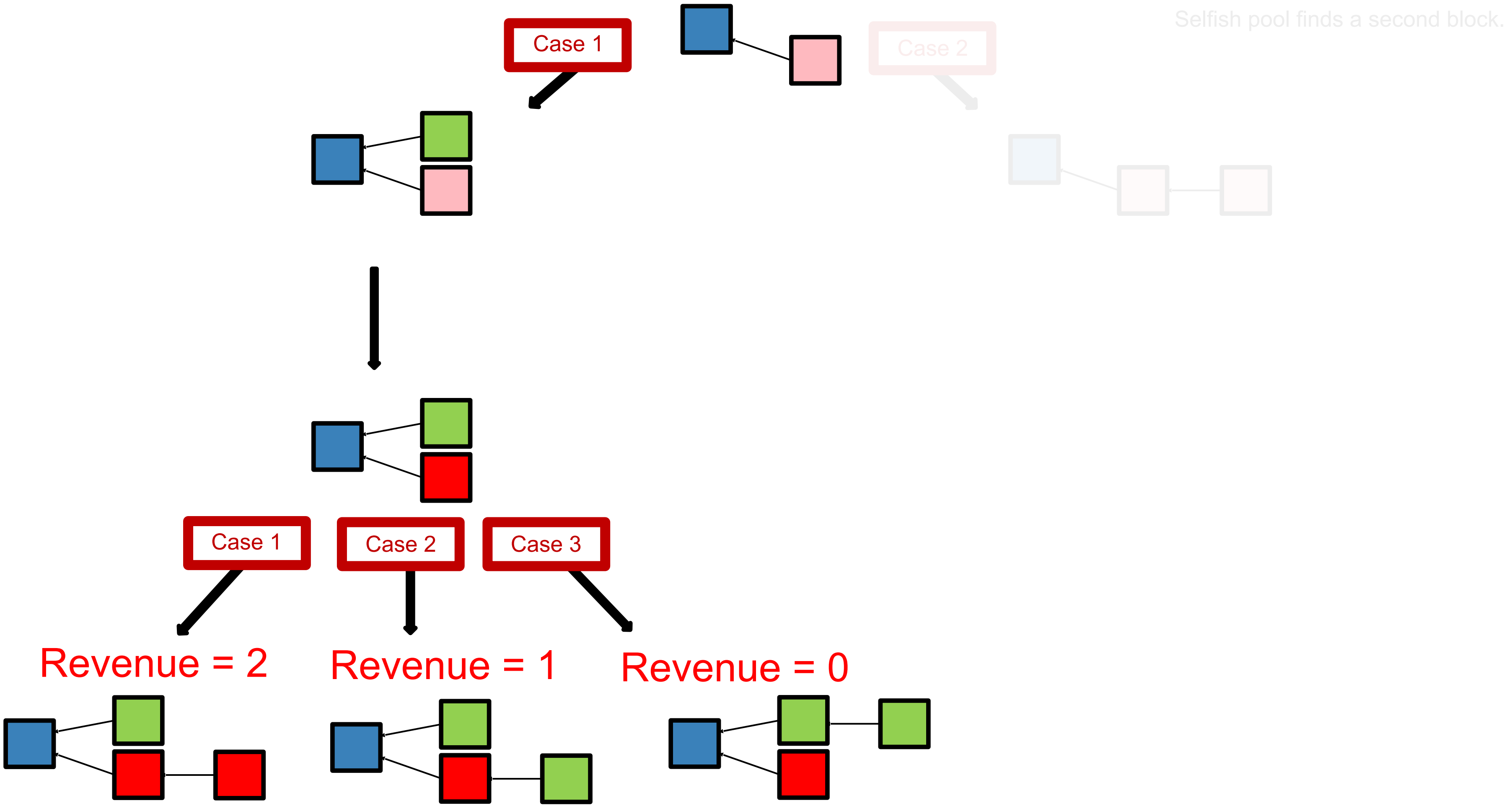


Honest miners discover a new block on the public head.
Case 1
Case 2
The selfish pool publishes its private branch.
Case 1
Selfish pool mines a second block and publishes it.
Revenue = 2


Honest miners discover a new block on the public head.
Case 1
Case 2
The selfish pool publishes its private branch.
Case 1
Case 2
Revenue = 2
Honest miners mine a block after the pool's revealed block.
Revenue = 1

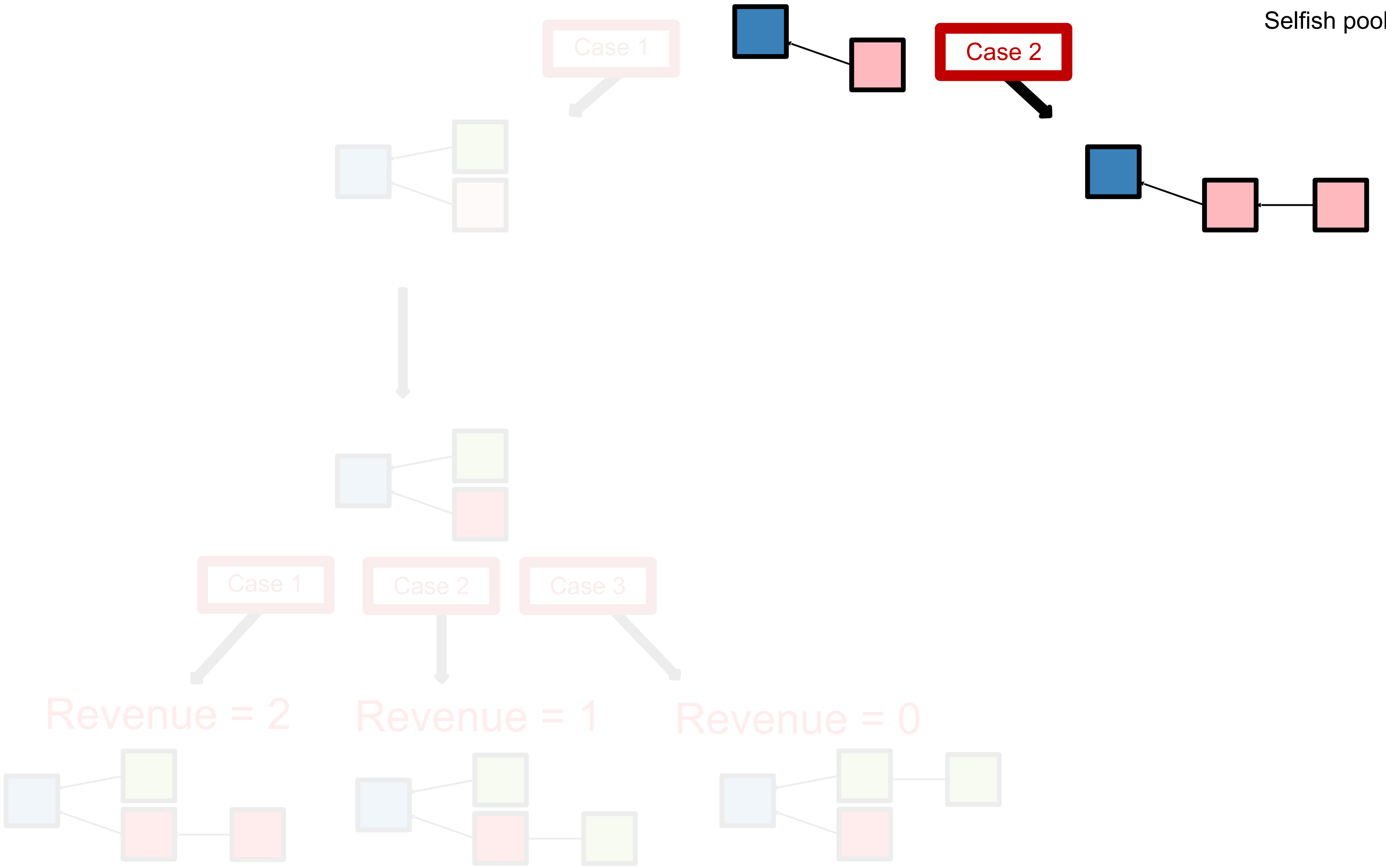


Honest miners discover a new block on the public head.
Case 1
Case 2
The selfish pool publishes its private branch.
Honest miners mine a block after their own block.
Revenue = 0
Case 1
Case 2
Case 3
Revenue = 2
Revenue = 1

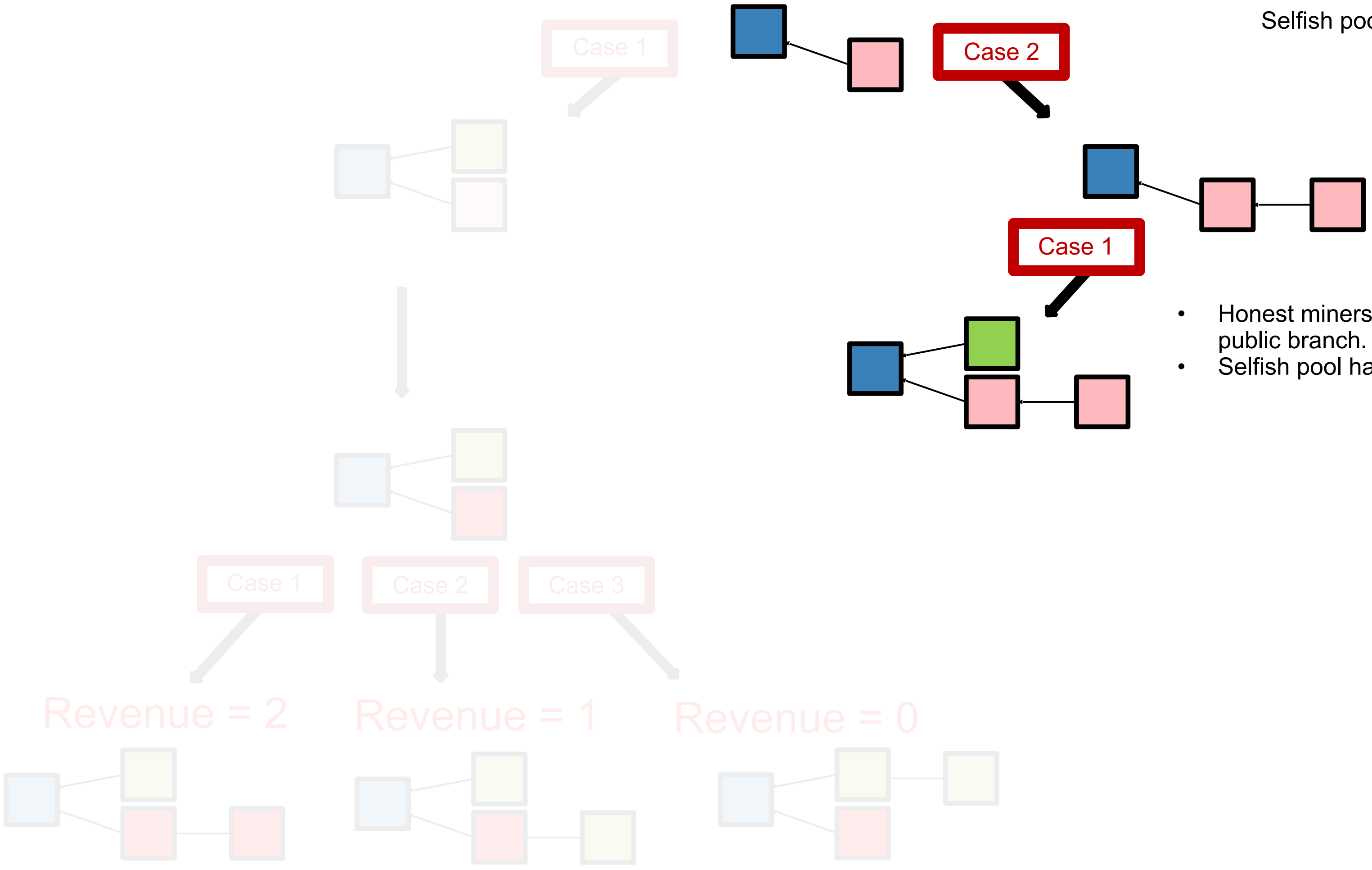


Selfish pool finds a second block.
Case 1
Case 2
Case 1
Case 2
Case 3
Revenue = 2
Revenue = 1
Revenue = 0

Selfish pool finds a second block.

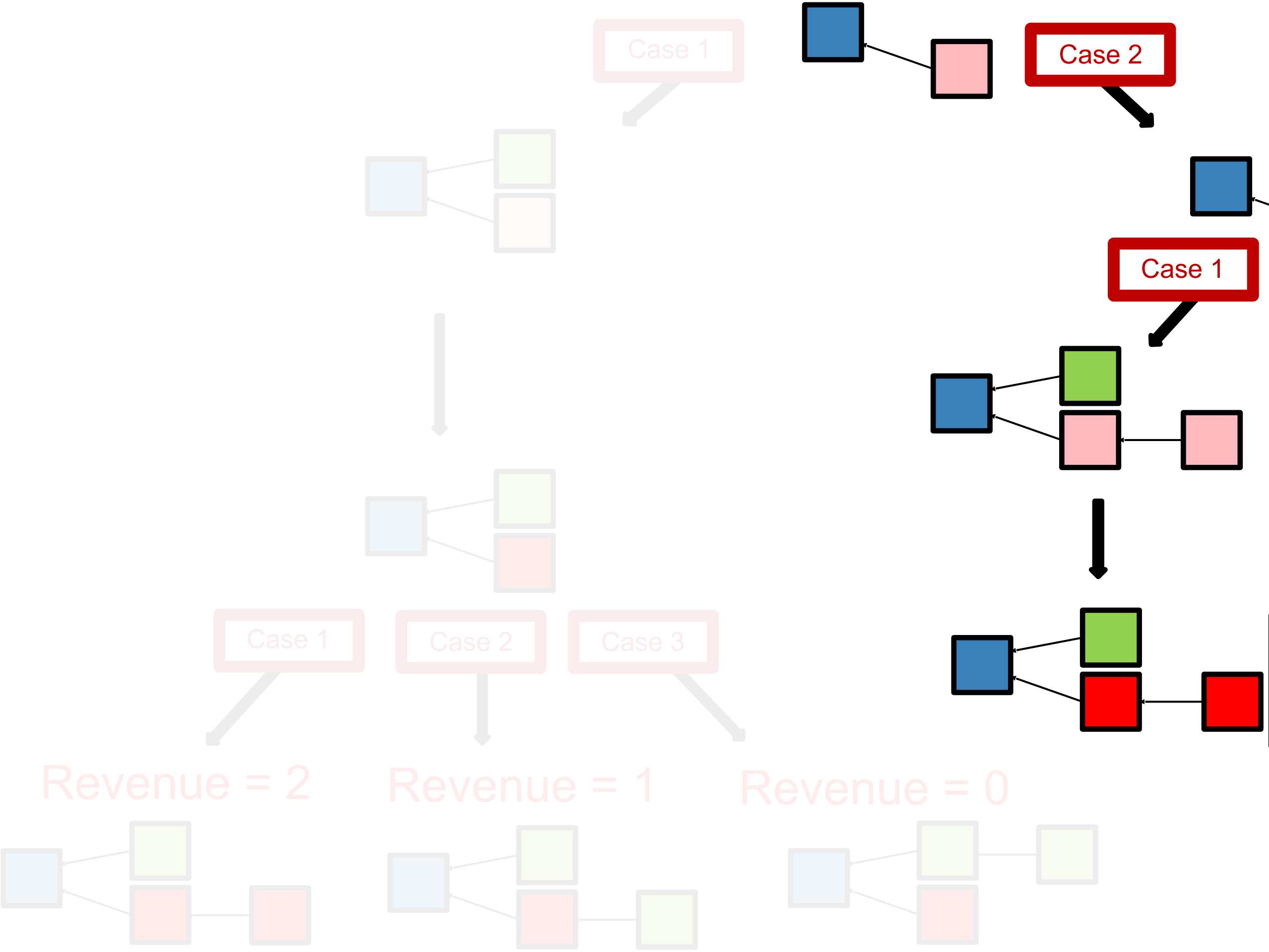


Selfish pool finds a second block.



- Honest miners mine a block on the public branch.
- Selfish pool has a lead of 1 block.

Selfish pool finds a second block.

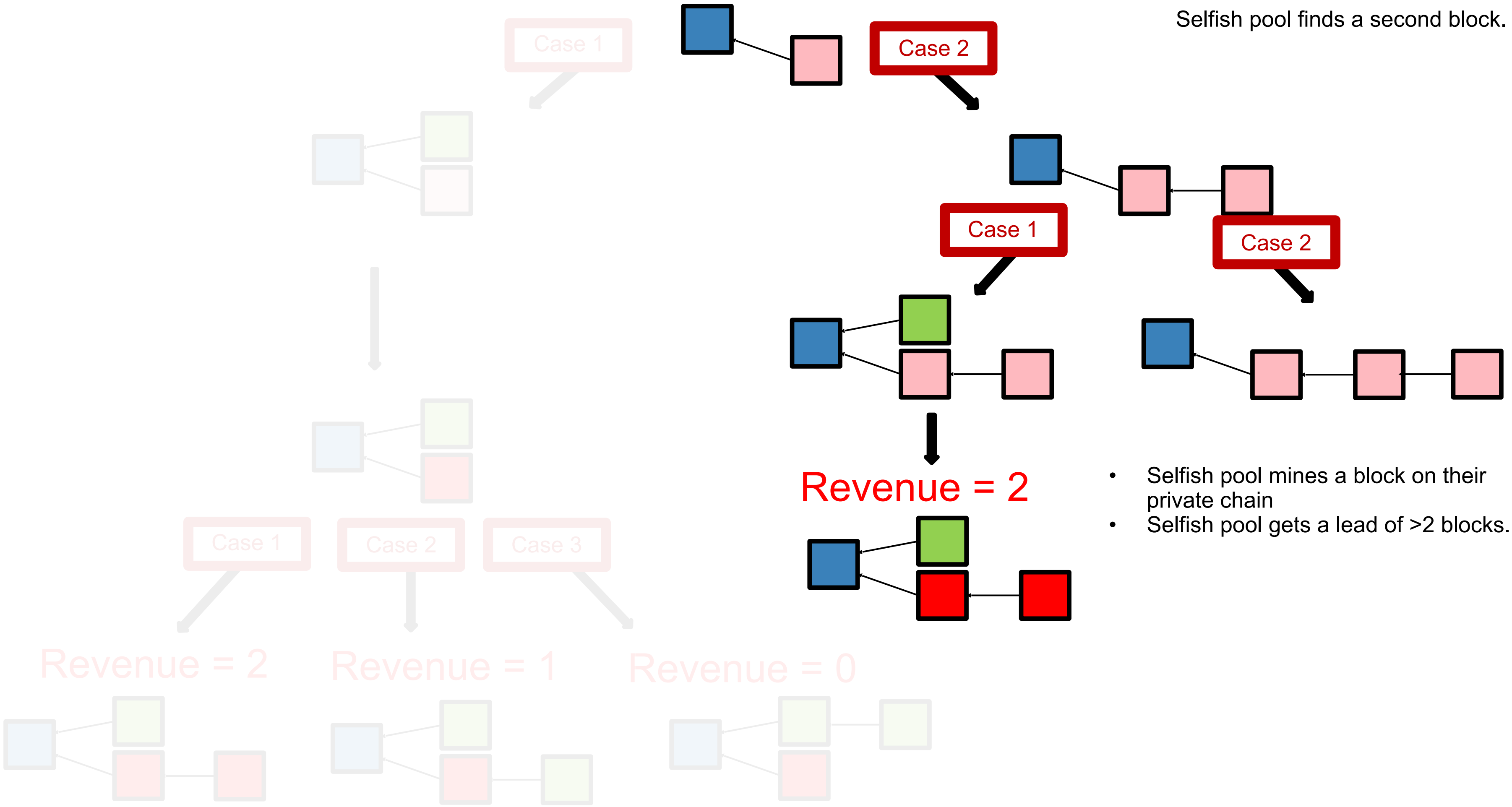


- Honest miners mine a block on the public branch.
- Selfish pool has a lead of 1 block.

Selfish Pool publishes the entire chain.

Revenue = 2

Selfish pool finds a second block.



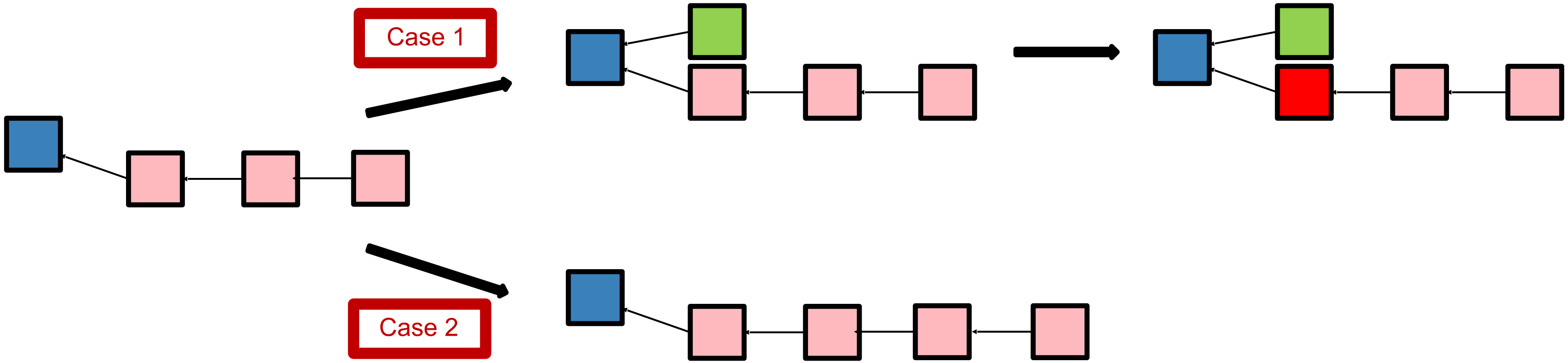
- Selfish pool mines a block on their private chain
- Selfish pool gets a lead of >2 blocks.

Selfish Pool gets a lead of >2 blocks

- Selfish pool continues to mine on its private branch.
- For each subsequent block mined by an honest party, it publishes one block from its private chain.
- Tries to maintain a lead of 2 blocks for as long as possible.
- If the lead reduces to 1, it publishes its private branch.

Earns revenue for all its blocks.

Selfish Pool gets a lead of >2 blocks



If the selfish pool is in minority, then with a very high probability this lead will eventually reduce to one block.

Analysis

- Set of miners in the system : $1, \ldots, n$
- Miner i has mining power: m_i

$$\sum_{i=1}^{n} m_i = 1$$

- Let the total mining power of selfish pool be: α
- Mining power of others: $(1 - \alpha)$
- Ratio of honest miners that choose to mine on pool's block: γ
- Ratio of honest miners that choose to mine on the other block : $(1 - \gamma)$

Analysis: Revenue Rate (Ideal Case)

- Revenue rate of each agent is the revenue earned by it for each block mined in the system.
- Let revenue rate of selfish pool be: r_{pool}
- Let total revenue rate of others be: r_{others}
- Revenue rate should be proportional to the mining power.

$$r_{pool} \propto \alpha$$

- Ideally, $r_{pool} + r_{others} = 1$

Analysis: Revenue Rate (Selfish Mining)

- Since selfish mining causes intentional branching in the blockchain, several mined blocks are not included in the blockchain.
- Total block generation rate drops.
- As a result, $r_{pool} + r_{others} < 1$

Analysis: Revenue Rate Ratio

- Actual revenue rate of each agent is the revenue rate ratio.
- Revenue rate ratio of an agent is defined as the ratio of its blocks out of the total blocks added to the main chain

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

- Assuming honest majority,

$$0 \leq \alpha \leq \frac{1}{2}$$

- Selfish miners earn more revenue than their mining power if,

$$R_{pool} > \alpha$$

- For a given $\gamma,$ a selfish miners pool of size α earns more revenue than its relative size for,

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

$$\boxed{\frac{1-\gamma}{3-2\gamma} \leq \alpha \leq \frac{1}{2}}$$

$$\boxed{\frac{1-\gamma}{3-2\gamma} \leq \alpha \leq \frac{1}{2}}$$

- Honest miners always mine on the pool's branch

  For $\gamma = 1, \quad 0 \leq \alpha \leq \frac{1}{2}$

- Honest miners randomly choose which branch to mine on

  For $\gamma = \frac{1}{2}, \quad \frac{1}{4} \leq \alpha \leq \frac{1}{2}$

- Honest miners never mine on the pool's branch

  For $\gamma = 0, \quad \frac{1}{3} \leq \alpha \leq \frac{1}{2}$

Can we control γ ?

- In case of multiple branches of the same length:
  - A miner mines and propagates the first branch it received.
- There is no measure to guarantee a low γ.
- Sybil attack combined with selfish mining can lead to $\gamma \approx 1$.
  - In this case, a selfish pool of any size would earn more revenue than its mining power.
  - Rational miners will join the selfish pool.
  - The selfish pool would increase towards majority.

A modification for controlling γ

- In case a miner encounters multiple branches of the same length:
  - He should propagate all the branches it receives.
  - He should choose which one to mine on uniformly at random.
- This change would yield $\gamma = \frac{1}{2}$.
- This change is backward compatible.

Selfish-mining attacks

- Surprising departure from previous assumptions
- Not yet observed in practice!
- Plausible reason: selfish-mining is detectable, could lead to a crash in exchange rates for Bitcoin

Punitive forking

- Suppose you want to blacklist transactions from address *X*
  - Freeze an individual's money forever
- **Extreme strategy:** announce that you will refuse to mine on any chain with a transaction from *X*

Punitive forking

- Suppose you want to blacklist transactions from address *X*
  - Freeze an individual's money forever
- **Extreme strategy:** announce that you will refuse to mine on any chain with a transaction from *X*

With $\alpha < 0.5$, you'll soon fall behind the network

Feather-forking strategy

- **To blacklist transactions from X,** announce that you will refuse to mine *directly* on any block with a transaction from *X*
  - but you'll concede after *n* confirming blocks

- Chance of pruning an offending block is $\boldsymbol{\alpha^2}$

Response to feather forking

- **For other miners,** including a transaction from *X* induces an $\boldsymbol{\alpha^2}$ chance of losing a block
- Might be safer to join in on the blacklist
- Can enforce a blacklist with $\boldsymbol{\alpha < 0.5}$**!**

Response to feather forking

- **For other miners**, including a transaction from *X* induces an $\boldsymbol{\alpha^2}$ chance of losing a block
- Might be safer to join in on the blacklist
- Can enforce a blacklist with $\boldsymbol{\alpha < 0.5}$**!**

Success depends on convincing other miners you'll fork

Feather-forking: what is it good for?

- Freezing individual bitcoin owners
  - ransom/extortion
  - law enforcement?
- Enforcing a minimum transaction fee
  - Current transaction fees are low (about 2% of revenue)
  - But may become significant when mining reward becomes low

Summary

- Miners are free to implement any strategy
- Very little non-default behavior in the wild
- Game-theoretic analysis necessary
- Very recent work: [Badertscher-Garay-Maurer-Tshudi-Zikas, EUROCRYPT’18]