Blockchain 4

PoW and Forks

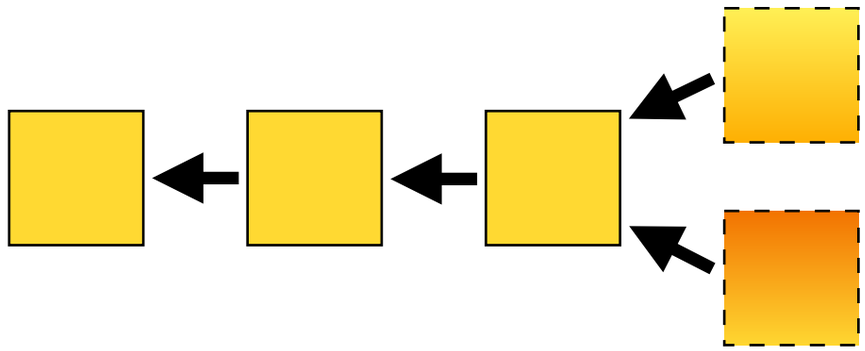
Leander Jehl

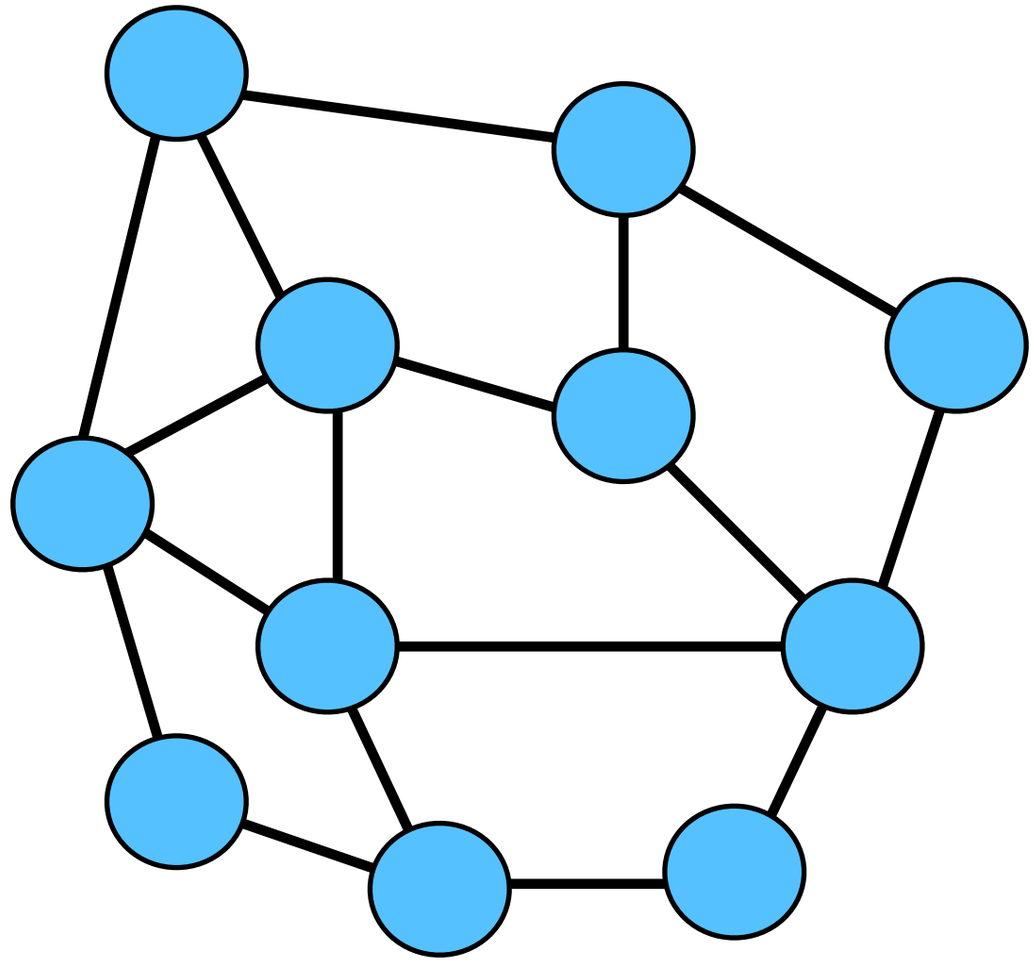
DAT650 Blockchain technology

Forks and longest chain rule

Forks

- A fork is if multiple blocks have the same predecessor





- Why: Two blocks found “concurrently”

Forks

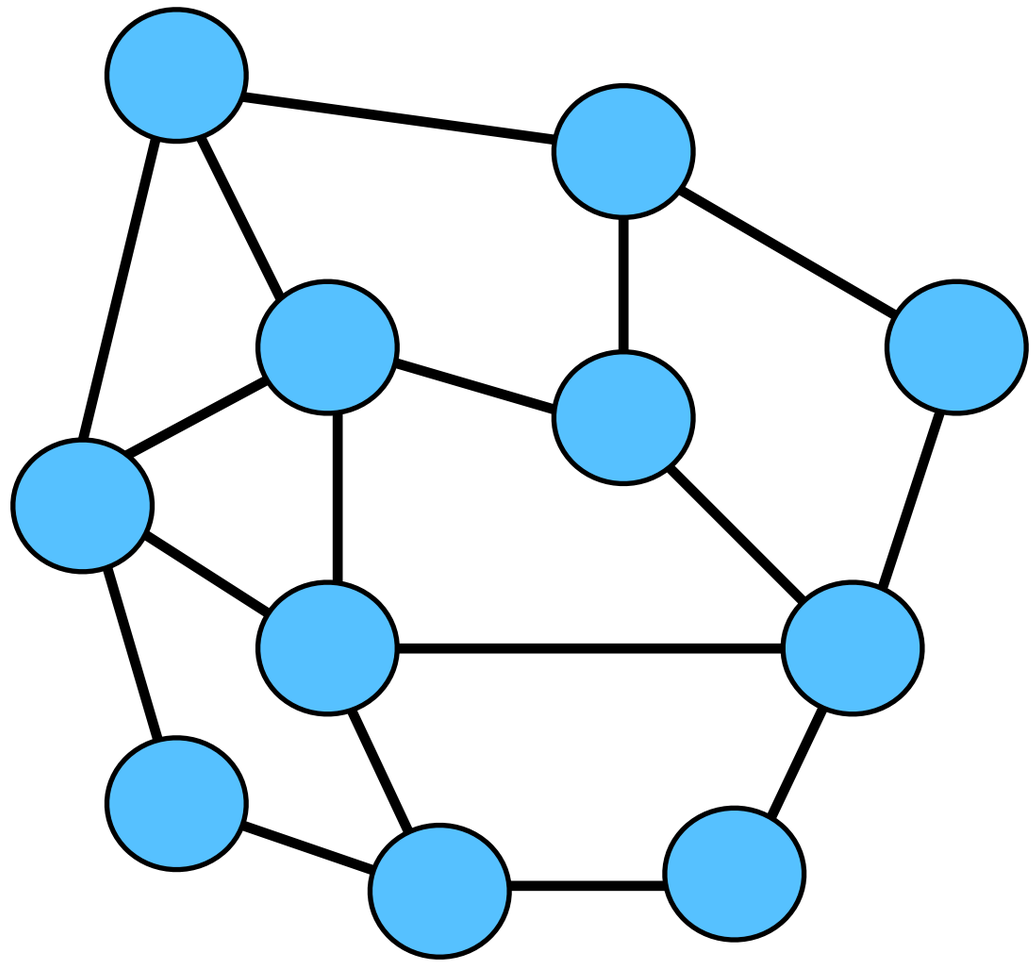
Proof of work workflow

Every node does:

- collect transaction to form block data
- try to solve PoW (*find nonce*)
- the first to solve PoW publishes block to everybody

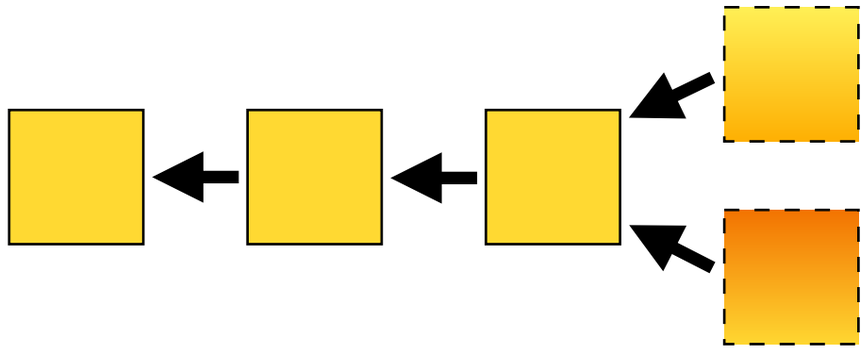
another block found before end of propagation

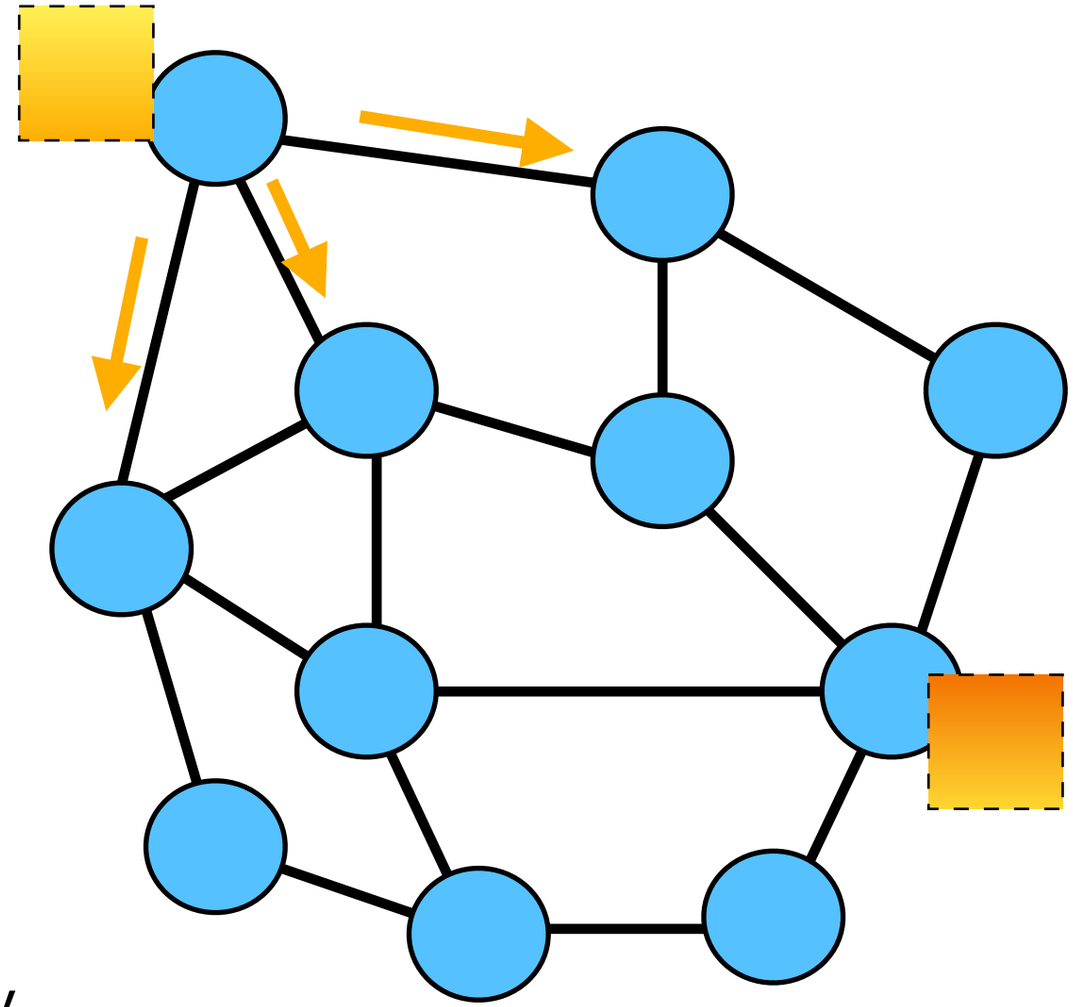
- all check PoW, *validate Block,* apply transactions, continue



Forks

- A fork is if multiple blocks have the same predecessor



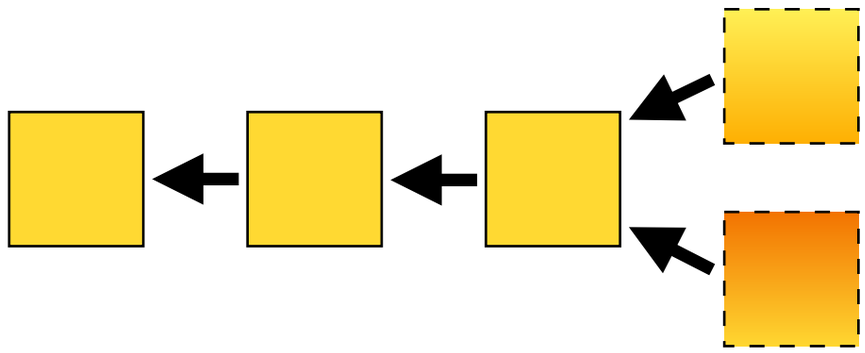


- Why: Two blocks found “concurrently”
- Bitcoin 2013: avg. 12.6sec block delivery [Decker, Wattenhofer]

Forks

Longest chain rule

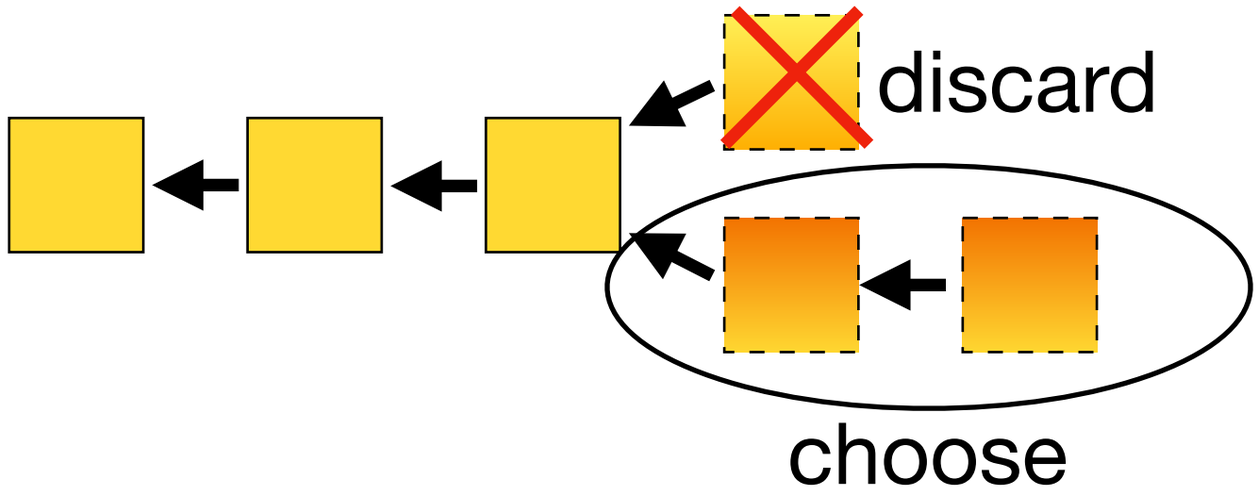
- If a fork exists, all nodes should adopt the longest chain.



Forks

Longest chain rule

- If a fork exists, all nodes should adopt the longest chain.



Forks

Longest chain rule

- If a fork exists, all nodes should adopt the longest chain.

Problems:

- Blocks & Transactions in smaller chain are discarded
  - Miners loose reward
  - Some transactions may be only in one fork
  - Two conflicting transactions may be included in different forks (double spend)

Forks

Math: How likely is a fork

p_{sec} probability a block is found in one second

δ average time to get a block from the network

Theorem:

$$P[\text{fork}] = 1 - (1 - p_{sec})^{\delta}$$

Forks

Reparametrization

Fork probability depends on

- Network delay
  time to propagate a block
- PoW difficulty

lower difficulty → more forks

lower difficulty → faster blocks → more throughput

Forks

Reparametrization

Fork probability depends on

- Network delay
  time to propagate a block
- PoW difficulty

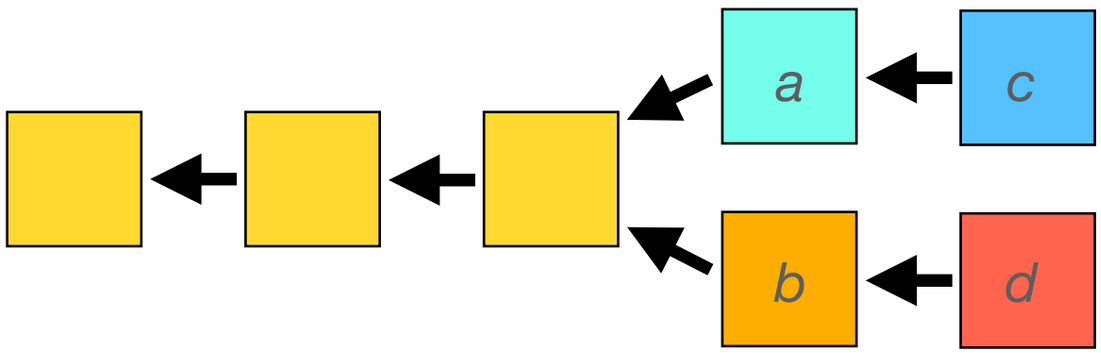
larger blocks → more throughput

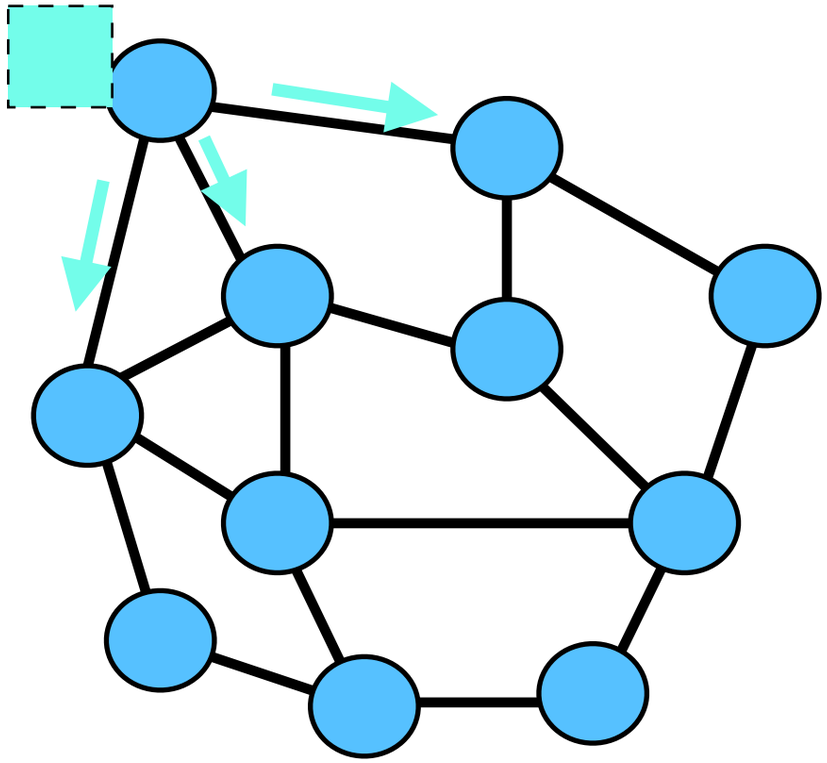
larger blocks → slow propagation → more forks

Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
  - *d* found while *c* was propagated.

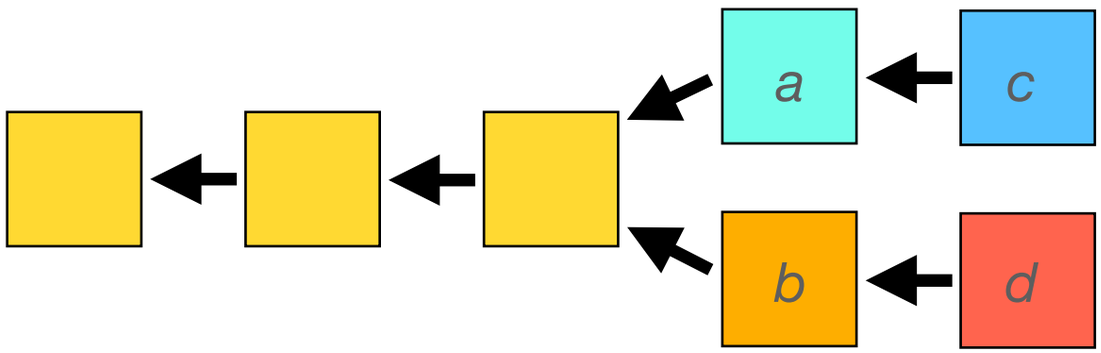


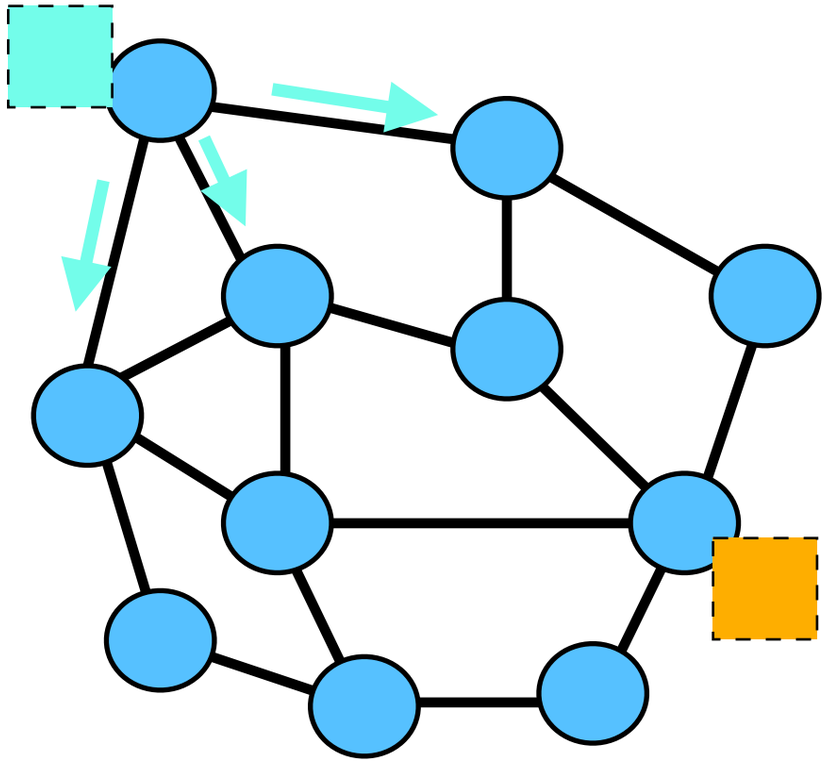


Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
  - *d* found while *c* was propagated.

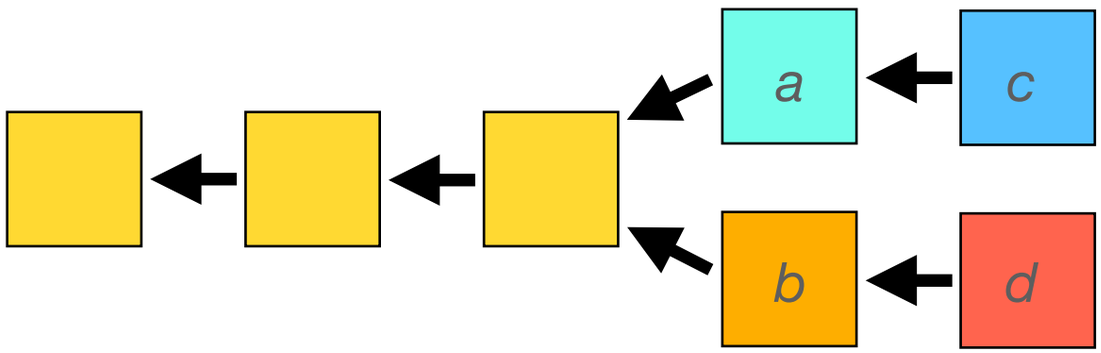


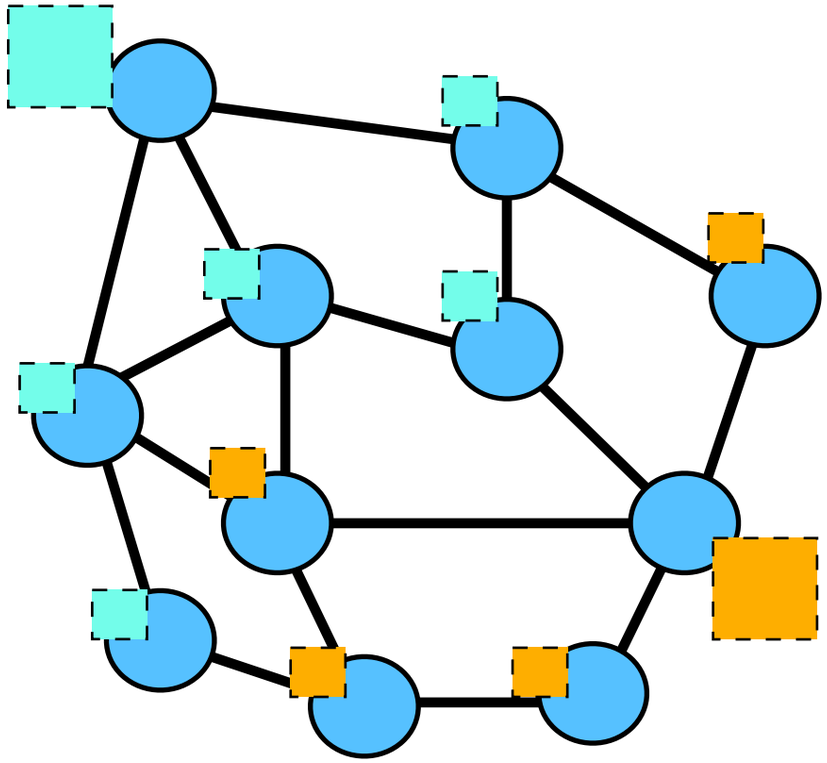


Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
    - *d* found while *c* was propagated.

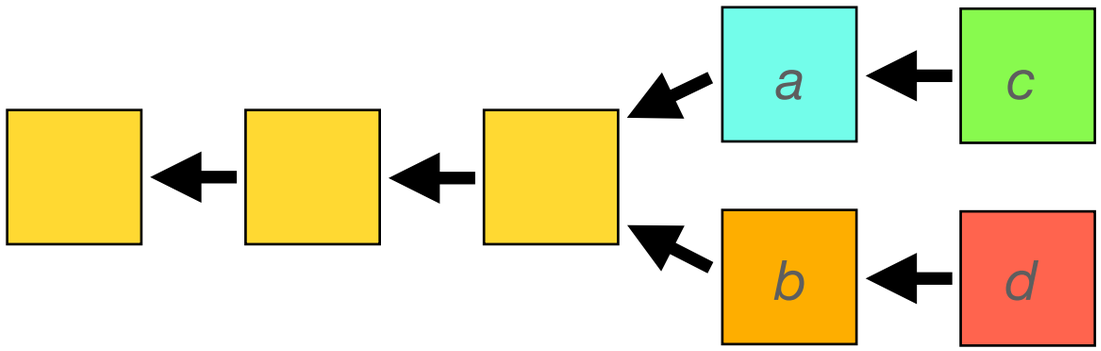


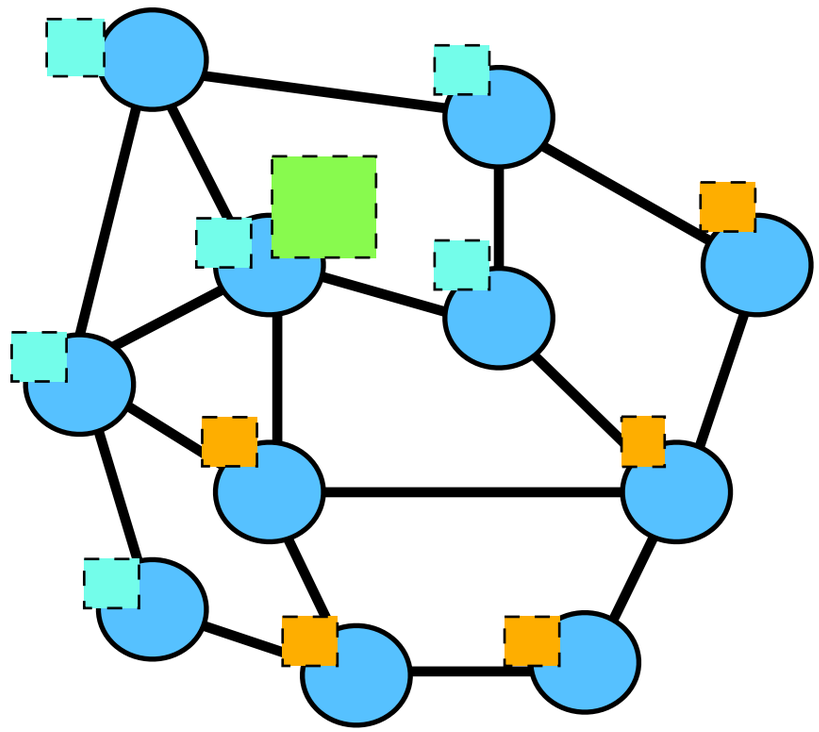


Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
    - *d* found while *c* was propagated.

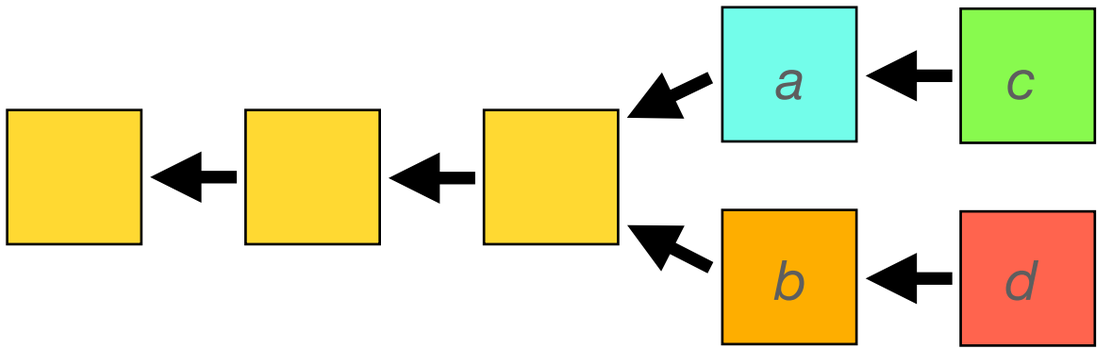




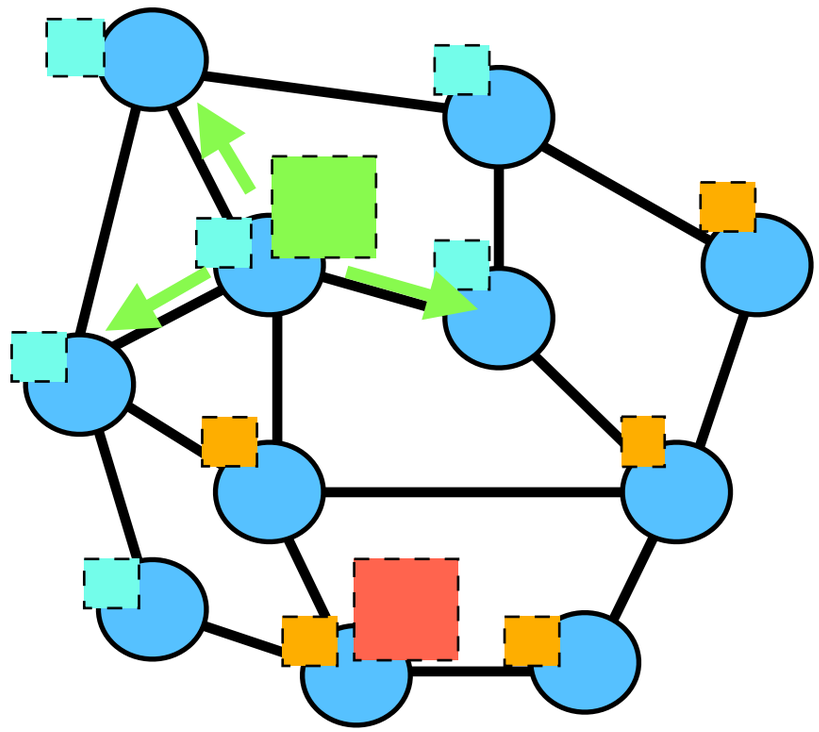
Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
    - *d* found while *c* was propagated.



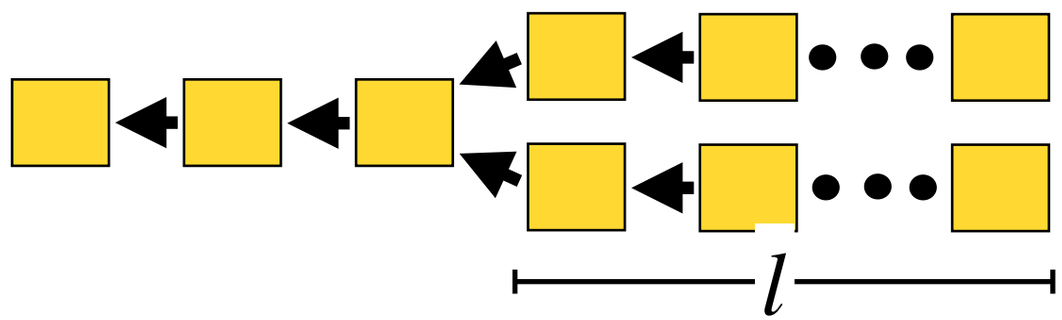
- Probability for second for smaller than the first.



Forks

Multiple forks

- Multiple forks may arrise after each other.
- Probability for second for smaller than the first.
- Probability for l forks decreases exponentially
- $P[l \times \text{fork}] \leq P[\text{fork}]^l$



Wait for l blocks to consider a transaction confirmed.

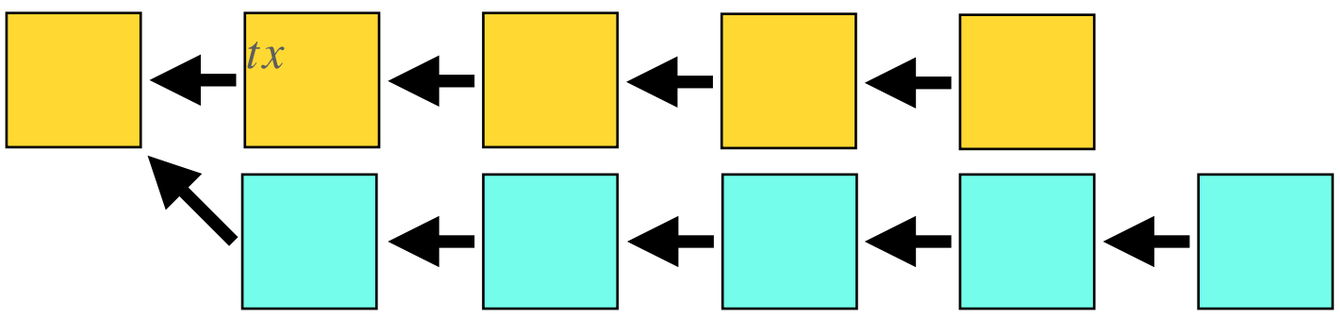
Attacks

Attacks

51% attack

- Assume the attacker has $\alpha > 50\,\%$ of the hashing power.
    - Attacker can grow a private chain faster than the public chain.

A private chain is a fork with blocks not propagated through the network.



Attacker can:
- **Double spend**
- **Get all the reward**

Attacks

Stubborn mining:

- Attacker does not follow longest chain rule.

Selfish mining:

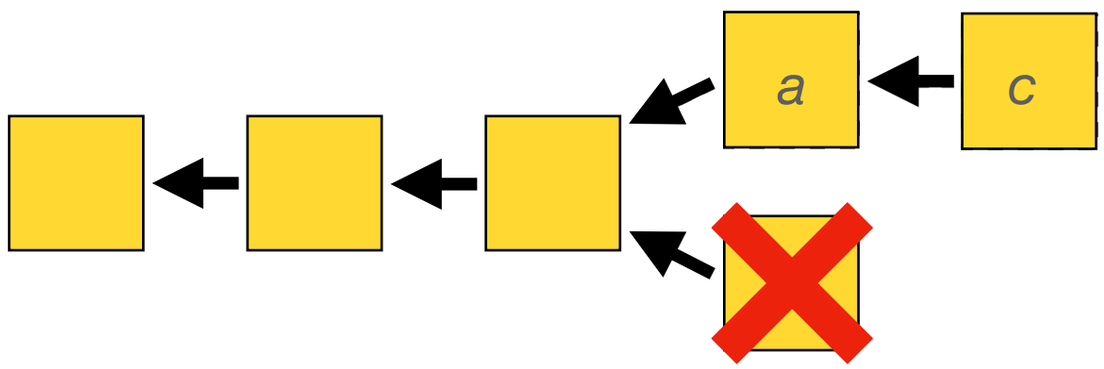
- Attacker keeps blocks secret.

Attacks

Selfish mining

Case 1, successfull attack:

1. attacker finds block *a,* keeps it secret
2. attacker finds block *c,* keeps it secret
3. other nodes find block *b* and propagate it
4. attacker propagates blocks *a* and *c*

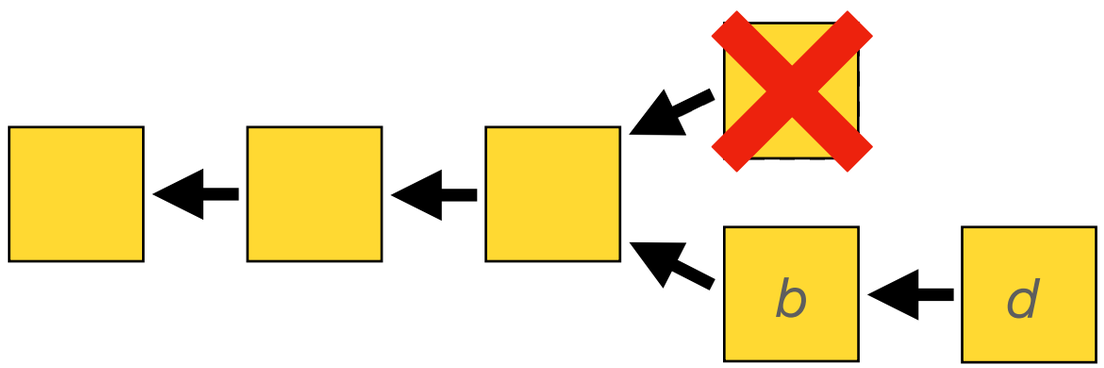


Attacks

Selfish mining

Case 2, unsuccessfull attack:

1. attacker finds block *a,* keeps it secret
2. other nodes find block *b* and propagate it
3. attacker propagates block *a*
4. other nodes find block *d* extending *b*

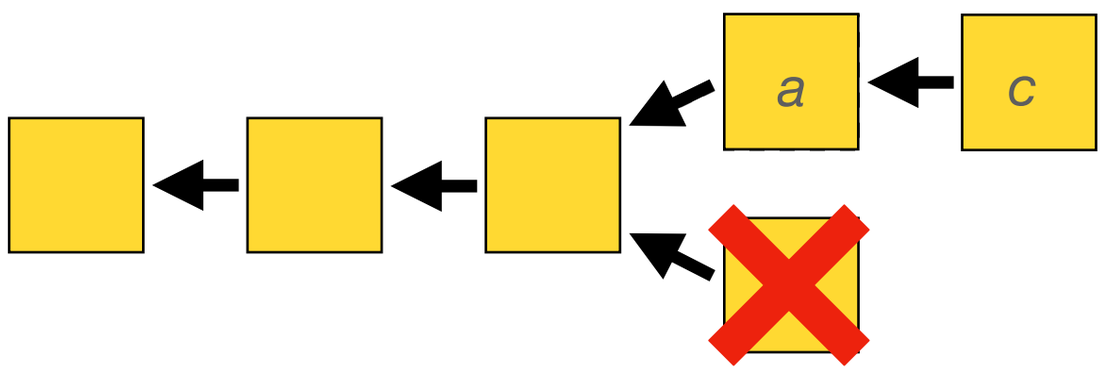


Attacks

Selfish mining

Case 3, kind of successfull attack:

1. attacker finds block *a,* keeps it secret
2. other nodes find block *b* and propagate it
3. attacker propagates block *a*
4. some node finds block *c* extending *a*

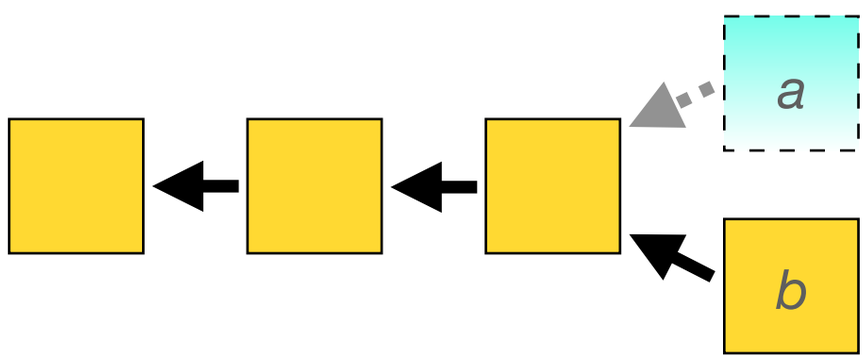


Attacks

Selfish mining

To get Case 3 instead of Case 2 attacker needs to

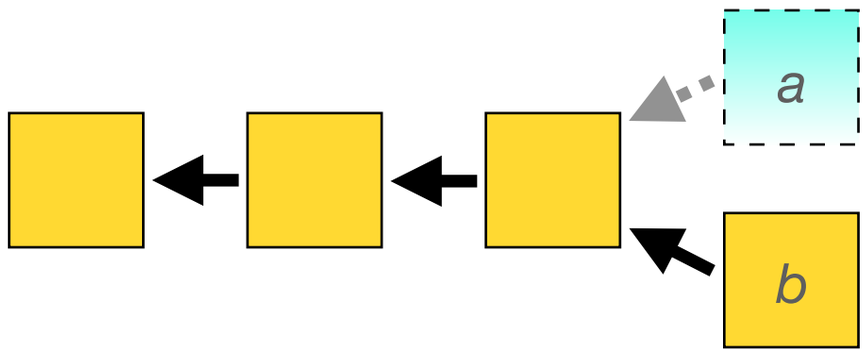
- detect new blocks fast
- propagate its block faster



Attacks

Selfish mining

- Attacker does not get more blocks, but others get less.
- Good control of network makes attack work better.



Attacks

Selfish mining

Algorithm 6 Selfish mining

Idea: Mine secretly, without immediately publishing newly found blocks
Let l_p be length of the public chain
Let l_s be length of the secret chain
if a new block b_p is published, i.e. l_p has increased by 1 **then**
if $l_p > l_s$ **then**
    Start mining on b_p
else if $l_p = l_s$ **then**
    Publish secretly mined block b_s
    Mine on b_s and immediately publish new block
else if $l_p = l_s - 1$ **then**
    Push all secretly mined blocks

Attacks

Selfish mining

α the attackers hashing power, and
γ be the attackers network power.

Selfish mining is profitable, if

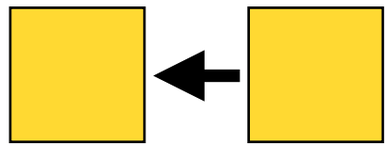
$\alpha > 0.33$

$\alpha > 0.25$ and $\gamma > 0.5$

$\alpha > 0$ and $\gamma = 1$

Selfish mining

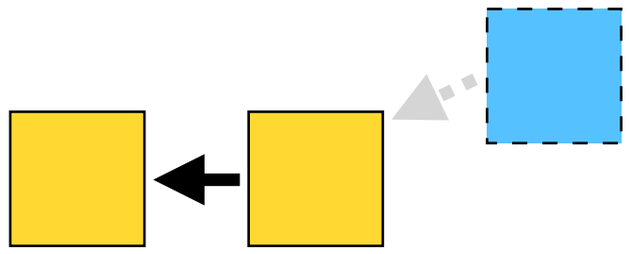
Example 1



Selfish mining

Example 1.0

i) The attacker finds a block and keeps it secret

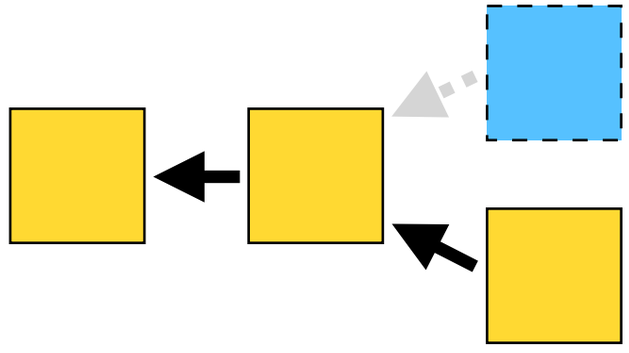


Selfish mining

Example 1.1

i) The attacker finds a block and keeps it secret

ii) The honest miners find a block



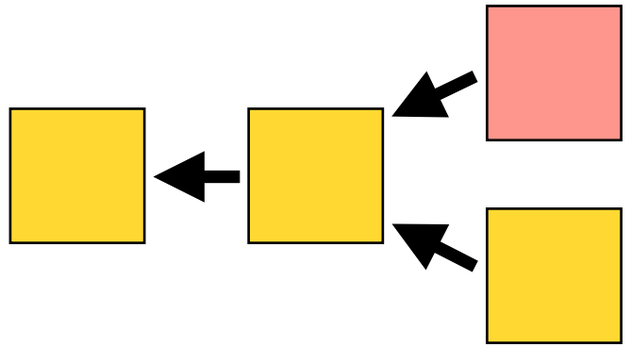
Selfish mining

Example 1.2

i) The attacker finds a block and keeps it secret

ii) The honest miners find a block

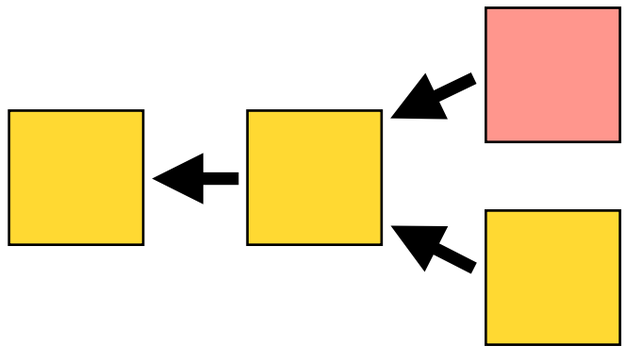
iii) The attacker publishes his block



Selfish mining

Example 1.3

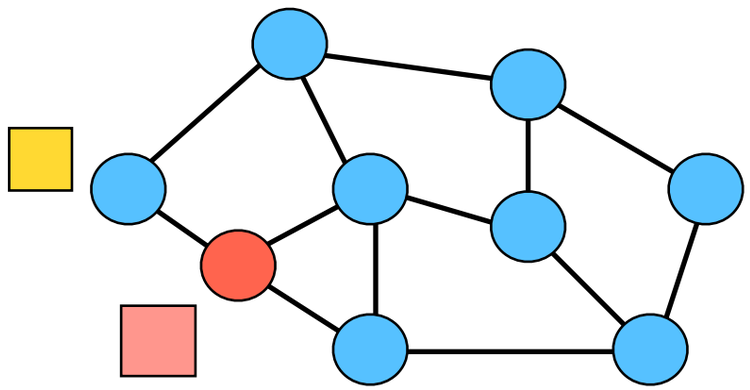
i) The attacker finds a block and keeps it secret



ii) The honest miners find a block

iii) The attacker publishes his block

iv) Honest miners mine on the block they see first.

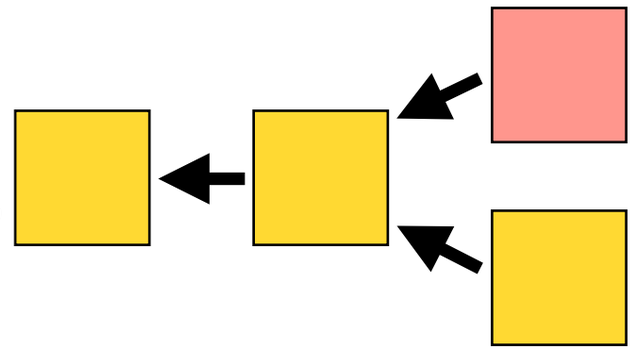


Attacker tries to publish his block faster.

Selfish mining

Example 1.3

iv) The attacker mines on top of his block.
Honest miners mine on the block they see first.



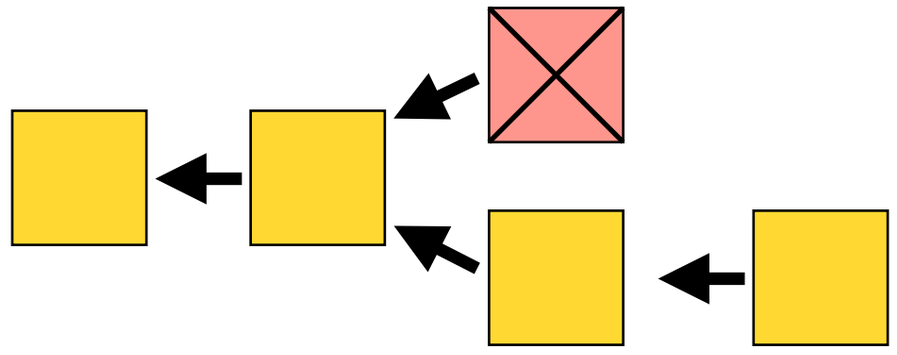
i) Honest miners with power $\gamma\beta$ mine on the attackers block

ii) Honest miners with power $(1-\gamma)\beta$ mine on the honest chain

Selfish mining

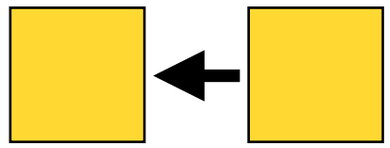
Example 1.4

iv) The honest miners chain is extended the attackers block is discareded



Selfish mining

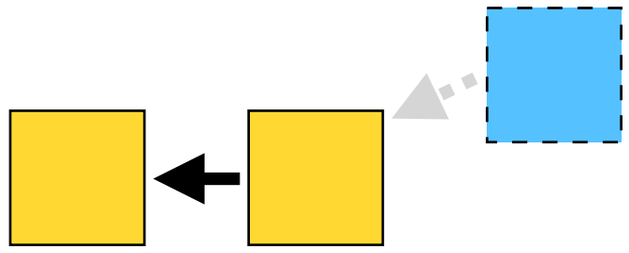
Example 2



Selfish mining

Example 2.0

i) The attacker finds a block and keeps it secret

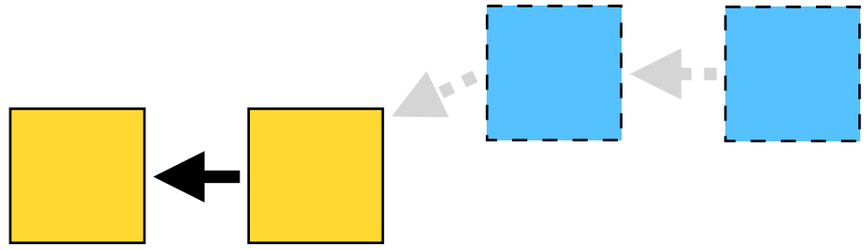


Selfish mining

Example 2.1

i) The attacker finds a block and keeps it secret

ii) The attacker finds another block and keeps it secret

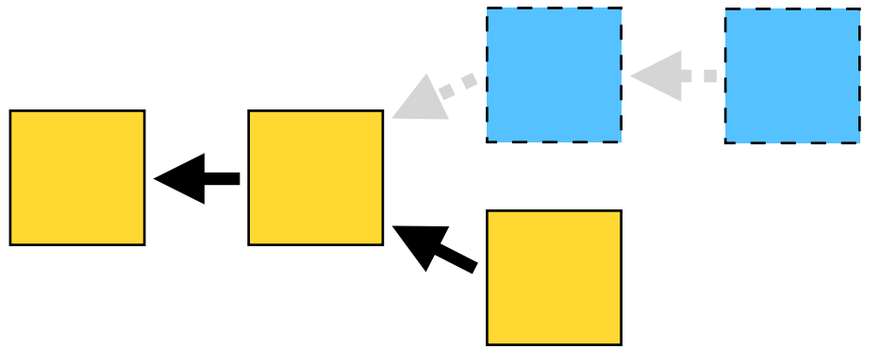


Selfish mining

Example 2.2

i) The attacker finds a block and keeps it secret

ii) The attacker finds another block and keeps it secret



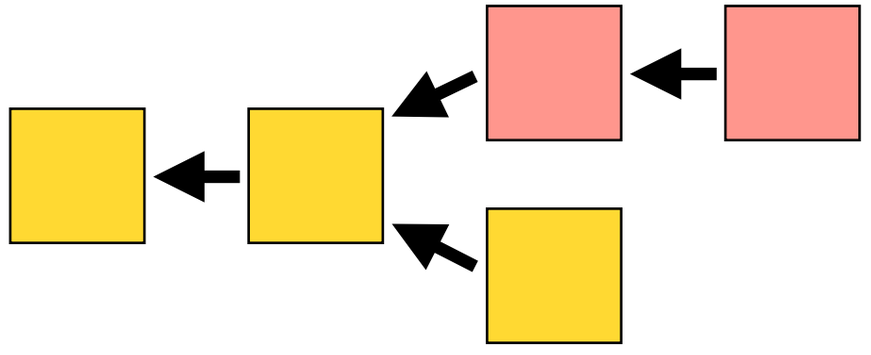
iii) The honest miners find a block

Selfish mining

Example 2.3

i) The attacker finds a block and keeps it secret

ii) The attacker finds another block and keeps it secret



iii) The honest miners find a block

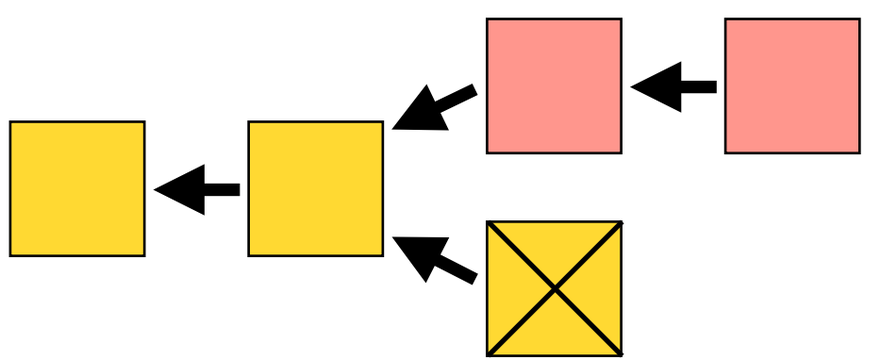
iv) The attacker publishes both his blocks

Selfish mining

Example 2.4

i) The attacker finds a block and keeps it secret

ii) The attacker finds another block and keeps it secret



iii) The honest miners find a block

iv) The attacker publishes both his blocks

v) The honest miners block is discarded

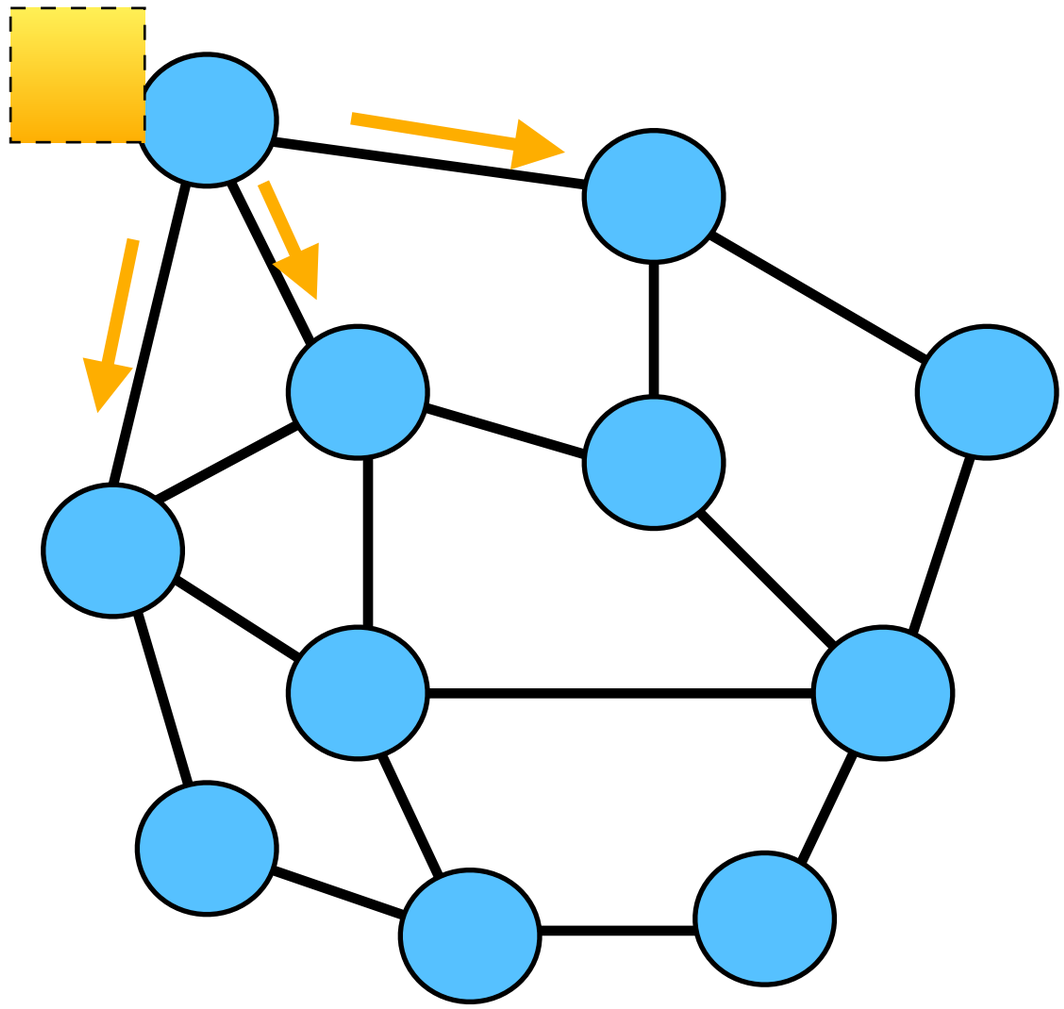
Attacks

Delivery denial

Broadcast block:

- Broadcast inventory message including block hash
- Receiving new inventory, request block
- Send block

Block is only send from one neighbor



Attacks

Delivery denial

Broadcast block:

- Broadcast inventory
- Request block
- Send block

Attack

- Broadcast inventory
- Do not send out blocks

  Victims wait for timeout.

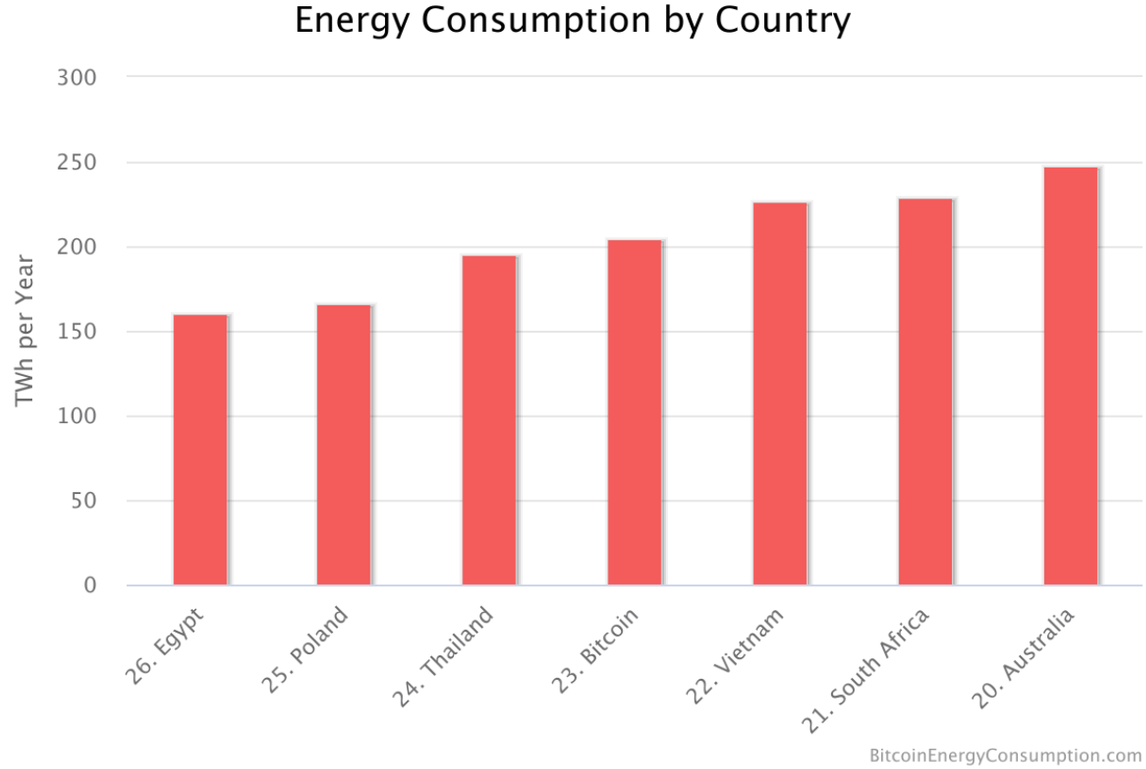
Bitcoin

Downsides

Throughput at most 7tx per second

Confirmation latency approx 1h

Enormous energy consumption



GHOST

Greedy heaviest-observed subtree

Increazing throughput with reparametrization gives more forks!

- bad for security (e.g. selfish mining)

GHOST: *Instead of longest chain, always select block with the heaviest subtree* (i.e. most blocks in subtree).

GHOST

Greedy heaviest-observed subtree

Increazing throughput with reparametrization gives more forks!

- bad for security (e.g. selfish mining)

GHOST: *Instead of longest chain, always select block with the heaviest subtree* (i.e. most blocks in subtree).

- same as Longest chain if a single fork
- in selfish mining, attackers chain does not have forks
- causing forks, e.g. through network attack does not help attacker

GHOST

Greedy heaviest-observed subtree

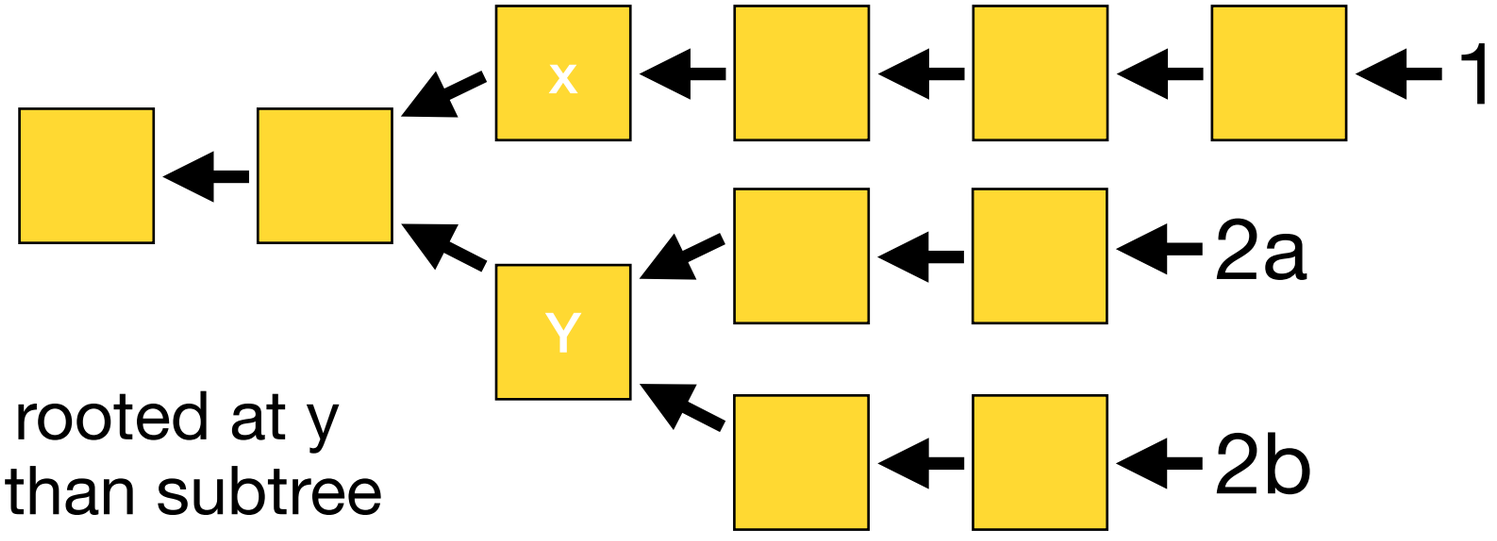
Example:

Longest chain rule:

Mine at 1

GHOST:

Mine at 2a or 2b



- Because subtree rooted at y has more blocks than subtree rooted at x