Mining Pools and Security of Bitcoin

Stefan Dziembowski
University of Warsaw

Plan

1. Mining pools 
2. Security of Bitcoin

Mining pools

Miners create cartels called

the mining pools

This allows them to reduce the variance of their income.

Note

The **total hashrate** of the Bitcoin system as of 6.12.2018

number of blocks in 1 year

$$\frac{36{,}000{,}000 \text{ terahash / s}}{14 \text{ terahash/ s}} \approx 2{,}571{,}428 \approx 49 \cdot (365 \cdot 24 \cdot 6)$$

The **hashrate of the Antminer S9 mining rig** (USD 3,000)

The user has to wait on average around **49 years** to mine a block (**even if the difficulty does not increase**!)

The general picture

The mining pools are **operated centrally** or are designed in a **p2p** way.

Some of the mining pools **charge fees for their services**.

E.g. if the operator got **12.5 BTC** from mining then he will share **12.5 BTC – *fee*** among them
(and keep the ***fee*** to himself)

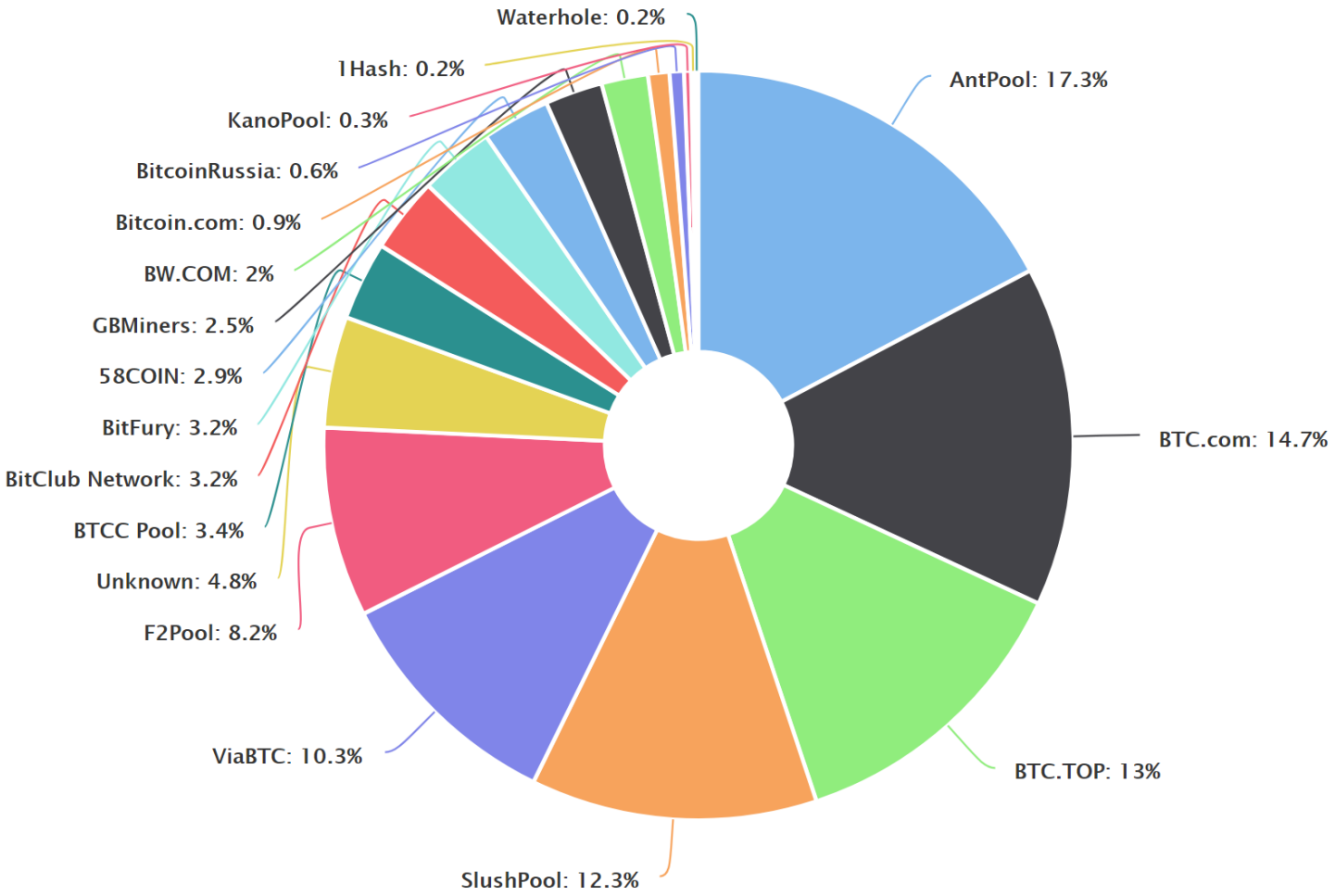
In other words:

- the **expected revenue** from pooled mining **is slightly lower** than the expected revenue from solo mining,
- but the **variance is significantly smaller**.

Tricky part: how to prevent cheating by miners? How to reward the miners?

Popular mining pools

As of Dec,



How to design a mining pool?

Simple idea:

this includes a coinbase transaction transferring the reward to **pk**.

miner

a list of transactions T^i and a hash $H(B_i)$

mining pool operator

tries to find **nonce** such that $H(\text{nonce}, H(B_i), T^i)$ starts with **n** zeros

current hardness parameter

if he finds such **nonce** then he sends it to the operator

Problem
How to verify how much work a miner really did?

once **nonce** is found by ***some*** of the pool members each of them is **rewarded proportionally to his work**.

A solution: "Proportional method"

a list of transactions T^i and a hash $H(B_i)$

miner

tries to find **nonce** such that $H(\text{nonce}, H(B_i), T^i)$ starts with **n** zeros

if he finds such a **nonce** then he sends it to the operator

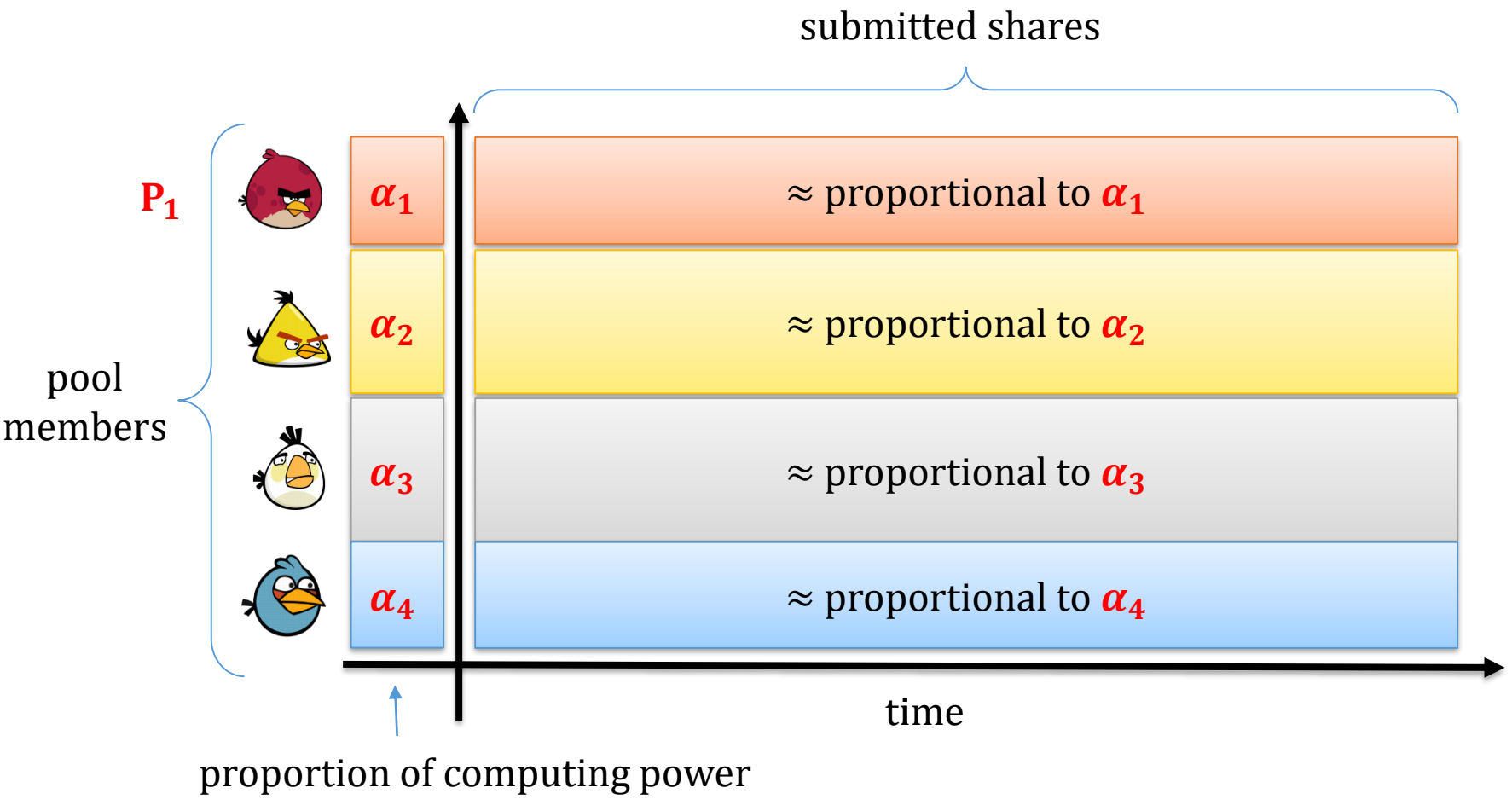
he also submits the "**partial solutions**", i.e. values **nonce** such that $H(\text{nonce}, H(B_i), T^i)$ starts with n' zeros

$n' \ll n$

mining pool operator

The "**amount of work**" is measured by the **number of "partial solutions"** submitted.

Works if the miners don't change the pools

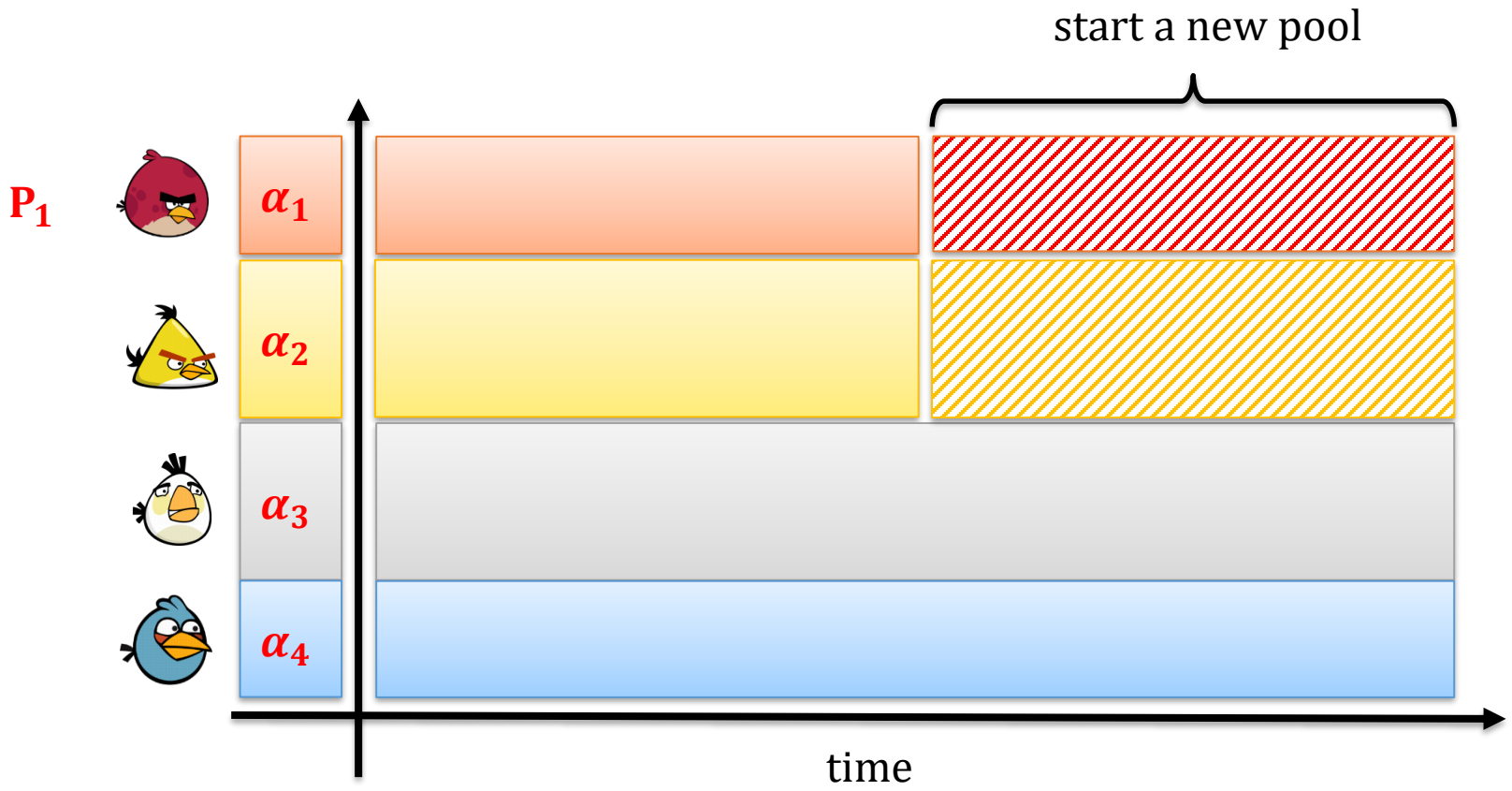


probability of that this pool wins: $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$

reward for P_1 in case it wins: $\mathbf{BTC\ 12.5} \cdot \frac{\alpha_1}{\alpha_1+\alpha_2+\alpha_3+\alpha_4}$

expected reward for P_1: $\mathbf{BTC\ 12.5} \cdot \alpha_1$

What if the miners change pools?



Now the expected revenue of P_1 is a sum of

- α_1 (from the new pool)
- plus the revenue from the old pool.

A problem with the proportional method: “Pool hopping”

It is profitable to escape from pools with lots of shares submitted.

(since such pools have a lot of “mouths to feed” there)

A solution: do not rewarding each share equally

Example: **Slush's method**

Use a **scoring function** that assigns to each share a **score s**.

Then assign rewards proportionally to the score.

Slush's scoring function: $s = \exp\left(\frac{T}{C}\right)$.

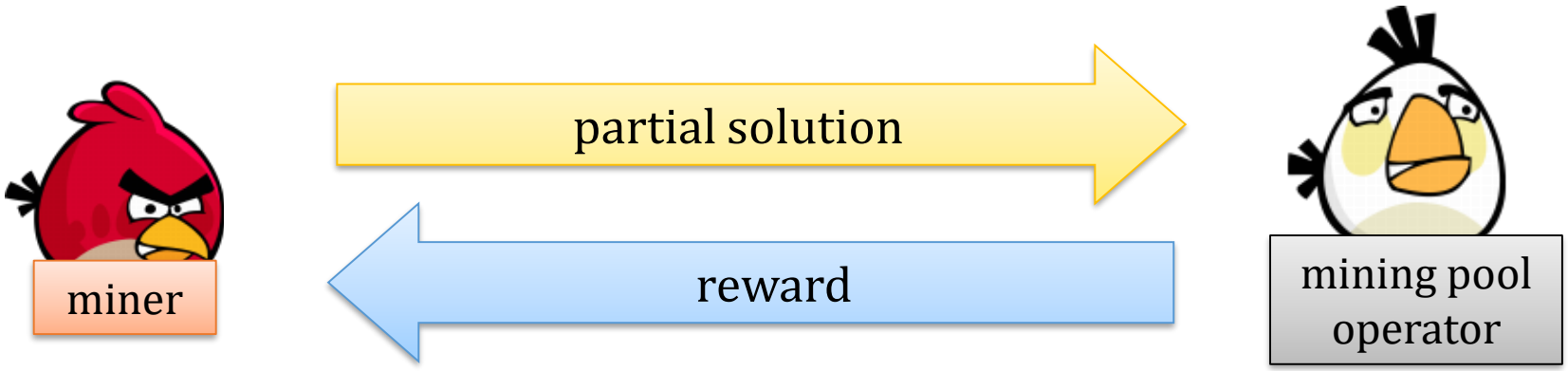
time since the beginning of this "round"

some constant

Intuitively: this gives advantage to miners who joined late.

Another solution: “Pay-per-share”

The operator pays per each partial solution no matter if he managed to extend the chain.



Major drawback: risky for the operator.

He needs to have some reserves to cover the potential losses.

Other methods

Score-based: Geometric method, Double geometric method, Pay-per-last-N-shares,

Improved pay-per-share: Maximum pay-per-share, Shared maximum pay-per-share, Equalized , Shared maximum pay-per-share,

⋮

(see **[Meni Rosenfeld, *Analysis of Bitcoin Pooled Mining Reward Systems*, 2011]**, **[Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden, *Incentive Compatibility of Bitcoin Mining Pool Reward Functions*]**)

How secure are these methods

We can assume that the mining **pool operator is honest**, since he has a reputation.

Much **harder** to avoid: attacks from **malicious miners**.

We discuss two of them:

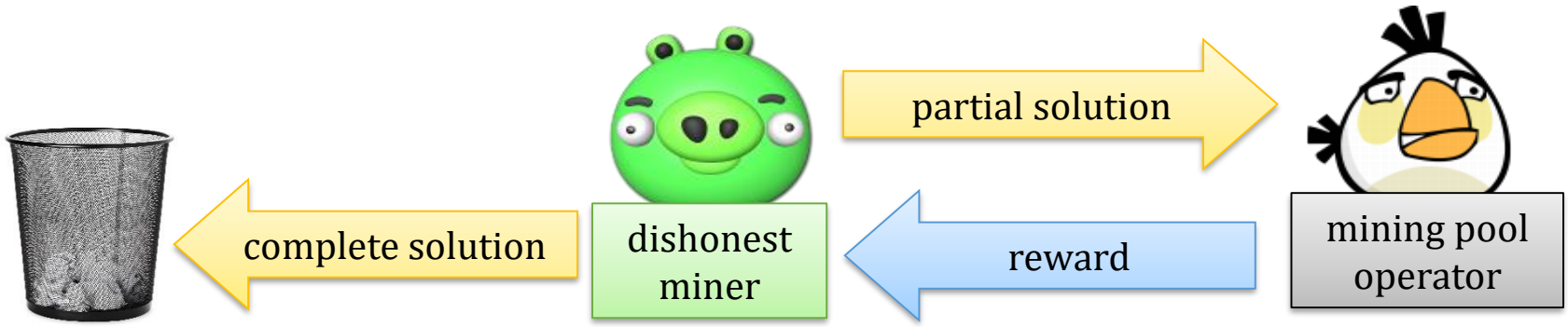
- **"sabotage",**
- **"lie-in-wait".**

a bit similar to the **selfish-mining attack** on Bitcoin that we discuss later

Both of them are based on **withholding certain blocks**.

A “Sabotage”attack on mining pools

Submit only the **partial** solutions.



Results:

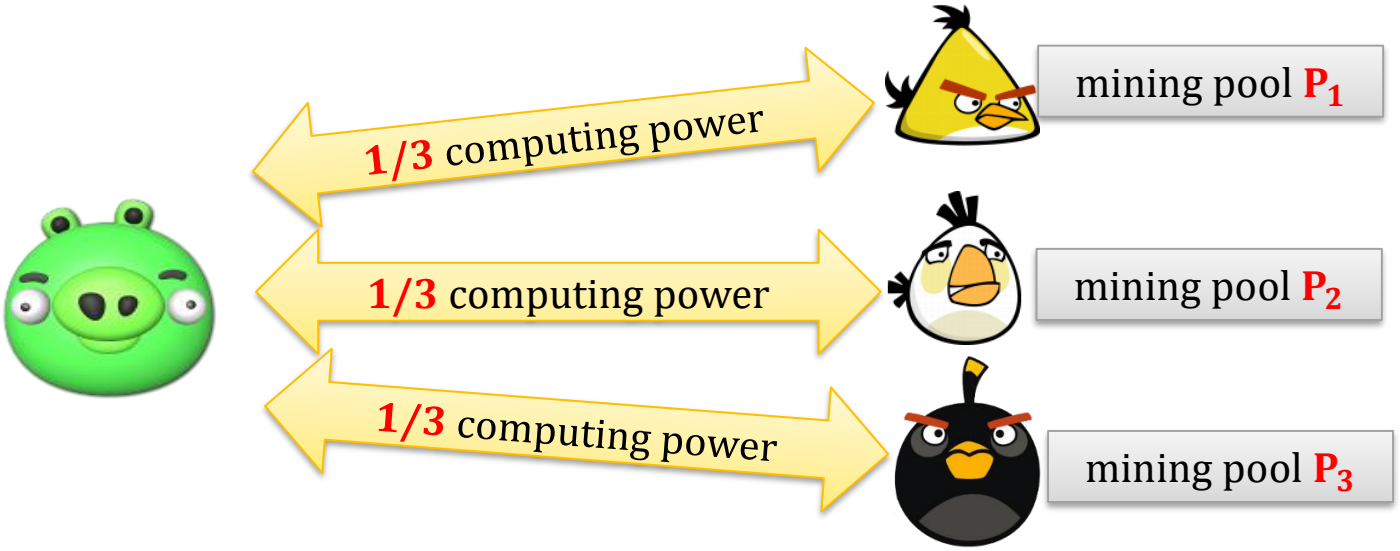
- the pool looses money
- the dishonest miner doesn’t earn anything (also looses a small amount)

Adversary’s goal: make the mining pool bankrupt
(e.g. he owns a competing pool).

It is rumored that in **June 2014** such an attack was executed against the mining pool Eligius**. Estimated loses: 300 BTC**.

Another attack: "lie-in-wait"

Mine for several mining pools:



Intuition: P_2 is a very likely winner

Once you find a solution for P_2 (say):

1. wait with submitting it
2. mine only for P_2
3. submit the solution to P_2 after some time.

It can be formally shown that this is profitable (see [Rosenfeld, 2011])

Can we have a mining pool without an operator?

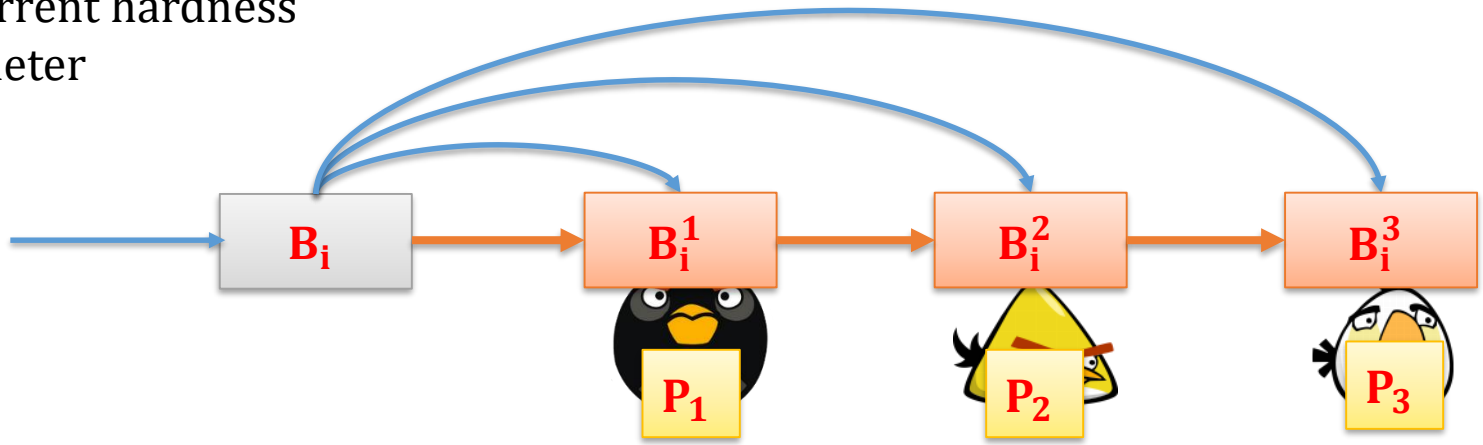
(remember: the operators typically charge a fee).

Answer: yes, using the

Peer-to-peer mining pools.

Peer-to-peer mining pools

n – current hardness parameter



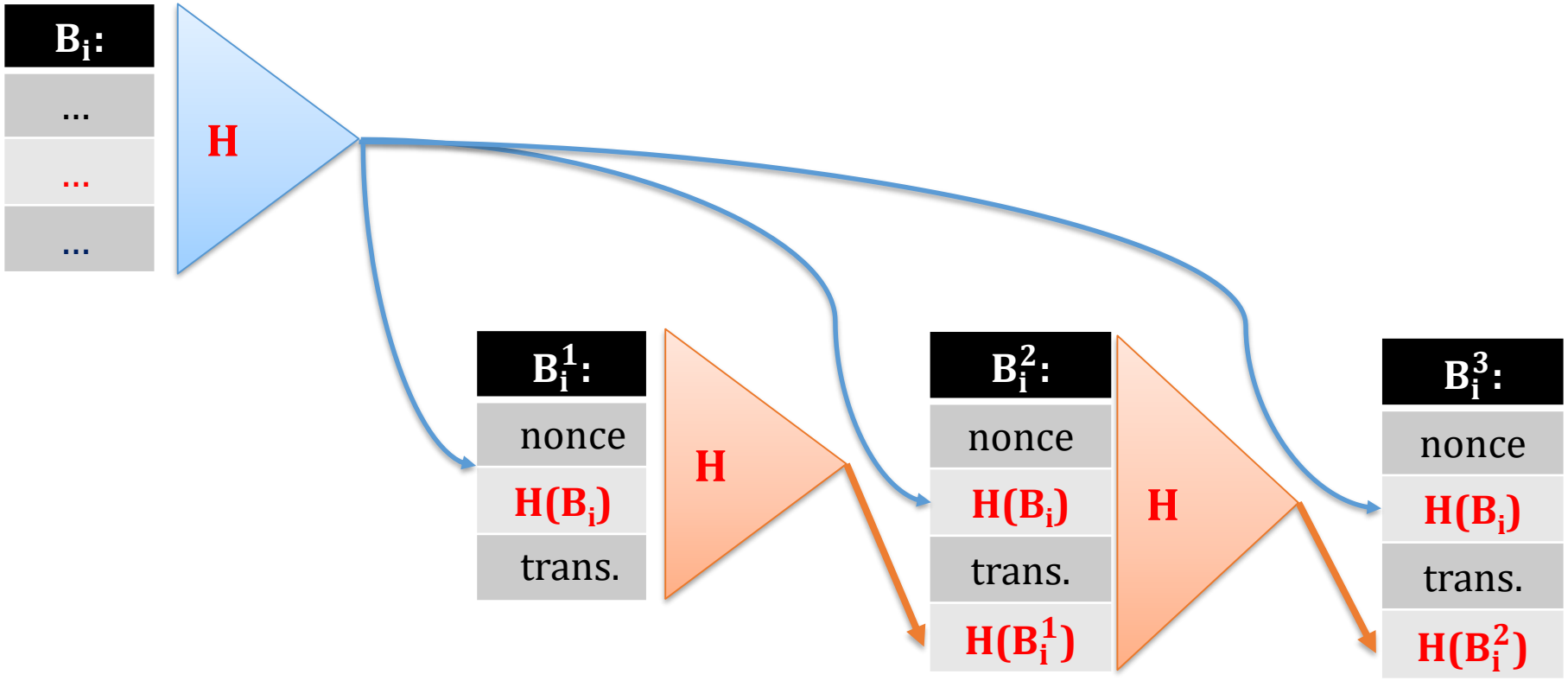
General idea: the miners create a blockchain with hardness parameter $n' \ll n$ on top of the last block B_i.

Every $B_i^1, B_i^2, \ldots$ is a valid extension of B_i (except that the hardness may be smaller than **n**).

The parameter n' is chosen is such a way that a new block appears often (say: **once per 30 sec**.)

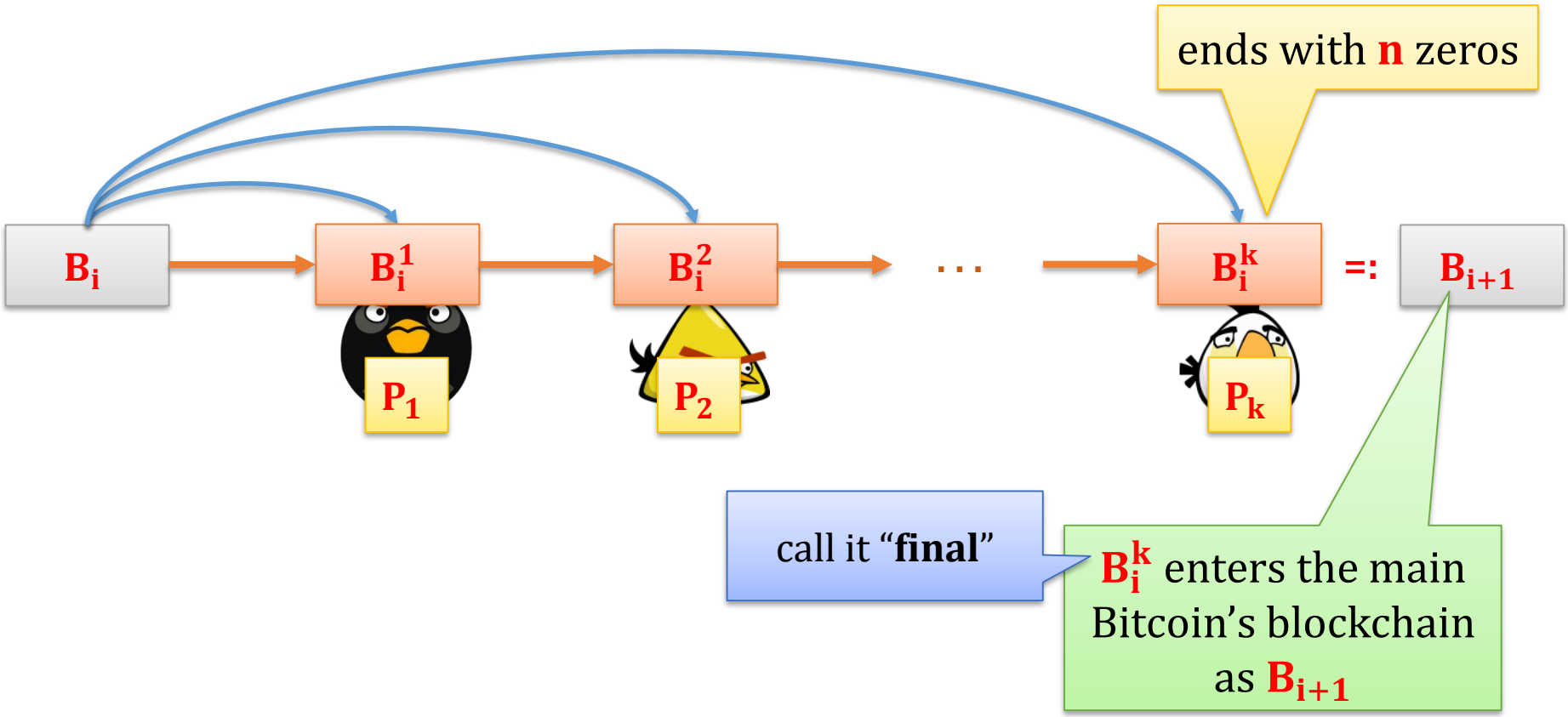
Hence: this has to be done using some **other fields in the block** (fortunately: blocks have space for this).

How is it done technically?

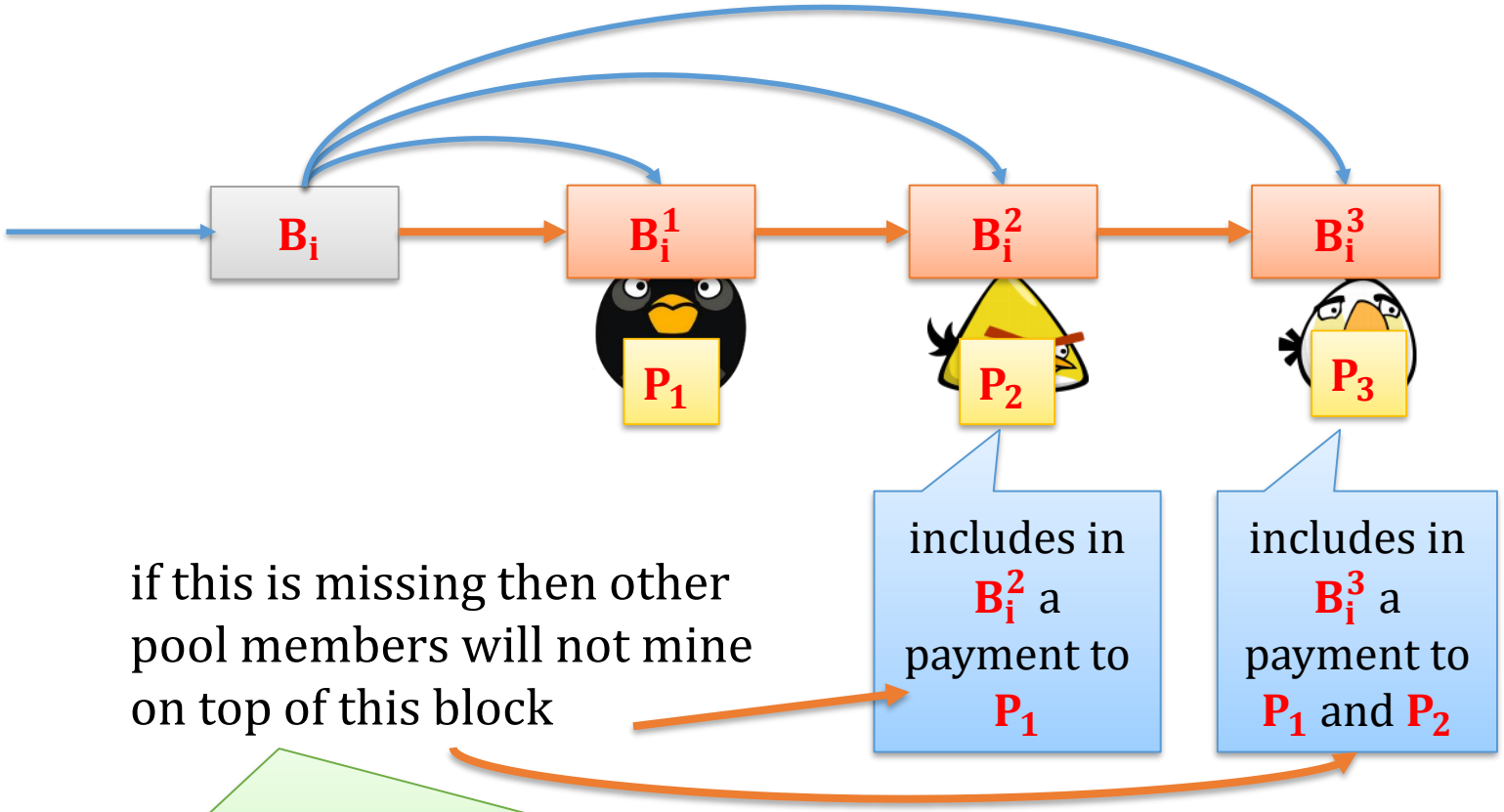


Bitcoin blocks contain fields that can be used to store $H(B_i^j)$'s.

Finally someone will find a block that extends B_i according to Bitcoin rules.



How to divide the revenue from mining?



Note: the miner does *not* know in advance if his block will be final. He has to choose the payment information **beforehand**.

Plan

1. Mining pools
2. Security of Bitcoin 

Possible attack goals

"Goldfinger attack"

Note: this can be done e.g. by a spectacular fork that lasts just for a few hours…

- **double spending**,
- get **more money from mining** than you should,
- get **a full control over the blockchain**
- "**short selling**" – bet that the price of BTC will drop and then destroy the system (to make the price of BTC go to zero),
- someone (government?) interested in **shutting Bitcoin down**…

What we do (not) know about Bitcoin's security?

Things to consider:

1. Programming errors
2. Transaction malleability
3. Lack of anonymity
4. Hardware mining
5. The "51% attack"
6. Mining pools
7. Selfish mining
8. Potential threats
9. Problems with key storage

Some notable cases of **programming errors**

- a block *74638* (**Aug 2010**) contained a transaction with **two outputs summing to over 184 billion BTC** – this was because of an **integer overflow** in Bitcoin software

  (solved by a software update and a "manual fork")
  one double spending observed (worth 10.000 USD).

- a **fork** at block *225430* (**March 2013**) caused by an **error in the software update** of Bitcoin Core
  (lasted 6 hours, solved by reverting to an older version of the software)

Moral: nothing can be really "completely distributed".
Sometimes human intervention is needed…

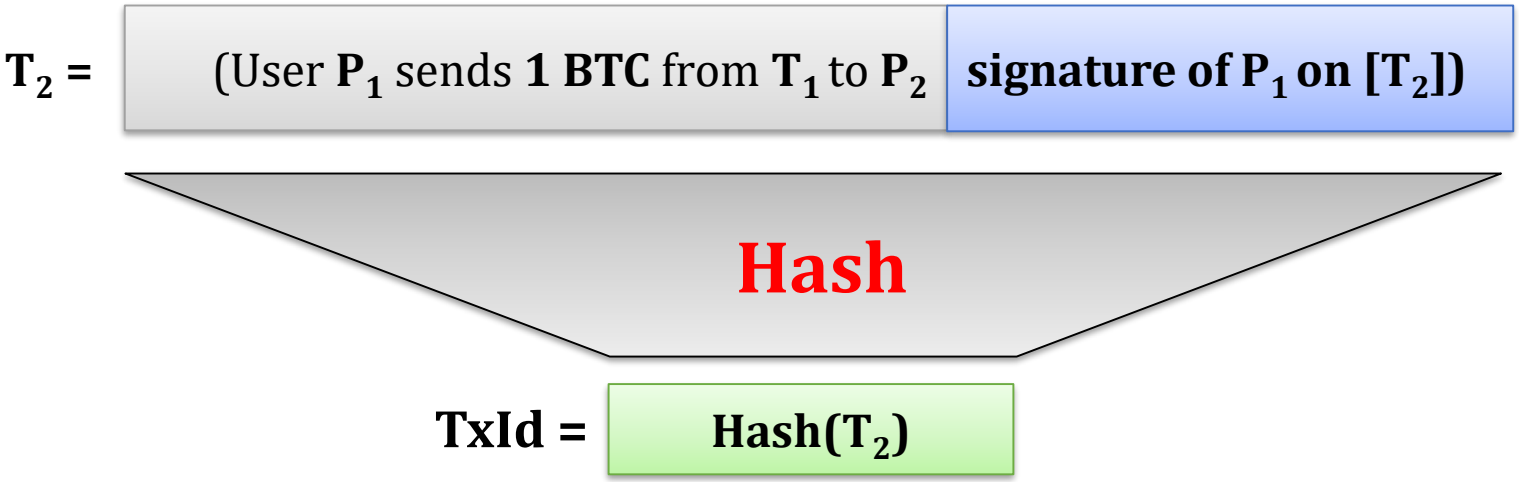
What we do (not) know about Bitcoin's security?

Things to consider:

1. Programming errors
2. Transaction malleability
3. Lack of anonymity
4. Hardware mining
5. The "51% attack"
6. Mining pools
7. Selfish mining
8. Potential threats
9. Problems with key storage

Transaction Malleability

Problem: transactions are identified by their **hashes**



Hence one can change **TxId** by mauling the signature:

How to do it?

Bitcoin uses **ECDSA** signatures. Hence:

$\sigma = (r,s)$

is a valid signature on

M w.r.t. **pk**

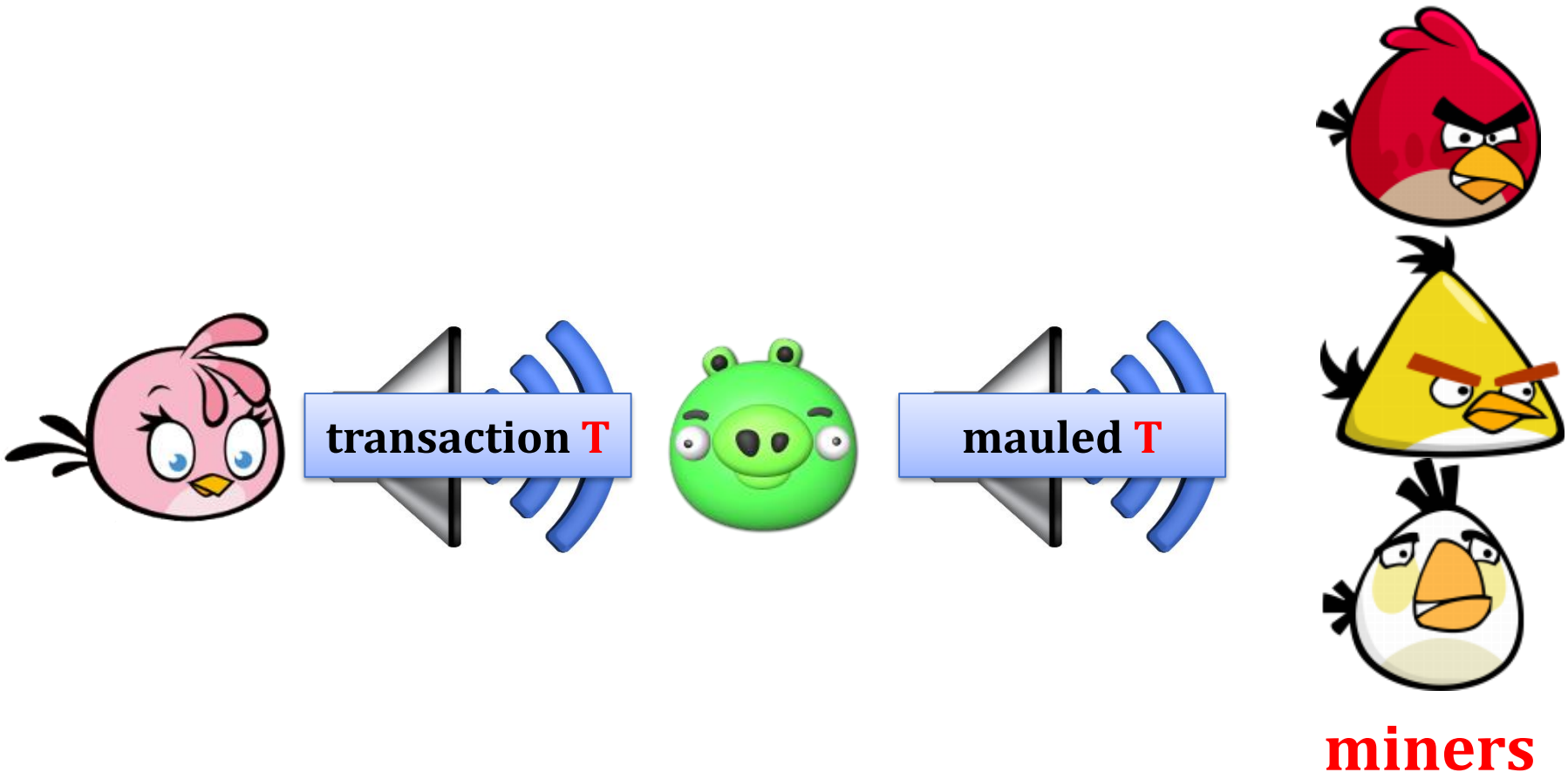
$\sigma' = (r, -s \pmod N)$

is a valid signature on

M w.r.t. **pk**

Other methods also exists…

What can the adversary do?



[Andrychowicz et al 2015]: very easy to perform in practice.

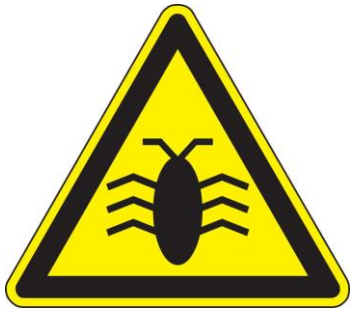
Is it a problem?

Often: **NO**

(the mauled transaction is semantically equivalent to the original one)

When things can go wrong?

- Bitcoin contracts
- buggy software



Claimed attack on MtGox

A

deposits 1 BTC

withdraws 1 BTC

MT.GOX

transaction
T = "MtGox sends 1 BTC to A"

transaction
T' = mauled transaction T

Since **MtGox** cannot see a transaction with **TxId Hash(T)** in the blockchain.

Thus it concludes that the **transaction did not happen.**
(so **A** can double spend)

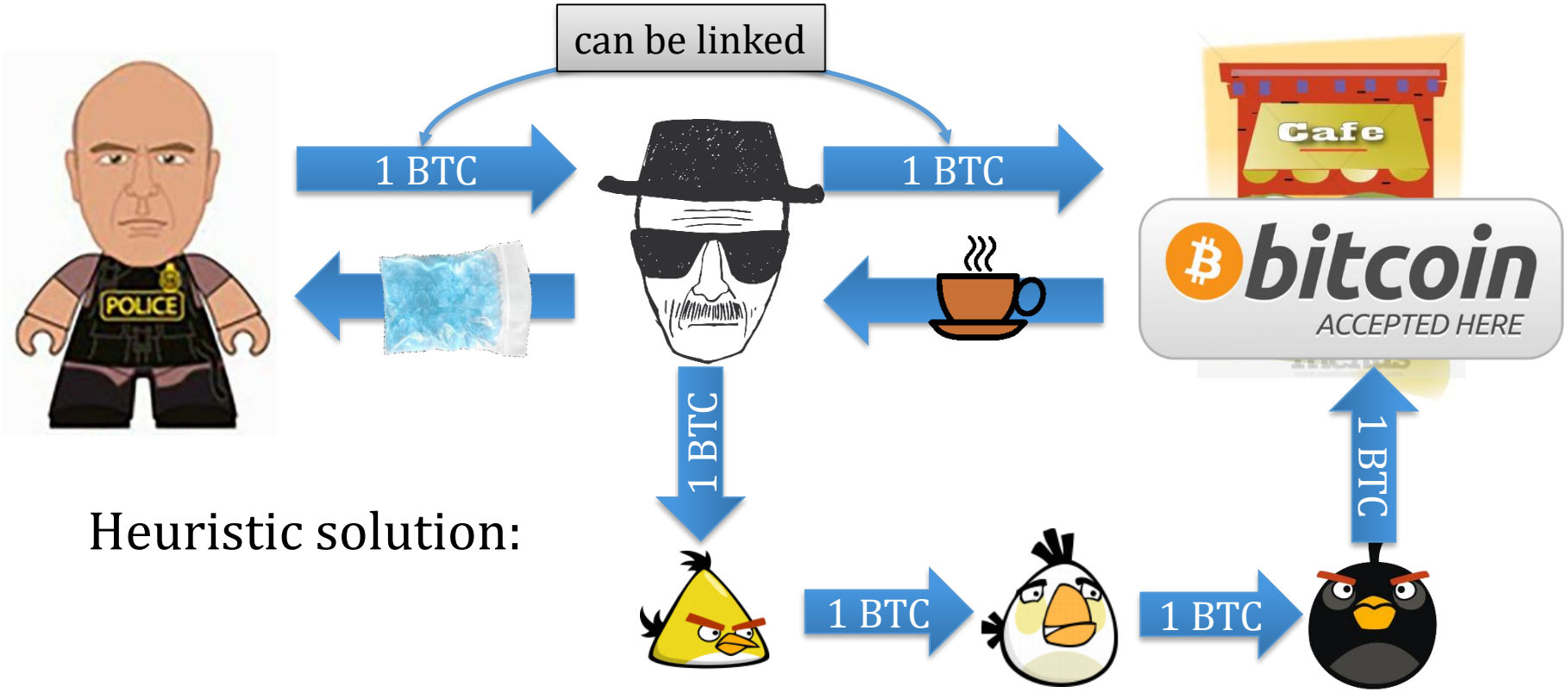
[Decker and Wattenhofer, ESORICS 2014]: this is probably not true.

What we do (not) know about Bitcoin's security?

Things to consider:

1. Programming errors
2. Transaction malleability
3. Lack of anonymity
4. Hardware mining
5. The "51% attack"
6. Mining pools
7. Selfish mining
8. Potential threats
9. Problems with key storage

One obvious problem: lack of anonymity



Heuristic solution:

Can sometimes be de-anonymized…

What we do (not) know about Bitcoin's security?

Things to consider:

1. Programming errors
2. Transaction malleability
3. Lack of anonymity
4. Hardware mining 
5. The "51% attack"
6. Mining pools
7. Selfish mining
8. Potential threats
9. Problems with key storage

Another problem/feature: hardware mining

History of mining: CPU → GPU → FPGA → ASIC

Examples of ASIC mining rigs:

AntMiner S7	AntMiner S9	Avalon6
		
Advertised Capacity: 4.73 Th/s	**Advertised Capacity:** 13.5 Th/s	**Advertised Capacity:** 3.5 Th/s
Power Efficiency: 0.25 W/Gh	**Power Efficiency:** 0.098 W/Gh	**Power Efficiency:** 0.29 W/Gh
Weight: 8.8 pounds	**Weight:** 8.1 pounds	**Weight:** 9.5 pounds
Guide: Yes	**Guide:** Yes	**Guide:** No
Price: $479.95	**Price:** $1,987.95	**Price:** $499.95
Buy from amazon.com	Buy from amazon.com	Buy from amazon.com
Appx. BTC Earned Per Month: 0.1645	**Appx. BTC Earned Per Month:** 0.3603	**Appx. BTC Earned Per Month:** 0.1232

Drawbacks of the hardware mining

1. Makes the whole process ``**non-democratic**”.

2. Easier to attack by **very powerful adversary**?

3. **Excludes some applications** (mining a as “micropayment”).

Advantages of the hardware mining

- Security against **botnets**.

- Makes the miners interested in the **long-term stability** of the system.

How "long term"?

the total hashrate can go up almost **100x** in one year…

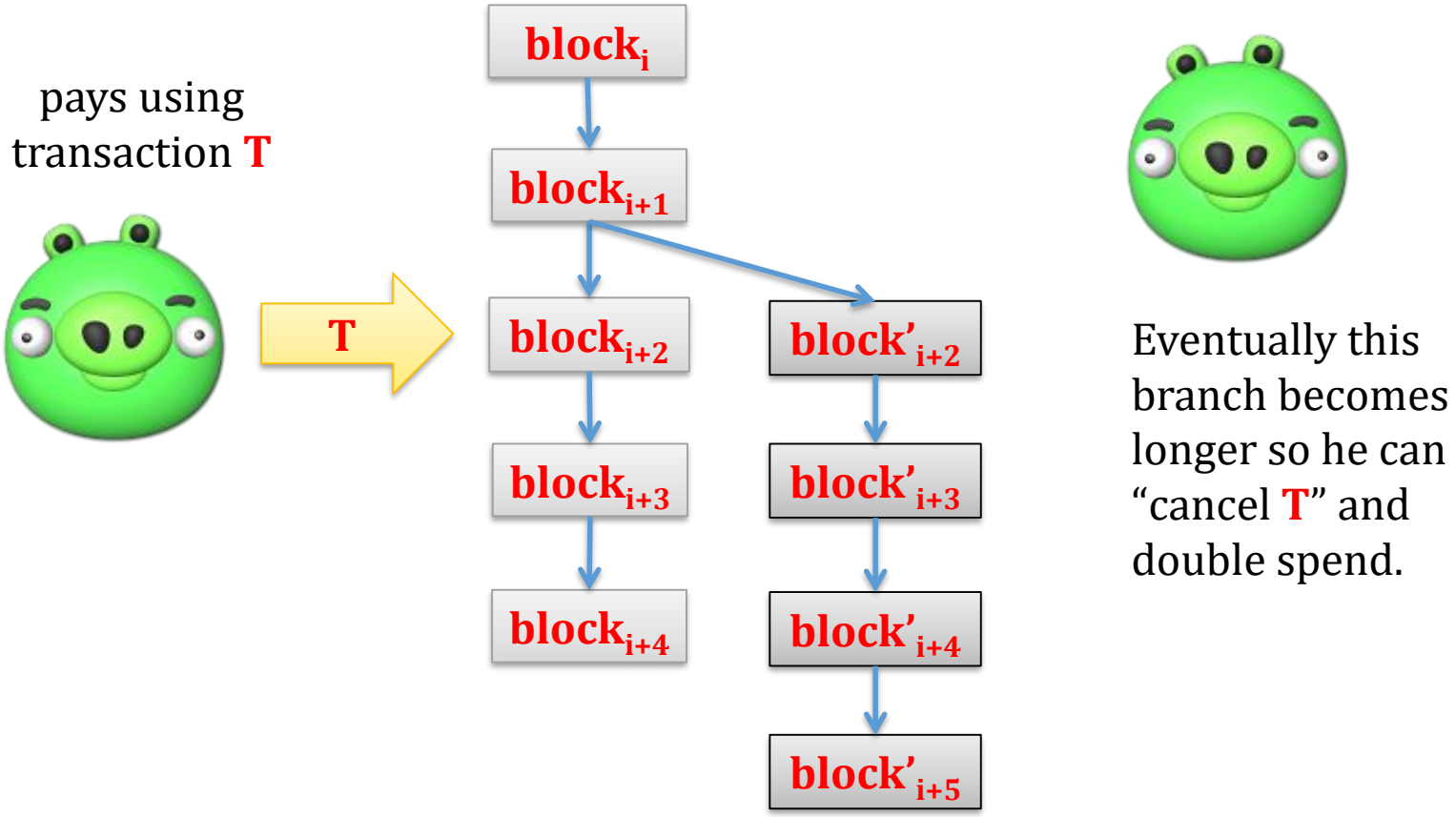
What we do (not) know about Bitcoin's security?

Things to consider:

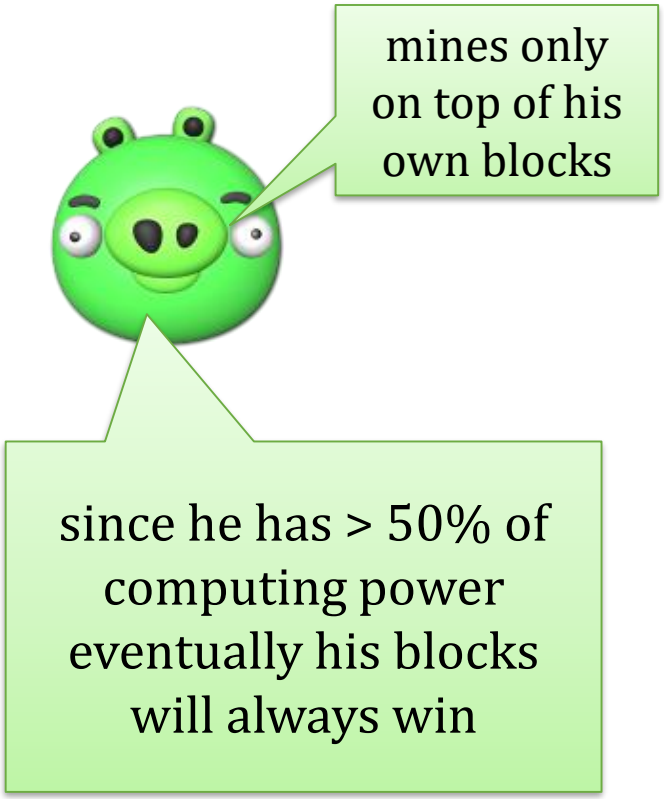
1. Programming errors
2. Transaction malleability
3. Lack of anonymity
4. Hardware mining
5. The "51% attack"
6. Mining pools
7. Selfish mining
8. Potential threats
9. Problems with key storage

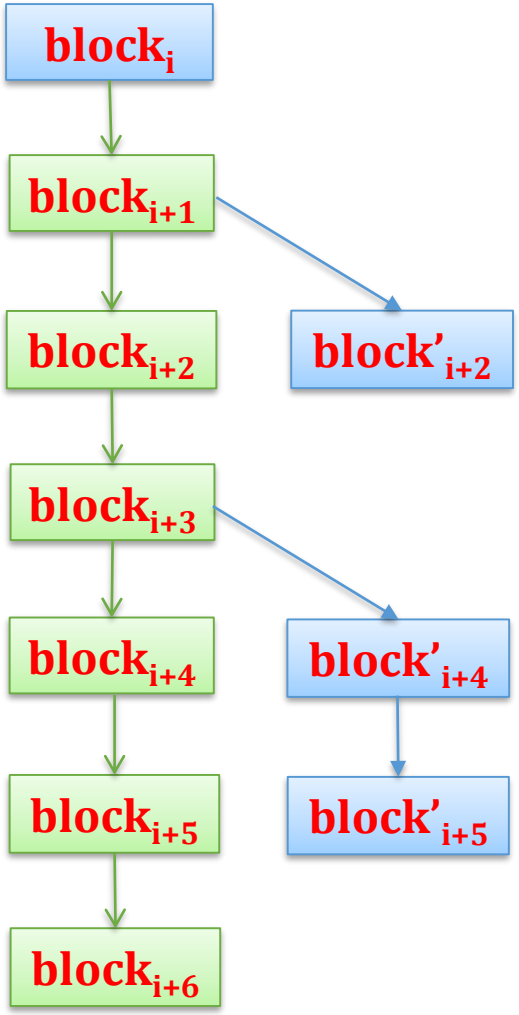
Easy to see

An adversary that controls majority of computing power can always break the system.



Moreover: “a 51% adversary can get a full control over the blockchain”





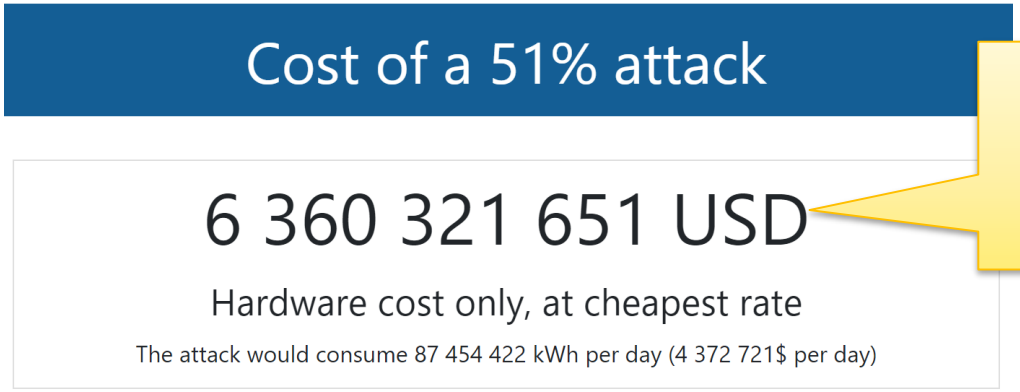
What can he do then?

1. **Censor** transactions
2. Publish only **empty blocks**
3. Ask for **very high fees**

⋮

What is the cost of the “51% attack”?

Current estimate (gobitcoin.io/tools/cost-51-attack):



around **9%** of Bitcoin’s market capitalization

Observation: maybe a rich adversary would not even need to pay this.

(it’s enough that he convinces everybody that he is really going to attack Bitcoin)

Why?

How to break Bitcoin?

1. Announce that you are going **to invest $4 billion to break Bitcoin**.
2. Start **buying second-hand hardware** from miners
3. Once the miners get convinced that BTC will be broken **they will sell it to you very cheaply**
4. So the total cost of your attack will be **much less than $4 billion**

Will the miners sell their equipment to you?

From the point of view of **game theory**: they should…

What is really our security assumption?

1. No cartel controls the majority of the computing power,
   or
2. The majority of participants is **100%** honest.

"As long as a **majority** of CPU power is controlled by nodes that are **not cooperating** to attack the network, they'll generate the longest chain and outpace attackers"

?

we proposed a peer-to-peer network using proof-of-work to record a public history of transactions that quickly **becomes computationally impractical for an attacker** to change **if honest nodes control a majority of CPU power**

In order for the Bitcoin to work we need a following (**strong**) assumption:

The majority behaves honestly even if it has incentives not to do so.

Is it realistic?

enthusiast:

Yes, since the majority is interested in maintaining the system

sceptics:

No, since this is not how capitalism works…

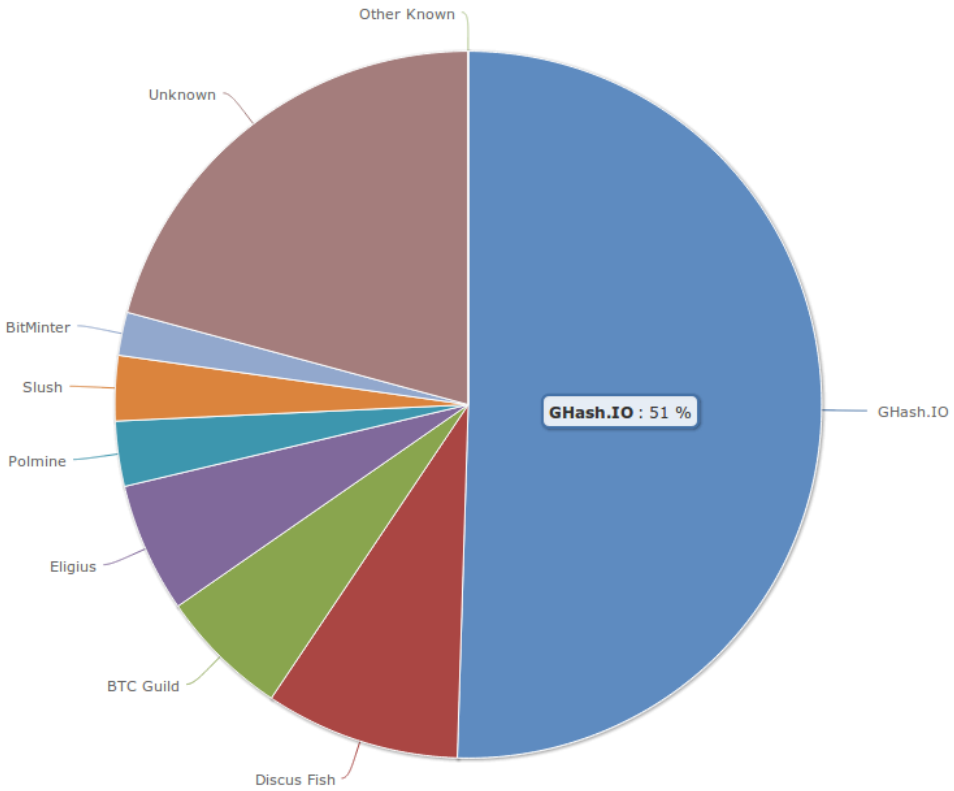
What we do (not) know about Bitcoin's security?

Things to consider:

1. Programming errors
2. Transaction malleability
3. Lack of anonymity
4. Hardware mining
5. The "51% attack"
6. Mining pools
7. Selfish mining
8. Potential threats
9. Problems with key storage

Risk associated to pooled mining

June 2014: the Ghash.io pool got > 50% of the total hashpower.



Then this percentage went down…

Observation

What we were promised:

“distributed currency independent from the central banks”

What we got (in June 2014):

“currency controlled by a single company”…

A problem

Individual miners sometimes do **not** **control** over what they mine.

For example in the **Stratum** protocol (commonly used by mining pools):

miners cannot choose Bitcoin transactions on their own

From mining.bitcoin.cz/stratum-mining:

> *"In my experience **99% of real miners don't care about transaction selection anyway, they just want the highest possible block reward**. At this point they share the same interest with pool operator, so there's no real reason to complicate mining protocol just for those 1% who want to create custom blocks for the pool."*

How to break Bitcoin? – version 2

1. Start a number of mining pools with a **negative fee.**
2. Wait until you get **>50%** of the total hashrate.

Will the miners join?

Another risk

Why not to **rent** the hashpower to perform the attack?

Conjecture

Maybe the only **reason why nobody broke Bitcoin yet** is that nobody was really interested in doing it?

(until recently it was impossible to short Bitcoin)

How to analyze it?

Use a **game-theoretic model**.

See:

[Joseph Bonneau, Edward W. Felten, Steven Goldfeder, Joshua A. Kroll and Arvind Narayanan, *Why buy when you can rent? Bribery attacks on Bitcoin consensus*, 2014]

What we do (not) know about Bitcoin's security?

Things to consider:

1. Programming errors
2. Transaction malleability
3. Lack of anonymity
4. Hardware mining
5. The "51% attack"
6. Mining pools
7. Selfish mining 
8. Potential threats
9. Problems with key storage

It turns out that even a dishonest minority can attack Bitcoin...

Selfish mining

Ittay Eyal, Emin Gun Sirer **Majority is not Enough: Bitcoin Mining is Vulnerable**

Basic idea: when you mine a new block keep it to yourself (also called **block withholding** strategy).

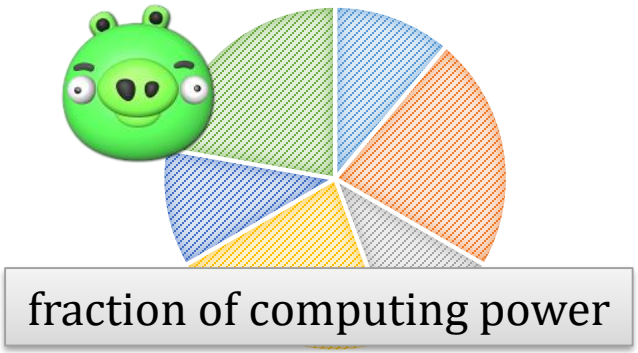
Goal: make the honest miners waste their effort at mining blocks that will never make it to the chain.

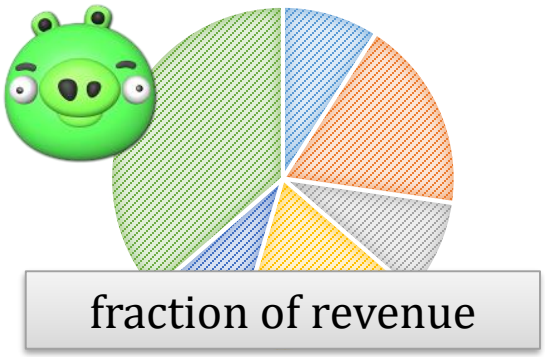
Observe

- the **proportion** of the blocks that you mine will be higher than it should be,
- **hence**: you will earn more than your share of computing power (since Bitcoin adjusts the difficulty)

Why is it bad?

If there is a strategy that is more beneficial than the honest strategy then **miners have an incentive to misbehave** ("*Bitcoin is not incentive compatible*")





(recall that with the honest strategy every miner whose computing power is an α-fraction of the total computing power gets an α-fraction of the revenue)

Moreover: the larger α is the more beneficial this strategy is.

Therefore: the miners have incentives to join a large pool that uses this strategy.

A simplifying assumption (for a moment)

What happens when there is a fork?

Bitcoin specification:

> "from two chains of equal length mine on the first one that you received".

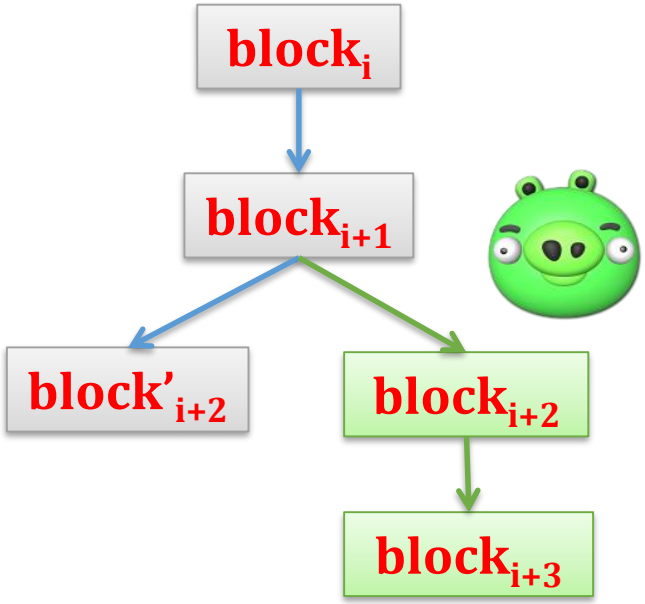
Assume that the adversary is always first (e.g. he puts a lot of "fake nodes" that act as sensors).

An observation

Assume that the adversary does not broadcast the new block that he found (and mines on it "privately").

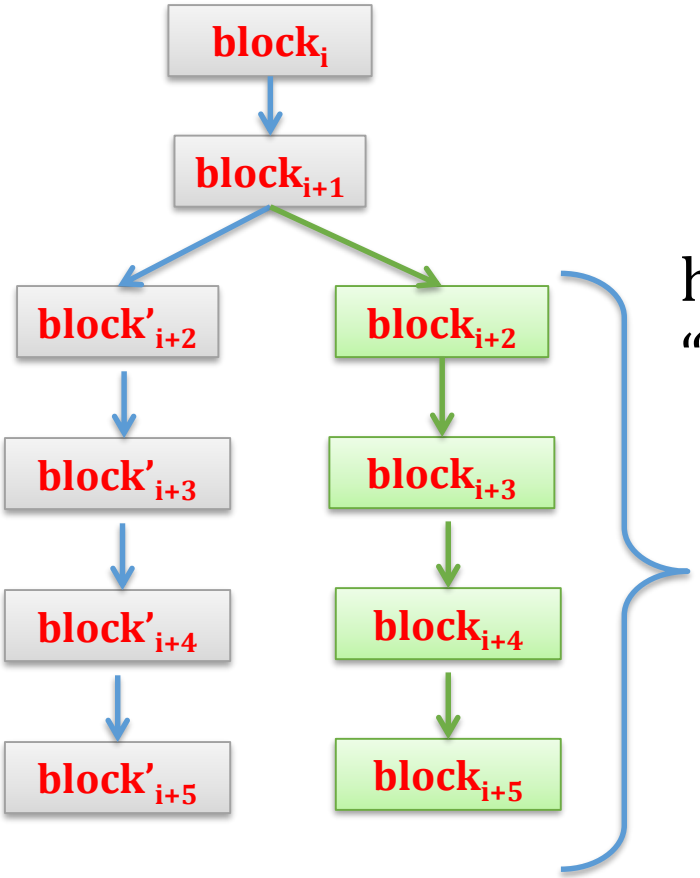
Two things can happen:

1. the **adversary** manages to extend his "private block chain" by one more block, or
2. the "**honest users**" manage to find an alternative extension.



In this case the **adversary** quickly publishes his block so he looses nothing

If the adversary is lucky then he obtains advantage over the honest miners.



he publishes his chain if the "public chain" equalizes with it

the reward for these blocks goes to him

Note: this works even if the adversary has minority of computing power.

Full attack

The assumption that “the adversary is always first” may look unrealistic.

Eyal and Sirer show a modification of this strategy that works **without this assumption**.

- **γ – probability that an honest user chooses adversary’s block**
- **α – fraction of adversary’s computing power**

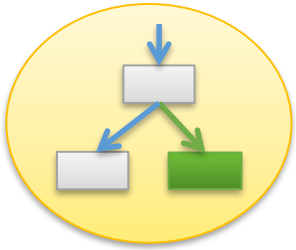
We present it on next slides.

Note

γ – probability that an honest user chooses adversary's block

α – fraction of adversary's computing power

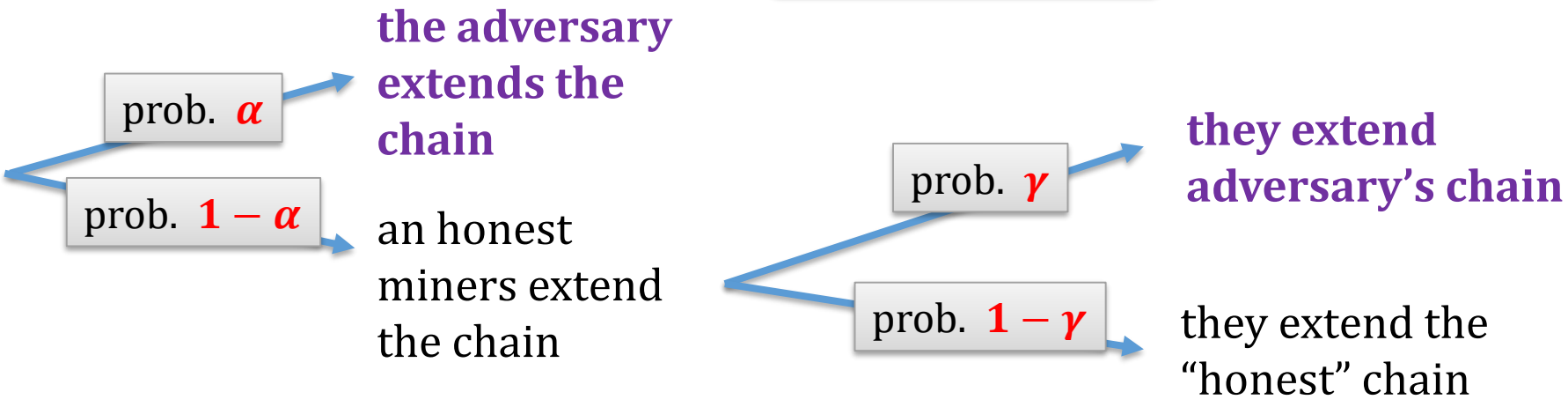
the probability that the adversary wins if there is a fork is equal to



$$\alpha + (1 - \alpha)\gamma$$

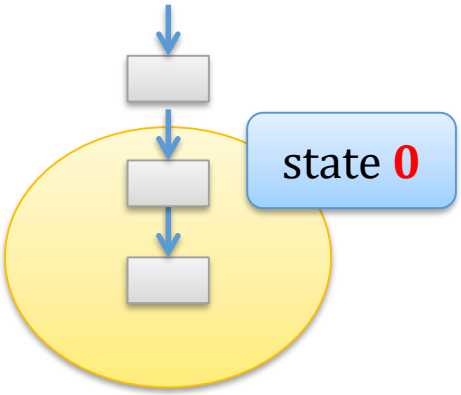
denote it δ

Why?

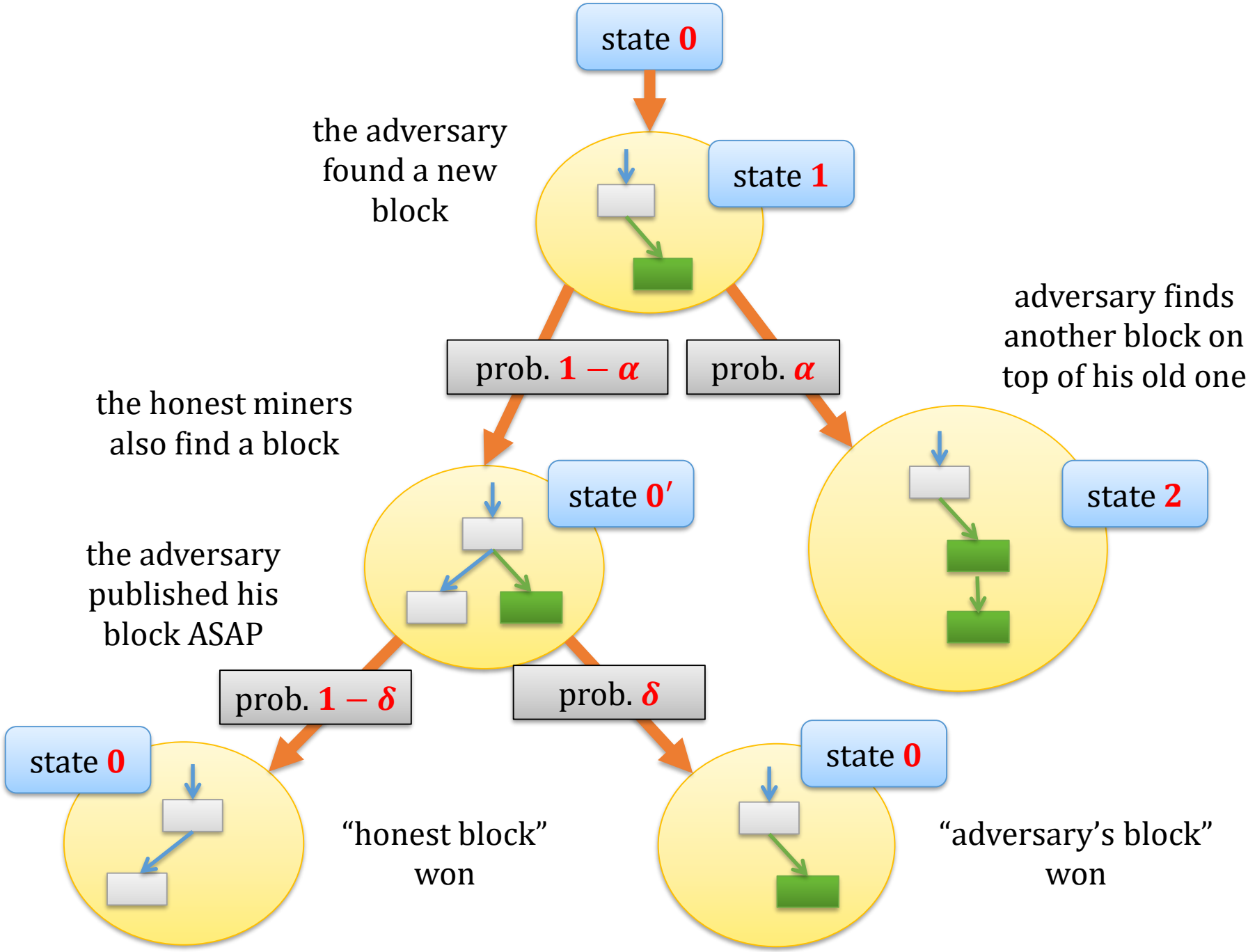


At the beginning of the attack we have:

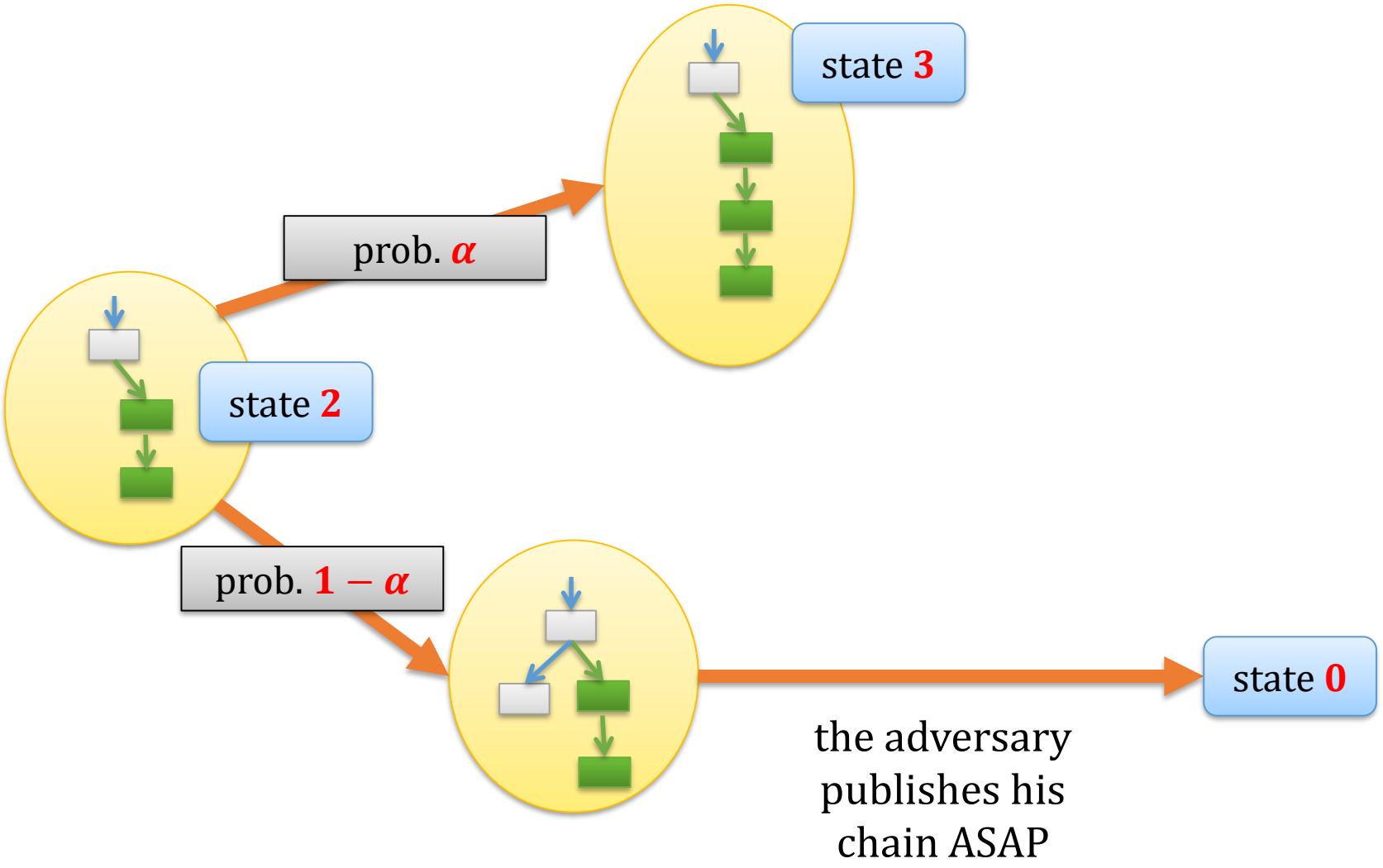
initial state: someone mined a new block and everyone is trying to extend it



First step: if the adversary finds a new block – he keeps it private.


state 0
the adversary found a new block
state 1
prob. 1 − α
prob. α
adversary finds another block on top of his old one
the honest miners also find a block
state 0′
state 2
the adversary published his block ASAP
prob. 1 − δ
prob. δ
state 0
state 0
“honest block” won
“adversary’s block” won

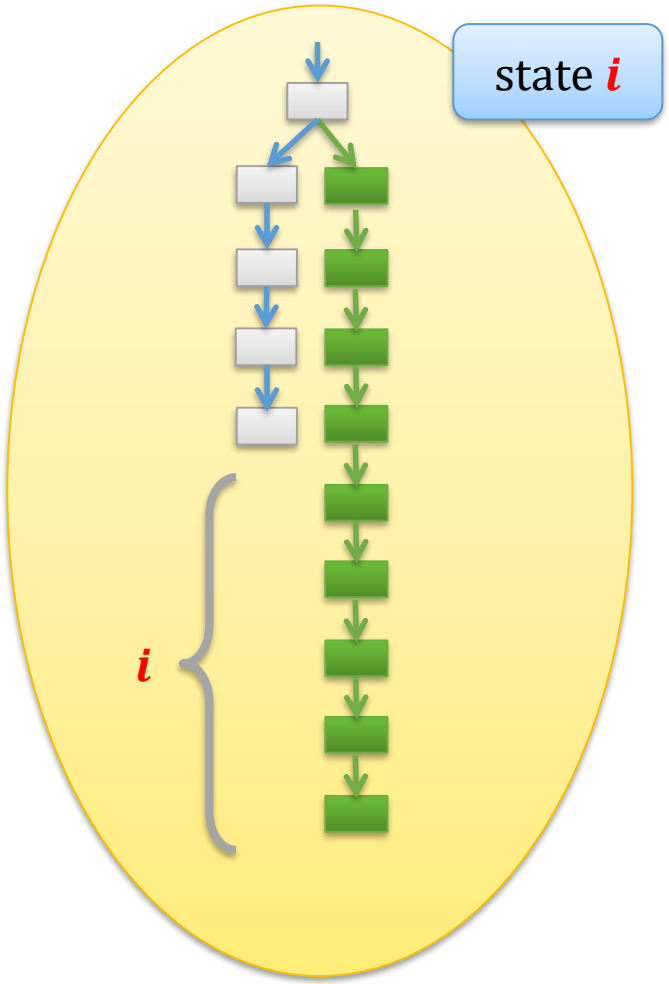
From state 2:


state 3
prob. α
state 2
prob. 1 − α
state 0
the adversary publishes his chain ASAP

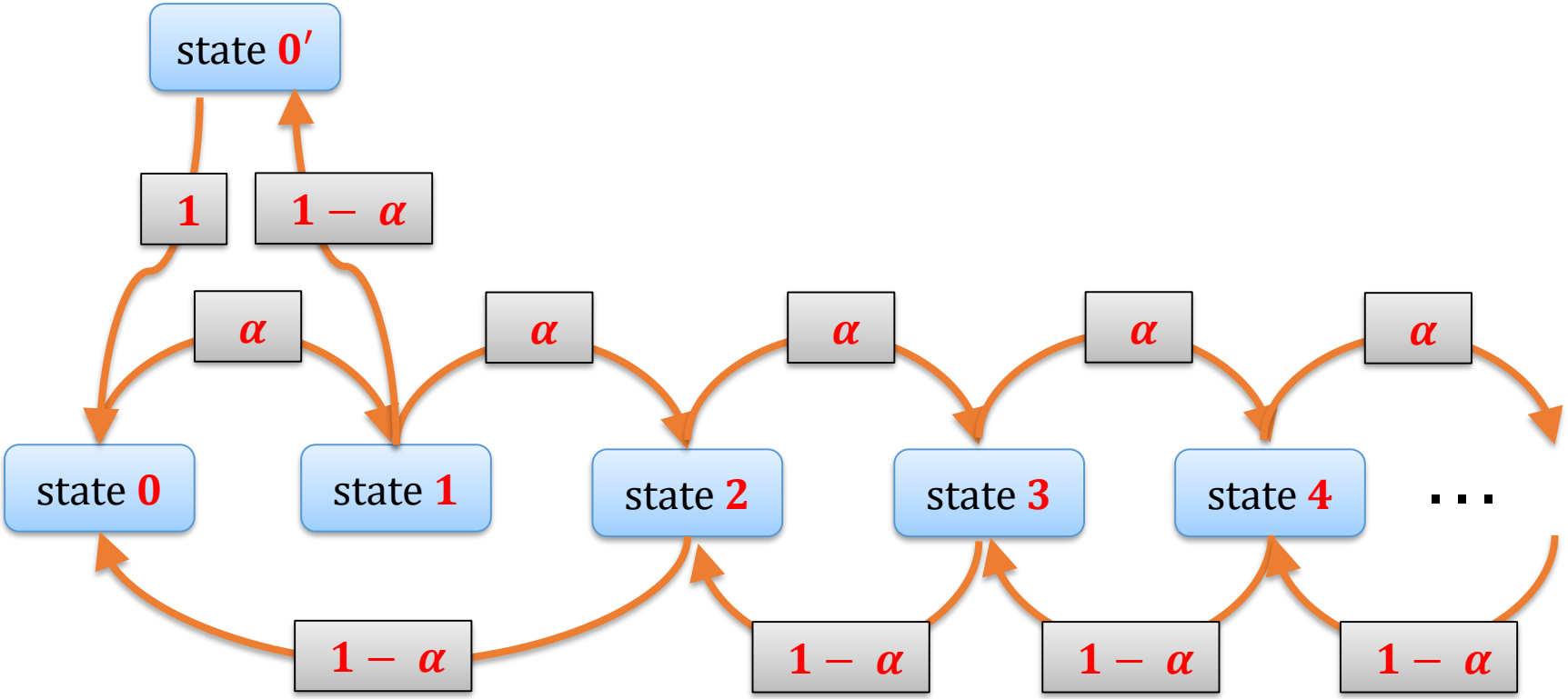
In general for $i \geq 2$

state i:

"the adversary has advantage i over the honest miners"

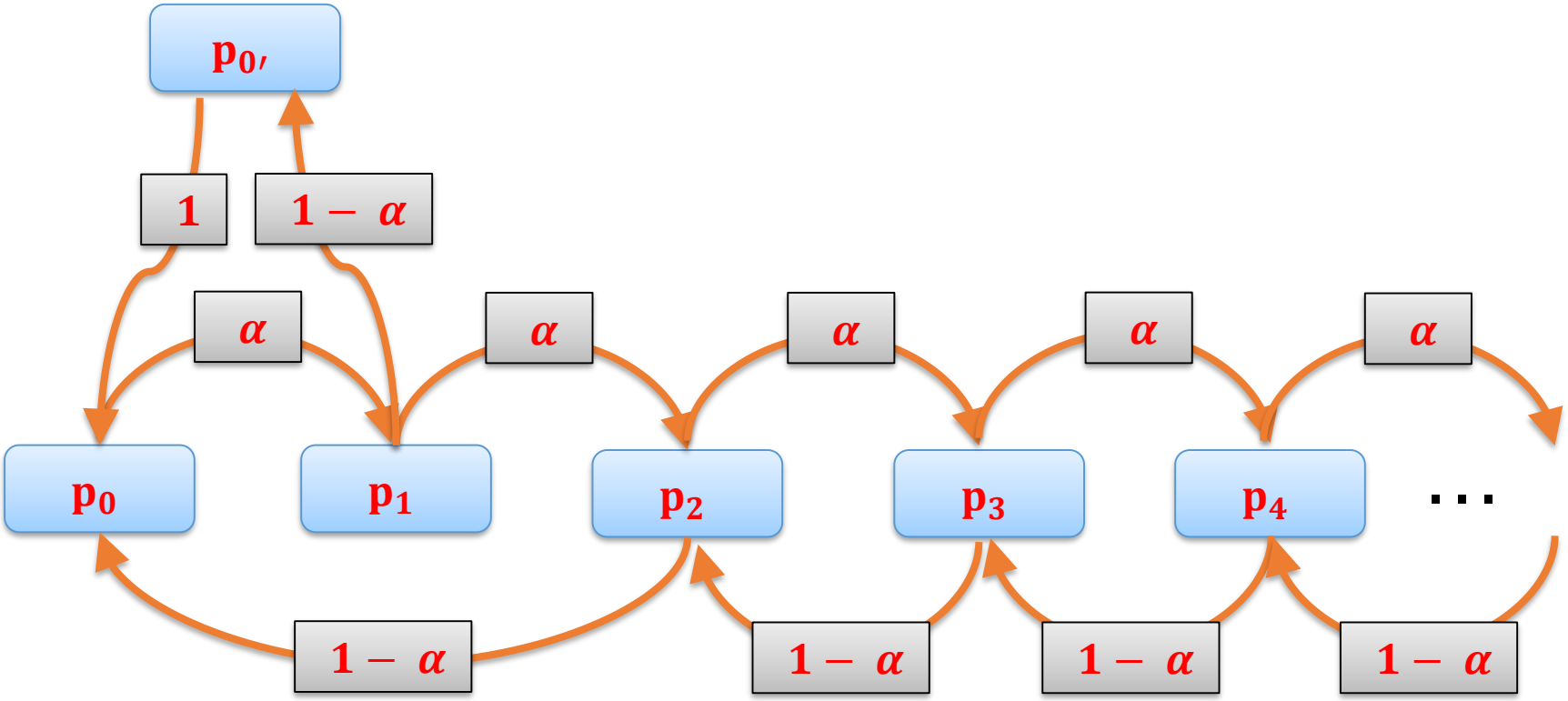


This leads to the following state machine:



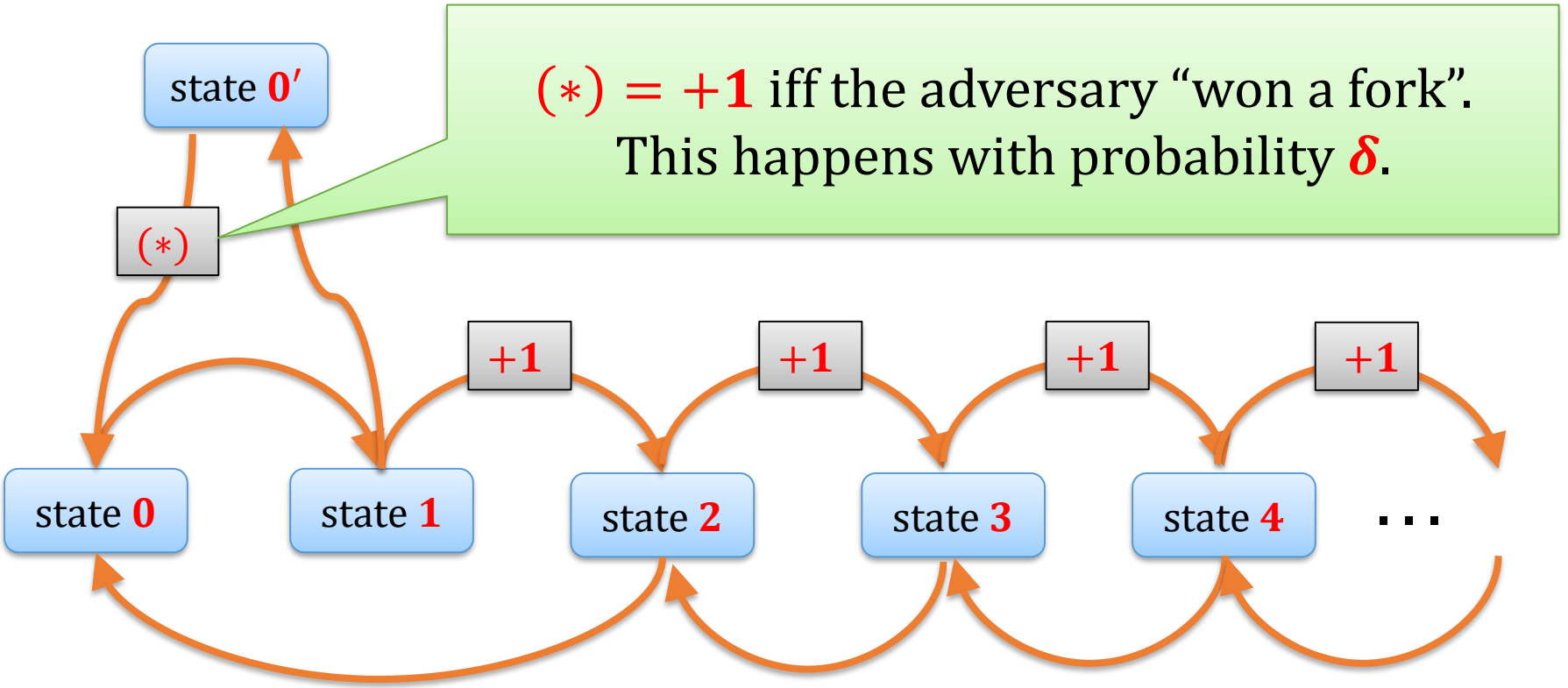
This converges to some stationary distribution $p_0, p_{0'}, p_1, p_2, p_2, \ldots$

We can find it using the theory of Markov chains



How to calculate adversary's revenue?

Look when the adversary "earns a block":

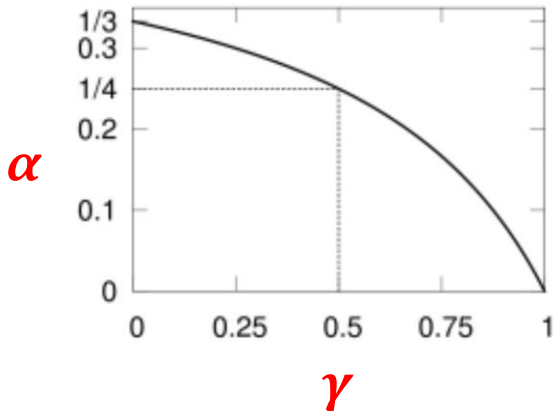


Hence the expected revenue of the adversary is equal to:

$$\delta \cdot p_{0'} + \alpha \cdot p_1 + \alpha \cdot p_2 + \cdots$$

The final result

Eyal and Sirer calculate this, and show that their strategy works as long as $\alpha > \frac{1-\gamma}{3-2\gamma}$



They also show that the larger α is the more beneficial this strategy is.

How to fix it?

One simple idea to make $\boldsymbol{\gamma = \frac{1}{2}}$:

Instruct the miners to mine on a **random chain**
(in case they receive to equal ones)

Another clever attack

Lear Bahack **Theoretical Bitcoin Attacks with less than Half of the Computational Power**

The "*Difficulty Raising Attack*" – exploits the way the difficulty is adjusted in Bitcoin.

What we do (not) know about Bitcoin's security?

Things to consider:

1. Programming errors
2. Transaction malleability
3. Lack of anonymity
4. Hardware mining
5. The "51% attack"
6. Mining pools
7. Selfish mining
8. Potential threats
9. Problems with key storage

Blocks without transactions

Example:

Block 310084

Short link: http://blockexplorer.com/b/310084

Hash: 000000000000000001179ab36c2c029dea89d6c5506042414f6d5acddc46c5c86
Previous block: 0000000000000002c8d1f61ef0a3d4c0bef36ccce8d54b938d01a7bba35e0d9
Next block: 00000000000000000351a0599912172d7e908f6fac551c28eba2dc6105347aedf
Time: 2014-07-10 12:39:45
Difficulty: 16 818 461 371.161111 ("Bits": 18415fd1)
Transactions: 1
Total BTC: 25
Size: 252 bytes
Merkle root: 660c7c9af8d8fec401b1076f1bff31afec79843ded9d920b8dae8bc19e0adea1
Nonce: 1454503193
Raw block

Transactions

Transaction	Fee	Size (kB)	From (amount)	To (amount)
660c7c9af8...	0	0.171	Generation: 25 + 0 total fees	1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY: 25

Reason: shorter blocks propagate faster.

In the future the opposite problem can happen

When the mining reward becomes negligible, we can experience:

Tragedy of the commons:

adding a transaction costs nothing, so the miners will not be able to keep the transaction fees high.

Another question

Verification of blocks takes time.

Recall that verification includes checking all transactions

Maybe it's cheaper not to verify?

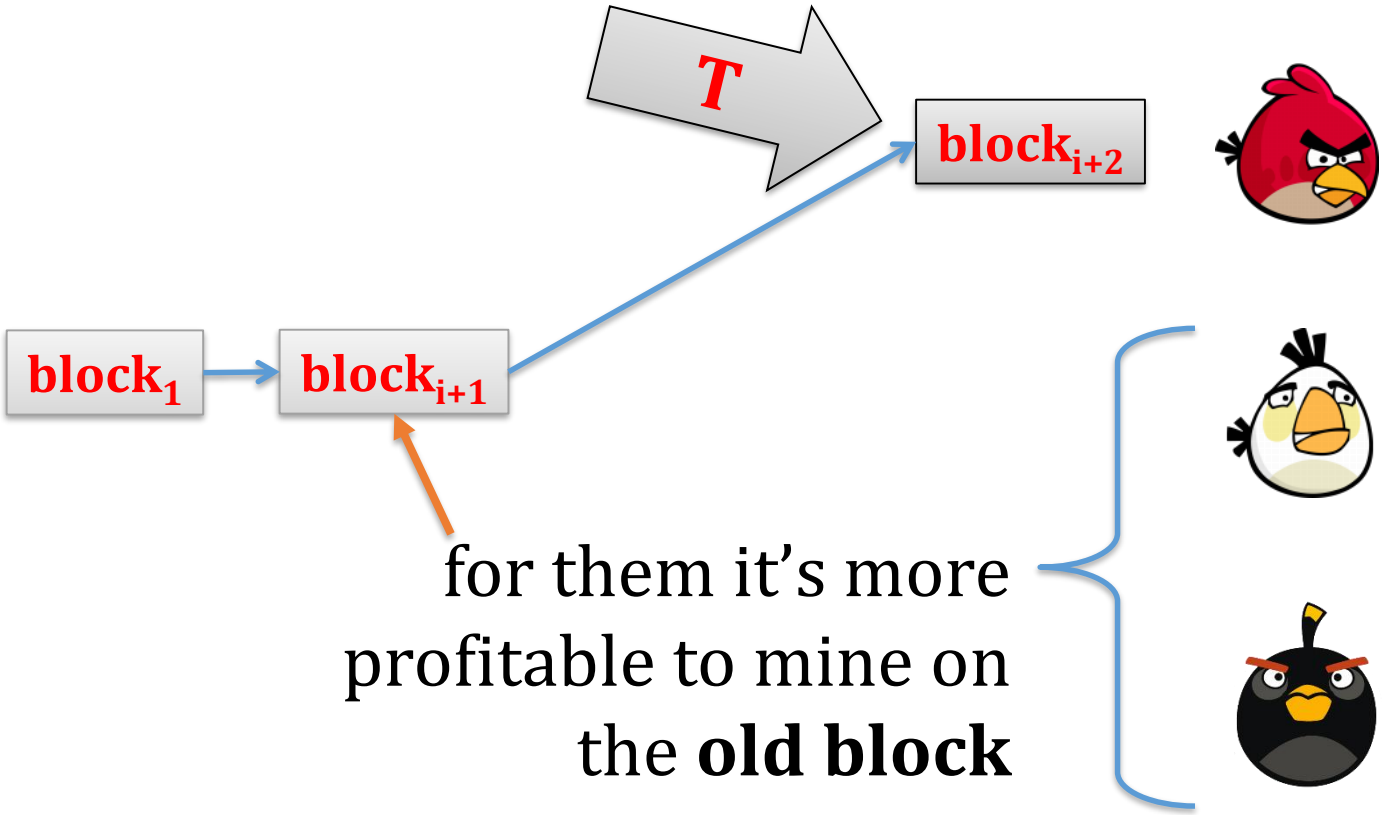
("verifier's dilemma")

(more relevant to Ethereum)

See **[Luu, Teutsch, Kulkarni, Saxena, Demystifying incentives in the consensus computer, ACM CCS 2015]**.

Yet another question

What happens if someone posts a transaction **T** with a very high fee (say **100 BTC**)?



Quantum attacks

Q: If quantum computers are built – can then "break blockchain"?

A: Yes, because ECDSA signature scheme is vulnerable to quantum attacks

Solutions:

- according to **cryptographers**: use **post-quantum signatures**
- according to **physicist**: construct a "**quantum blockchain**"

Recent article in “Nature”

“Yet, within ten years, quantum computers will be able to calculate the one-way functions, including blockchains, that are used to secure the Internet and financial transactions. Widely deployed one-way encryption will instantly become obsolete.”

What we do (not) know about Bitcoin's security?

Things to consider:

1. Programming errors
2. Transaction malleability
3. Lack of anonymity
4. Hardware mining
5. The "51% attack"
6. Mining pools
7. Selfish mining
8. Potential threats
9. Problems with key storage

A practical problem: How to store the bitcoins?

- storing in plaintext on the PC – **bad idea** (malware attacks)
- encrypting with a **password – susceptible to the dictionary attacks**
- better: **split the key** between several devices. Two options:
  - use the "**multisignature feature** of Bitcoin
  - use **secret sharing and the MPCs**
- store on the **USB memory** – also susceptible to **malware** (once connected to the PC).
- use a **smarter device** – more secure, especially if it has a display: