

Based on slides from:

Introduction to Cryptocurrencies

a tutorial

Stefan Dziembowski
University of Warsaw



With some added and
removed material





An extended abstract of this tutorial (including the references) is available at: www.crypto.edu.pl/Dziembowski/talks/bitcointutorial.pdf.

These slides are available at www.crypto.edu.pl/Dziembowski/talks.

Outline

1. Introduction to Bitcoin
2. Bitcoin mining pools
3. Security of Bitcoin
4. Smart contracts
5. Other cryptocurrencies
6. Conclusion

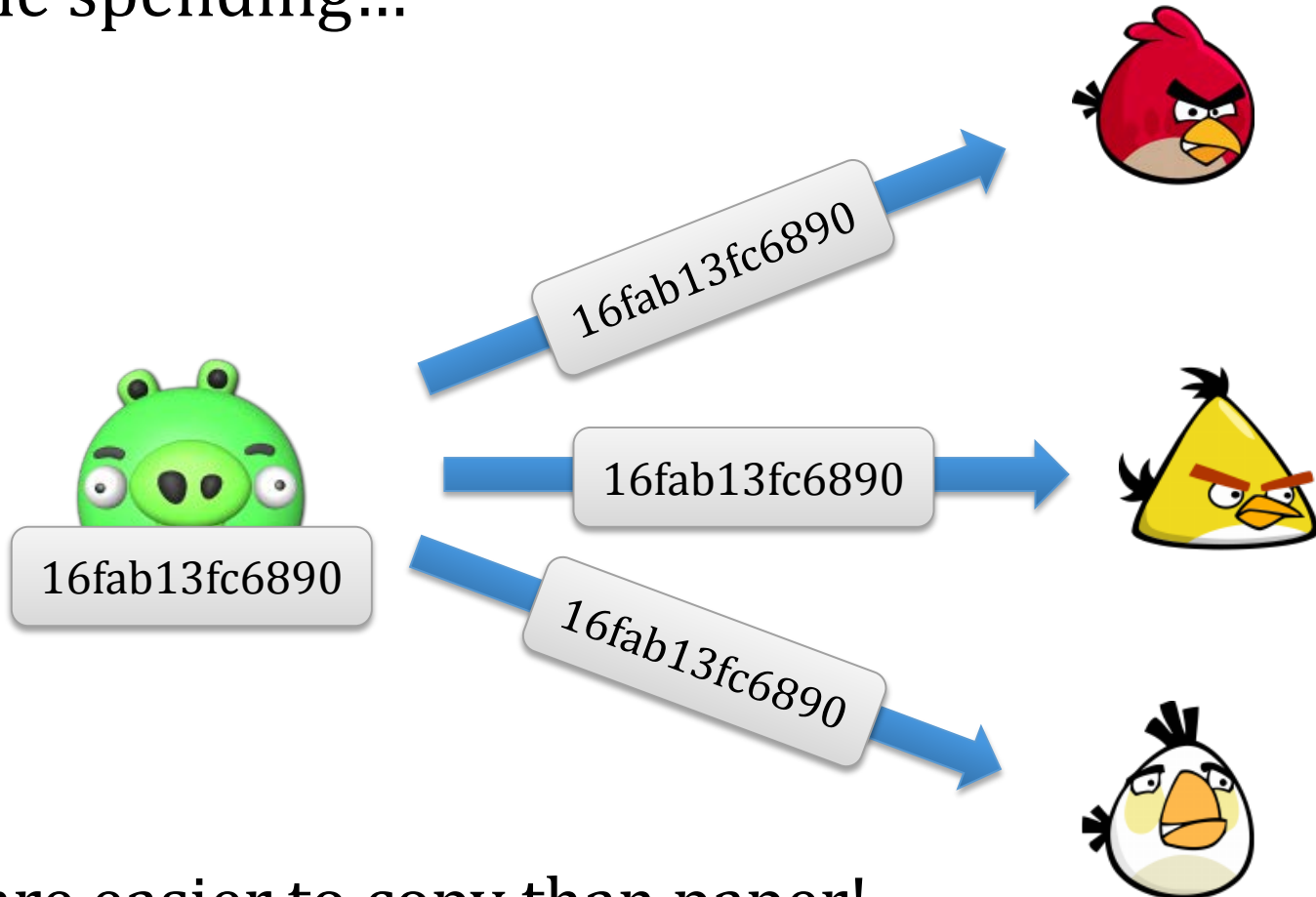
Part I

Introduction to Bitcoin

Main design principles

Main problem with the digital money

Double spending...



Bits are easier to copy than paper!

Bitcoin idea (simplified):

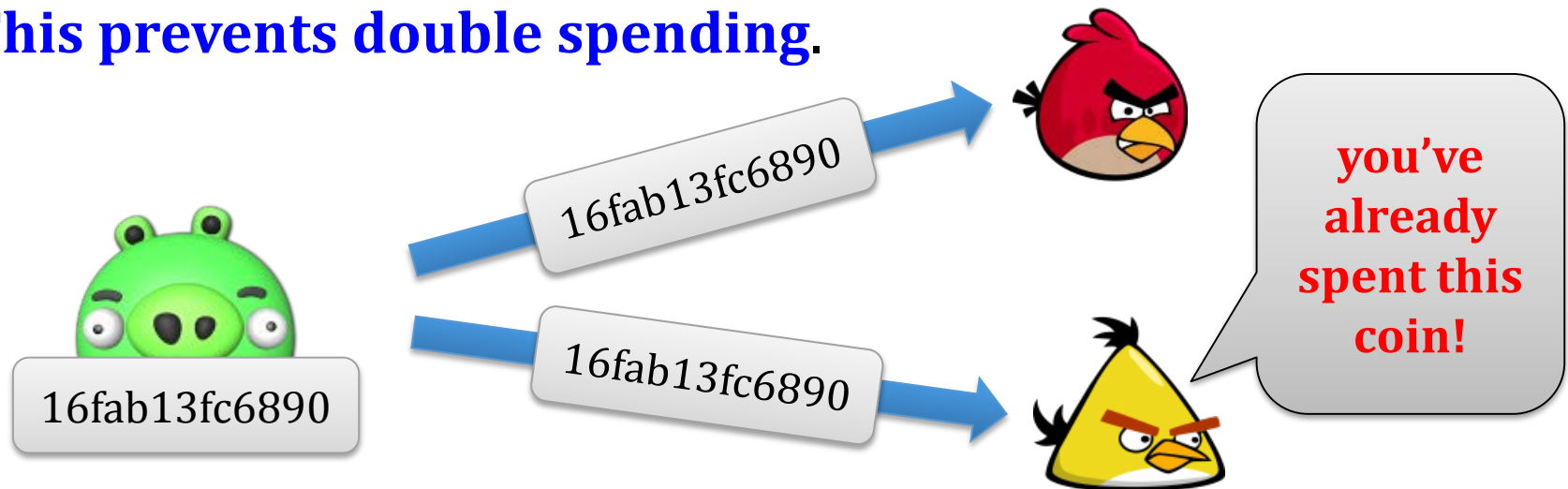
The users emulate a **public trusted bulletin-board** containing a list of transactions.

A transaction is of a form:



“User P_1 transfers a coin #16fab13fc6890 to user P_2 ”

This prevents double spending.



What needs to be discussed

1. How is the **trusted bulletin-board** maintained?
2. How are the users identified?
3. Where does the money come from?
4. What is the syntax of the transactions?



The Merkle-Damgård Transform = Blockchain

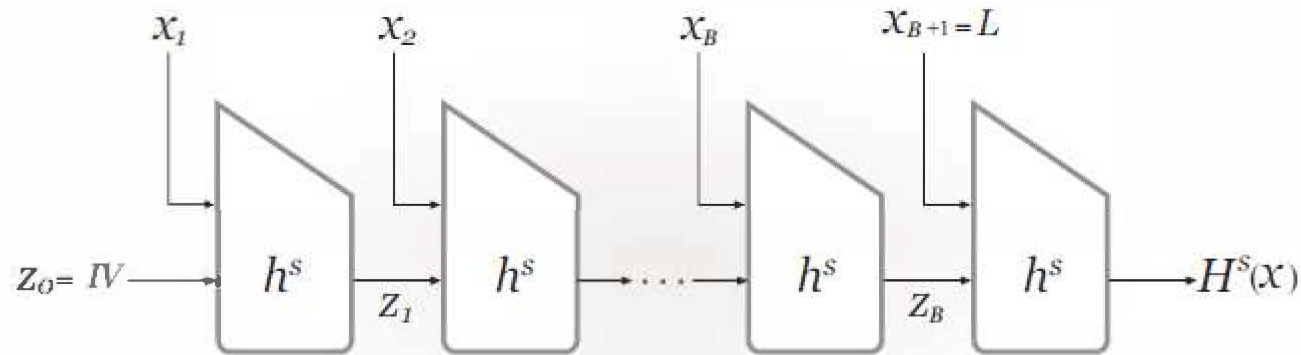


FIGURE 5.1: The Merkle-Damgård transform.

Problem: How to reach consensus on the correct final Z which fixes the **entire history**?

First attempt: Majority vote!

Problem

How to define “**majority**” in
a situation where
everybody can join the network?



The Bitcoin solution

Define the “majority” as

the majority of the computing power

Now creating multiple identities does not help!



How is this enforced?

Main idea:

- use **Proofs of Work**
- **incentivize** honest users to constantly participate in the process

The honest users can use their **idle CPU cycles**.

Nowadays: often done on **dedicated hardware**.

Proofs of work

Introduced by **Dwork and Naor** [Crypto 1992] as a countermeasure against spam.



Basic idea:

Force users to do some computational work:

 solve a **moderately difficult** “puzzle”

 (checking correctness of the solution has to be fast)

A simple hash-based PoW

H -- a hash function whose computation takes time **TIME(H)**



Prover

finds **s** such that **H(s,x)** starts with **n** zeros (in binary)

salt

“hardness parameter”



Verifier

checks if **H(s,x)** starts with **n** zeros

takes time **$2^n \cdot \text{TIME(H)}$**

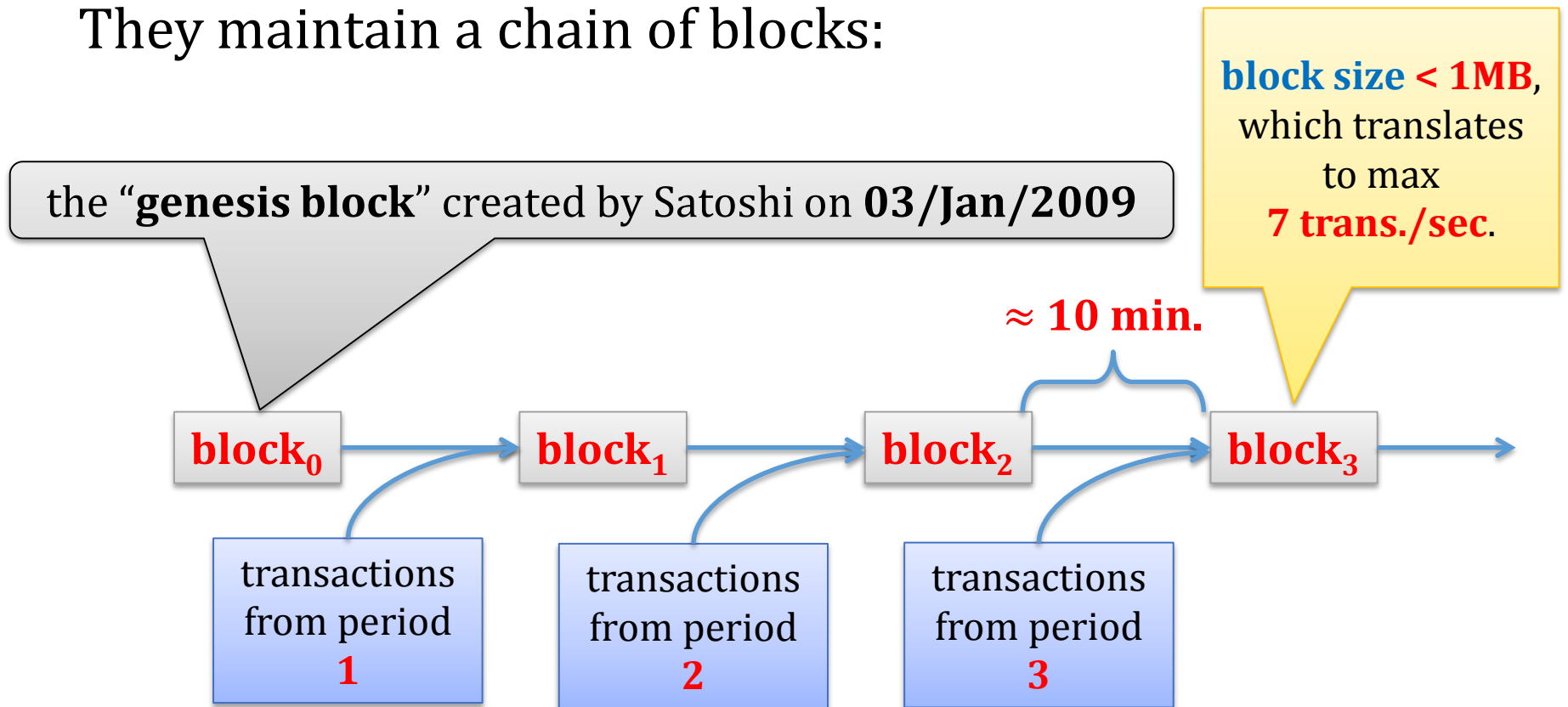
takes time **TIME(H)**

Main idea

The users participating in the scheme are called the “miners”.

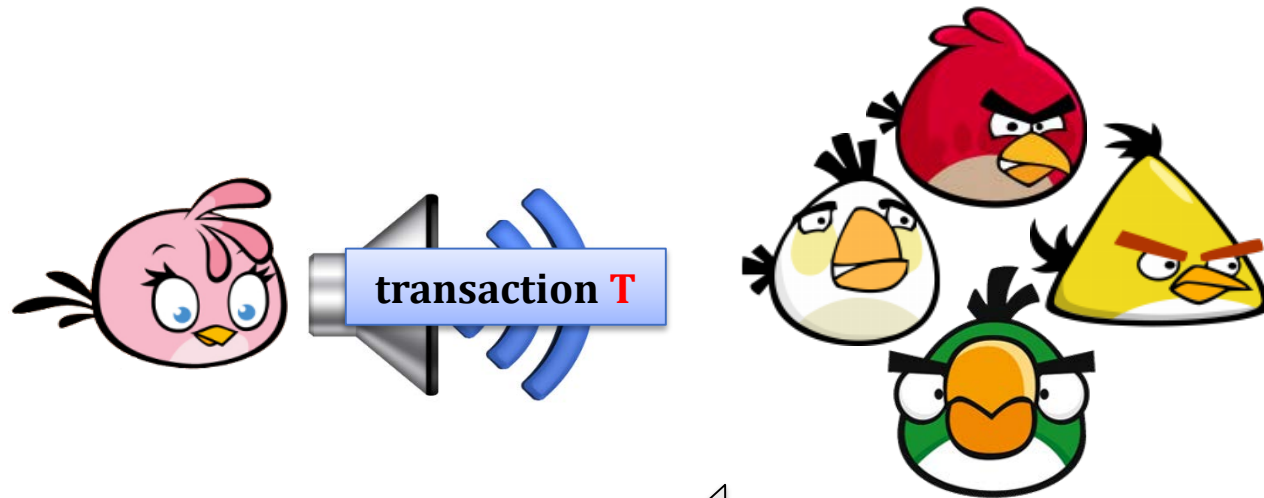


They maintain a chain of blocks:



How to post on the board

Just broadcast (over the internet) your transaction to the miners.



And hope they will **add it to the next block**.

the miners are incentivized to do it.


Important:

They **never add an invalid transaction** (e.g. double spending)

a chain with an invalid transaction is **itself not valid**, so no rational miner would do it.

Main principles

1. It is **computationally hard** to extend the chain.
2. Once a miner finds an extension he **broadcasts it to everybody**.
3. The users will always accept “**the longest chain**” as the valid one.

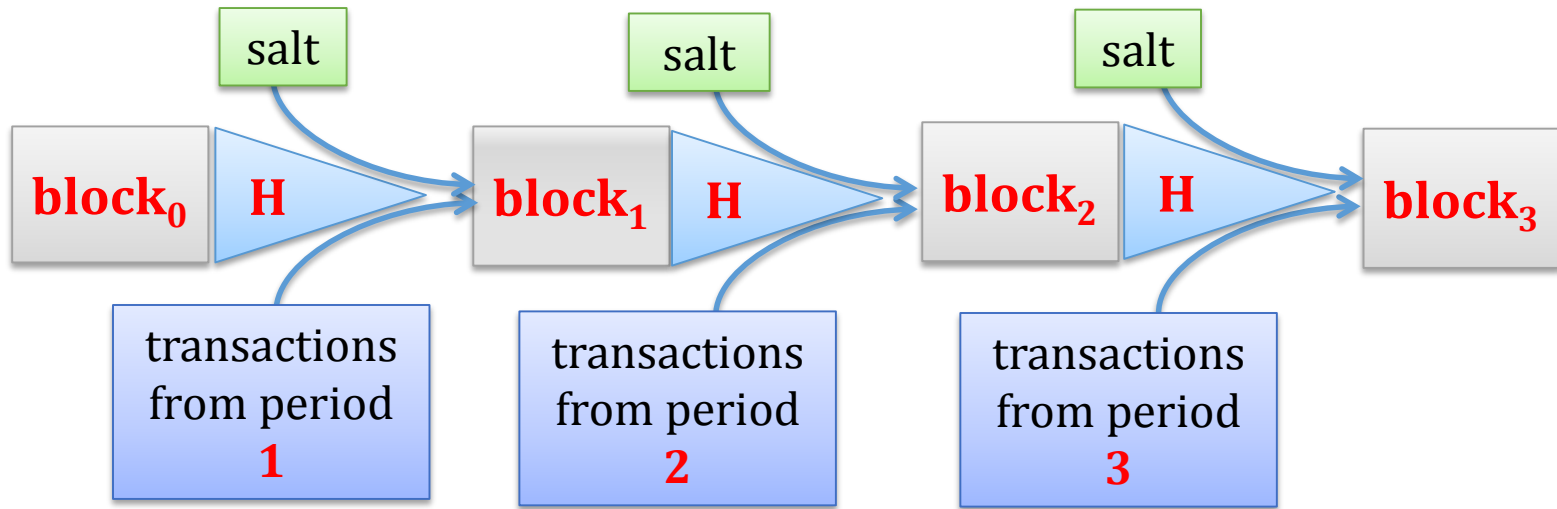


the system
incentivizes
them to do it

How are the PoWs used?

H – hash function

more concretely in Bitcoin: **H** is **SHA256**.



Main idea: to extend the chain one needs to find **salt** such that

$H(\text{salt}, H(\text{block}_i), \text{transactions})$ starts with some number **n** of zeros

“hardness parameter”

The hardness parameter is periodically changed

- The computing power of the miners **changes**.
- The miners should generate the new block **each 10 minutes** (on average).
- Therefore the hardness parameter **is periodically adjusted** to the mining power
- This happens once each **2016 blocks**.
- **Important**: the hardness adjustment is **automatic**, and depends on how much time it took to generate last 2016 blocks.

this is possible since every block contains a **time-stamp** produced by the miner who mined it



“Hashrate” = number of hashes computed per second

total hashrate over the last 2 years:



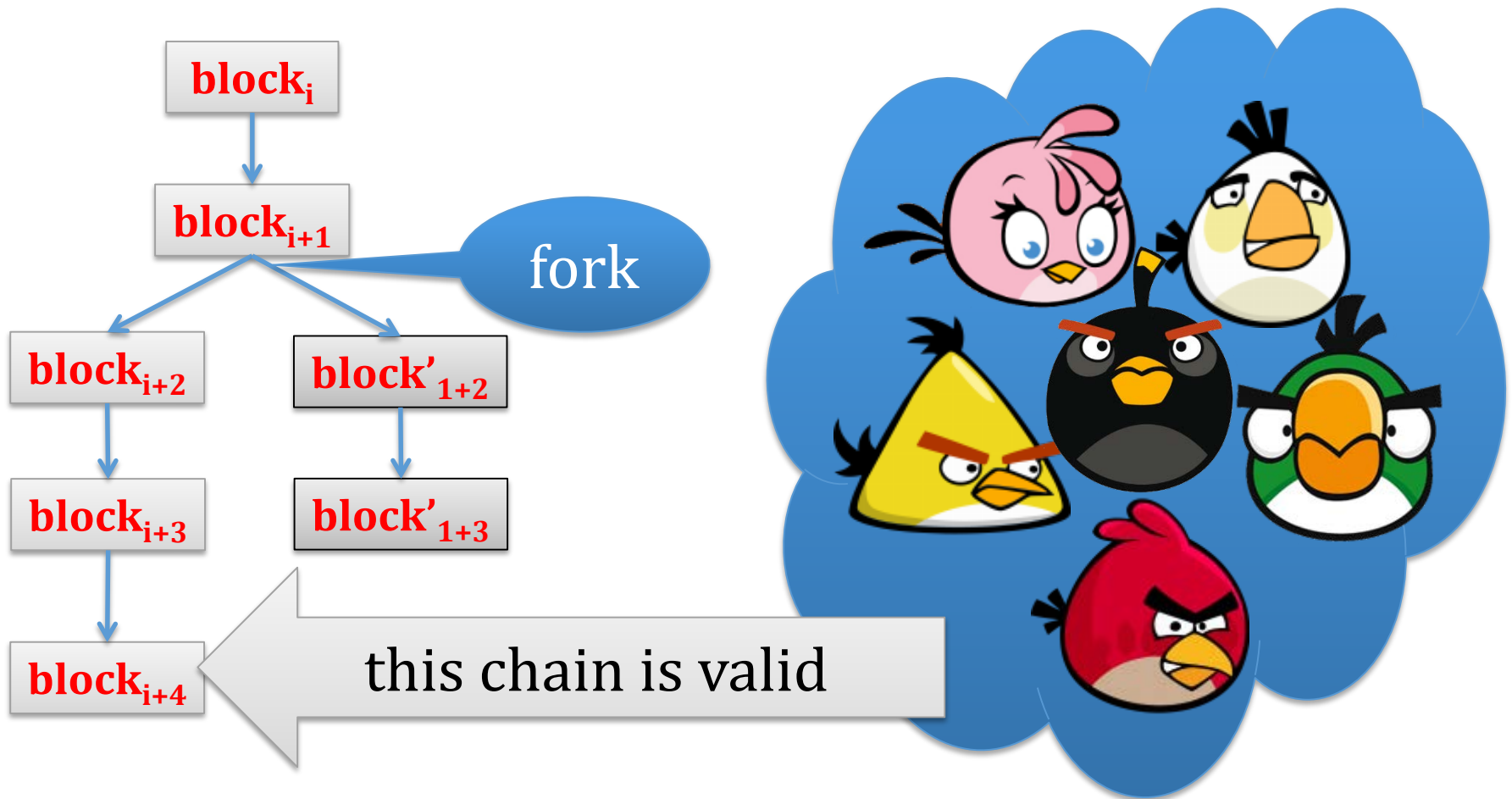
Note:

Sep 17 2013 : 990,986 GH/s
Sep 17 2014 : 280,257,530 GH/s
Sep 17 2015 : 385,067,688 GH/s

≈ 2⁵⁸ hash / second

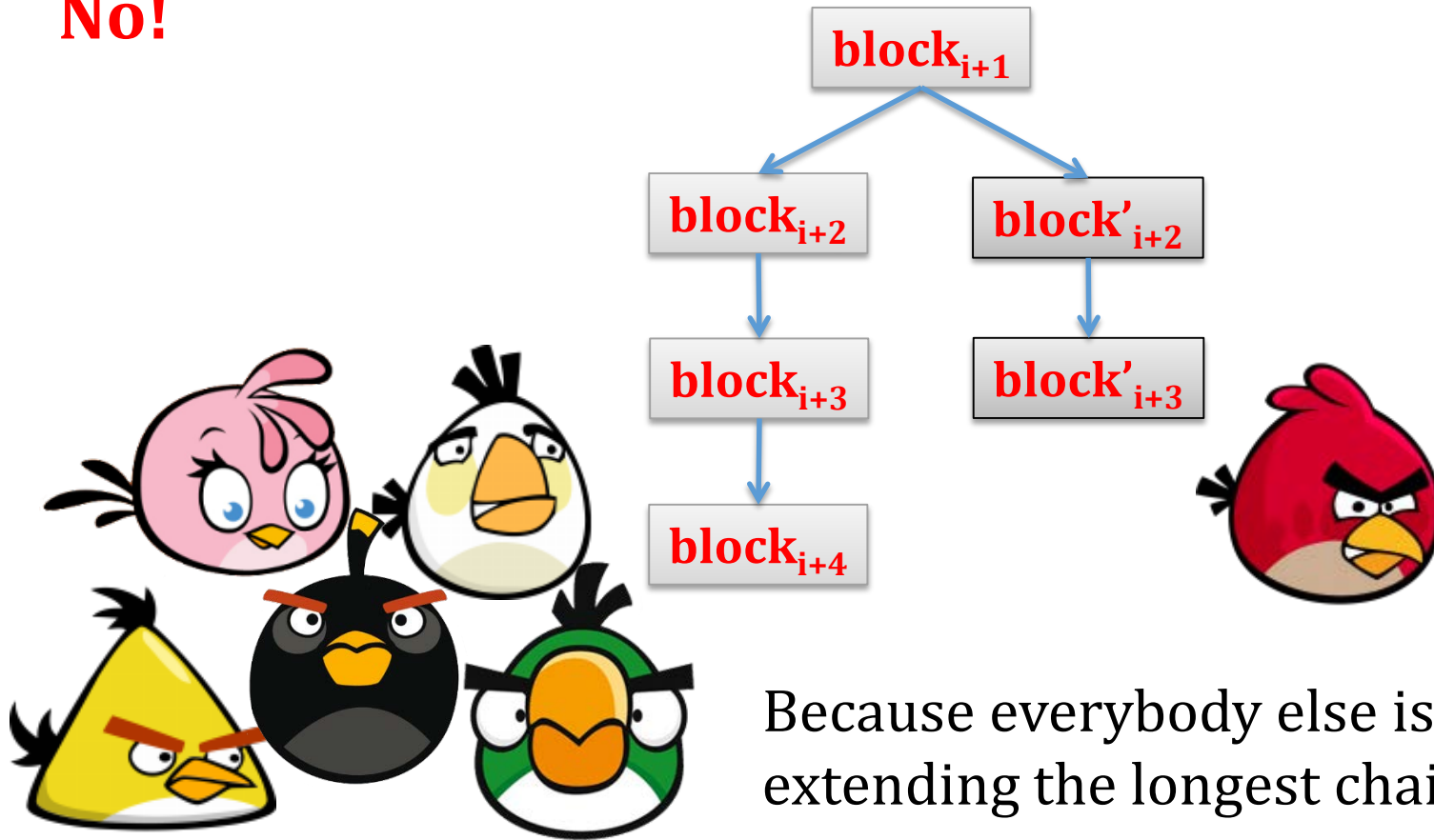
What if there is a “fork”?

For a moment let's say: the “**longest**” chain counts.



Does it make sense to “work” on a shorter chain?

No!

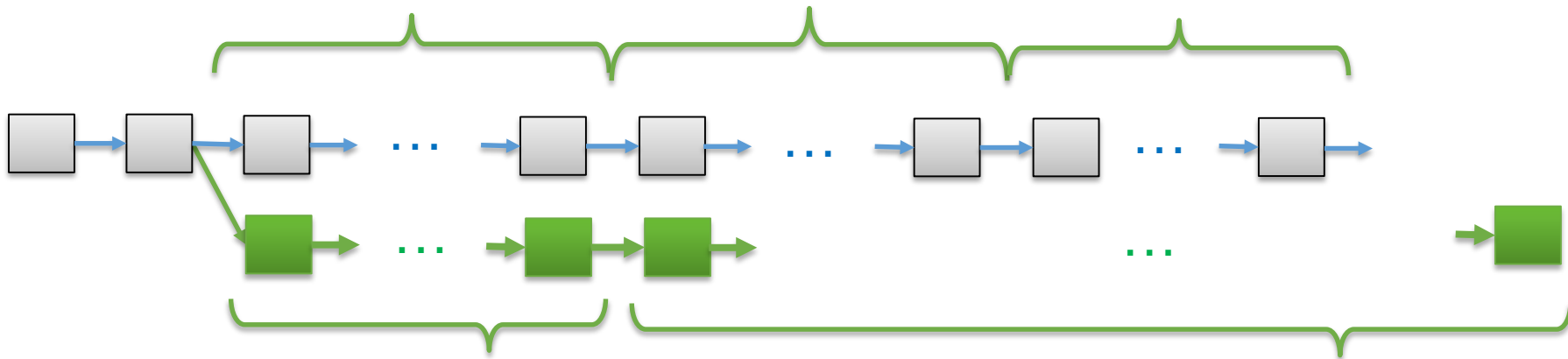


Because everybody else is working on extending the longest chain.

Recall: we assumed that the majority follows the protocol.

Since hardness is adjusted thus the following attack might be possible

the “2016 blocks” periods



the adversary
forks the chain:



(1)
he computes
(secretly) another
chain with **fake time-
stamps** (indicating
that it took zero time
to produce it)



(2)
the difficulty drops
dramatically, so he
can quickly produce
a chain longer than
the valid one, and
publish it.

Therefore

In Bitcoin it's not the **longest chain** but the **strongest chain** that matters.

The **strength of each block** is 2^n .

n – the hardness parameter in a given period

The **strength of the chain** is the sum of hardnesses of each block in it.

How are the miners incentivized to participate in this game?

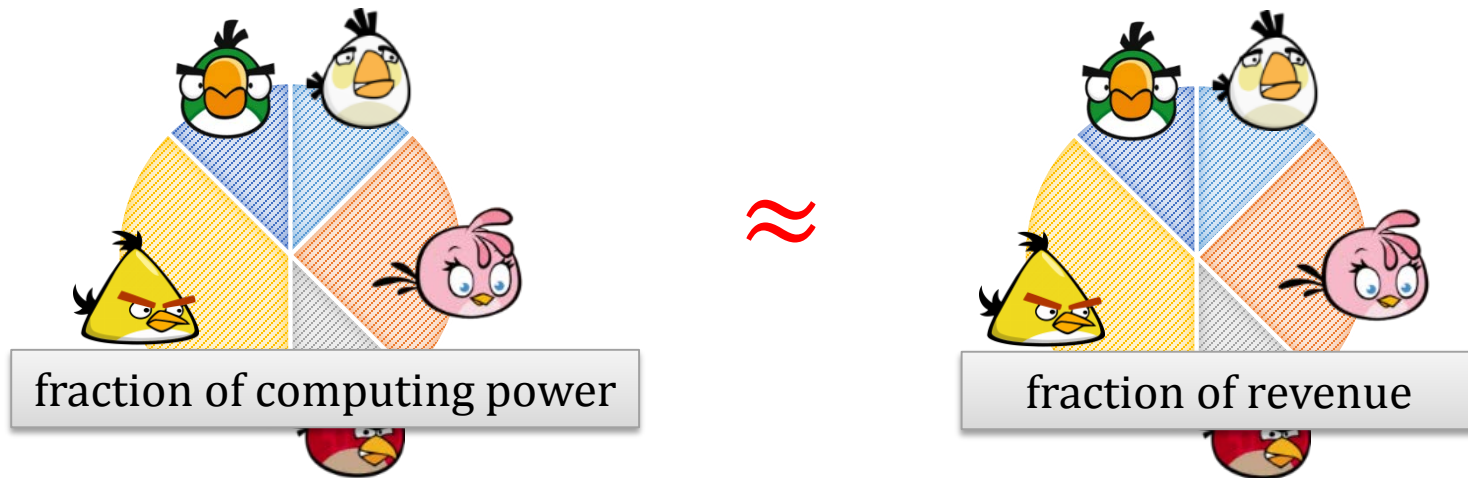
Short answer: they are paid (in Bitcoins) for this.
We will discuss it in detail later...



An important feature

Suppose everybody behaves according to the protocol
then:

every miner P_i whose **computing power is an α_i -fraction** of the total computing power **mines an α_i -fraction of the blocks**.



Intuitively this is because:

P_i 's chances of winning are **proportional to**
the number of times P_i can compute H in a given time frame.

What is needed to decide which blockchain is valid?

In theory: one needs to know **only**:

- the **initial rules of the game**
- the **genesis block B_0**

This can take several hours.

Note: as of **Oct 13, 2015**:
blockchain's size is **$\approx 45\text{MB}$** .

Then from many “candidate chains” choose the one that

- **verifies correctly** (starts **B_0** and is satisfies all the rules)
- is **the strongest**.

One doesn't even need to have access to the communication history.

In practice: it's not that simple...

we will talk about it in a moment

Freshness of the genesis block



I didn't know the genesis block before Bitcoin was launched (**Jan 3, 2009**)

Here is a heuristic “proof”:

Block₀ contained a hash of a title from a front page of the London Times on **Jan 3, 2009**

Chancellor on brink of
second bailout for banks

A recent paper that shows how to generate the genesis block in a distributed way: **[Andrychowicz, D., CRYPTO'15]**.

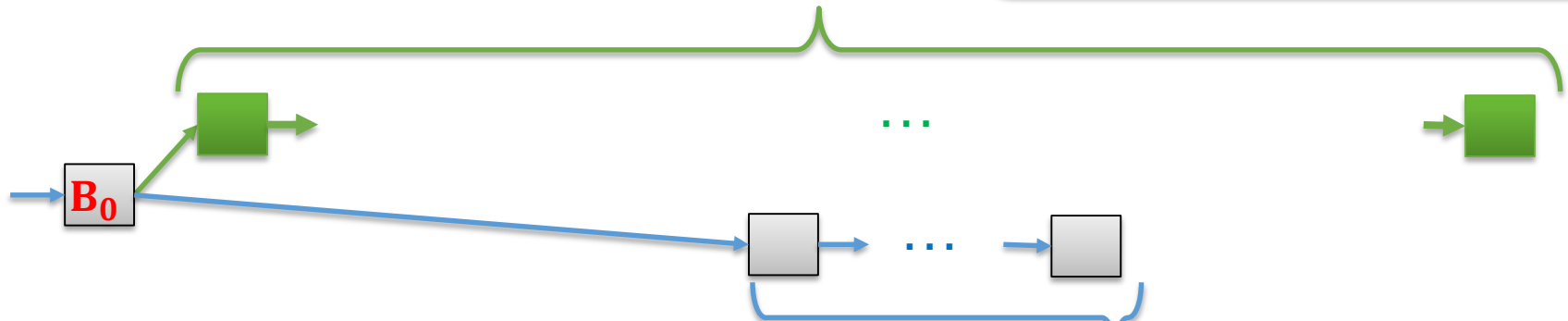
Why is this important?

Otherwise Satoshi could “pre-mine”
(mostly a theoretical threat today):



(1)
secretly start mining in
1980, produce a very
strong chain

(3)
On **Jan 03, 2010** publish
your secret chain



(2)
honest miners start working on **Jan 03, 2009**.
They have **more computing power** but **less time**.
So, **after 1 year** their chain is still **weaker** than the
one of Satoshi.



Checkpoints

Checkpoint – old block hash **hardcoded into Bitcoin software**.

From the **theoretical** point of view: ***not needed***.

Moreover: they go against the “decentralized” spirit of Bitcoin.

Still they have some **practical advantages**:

- they prevent some **DoS attacks** (flooding nodes with unusable chains)
- they prevent attacks involving **isolating nodes** and giving them fake chains,
- they can be viewed as an **optimization** for the initial blockchain download.

Protocol updates

The Bitcoin protocol **can be updated**.

Proposals for the Bitcoin updates can be submitted to the **Bitcoin foundation** in the form of the **Bitcoin Improvement Proposals (BIPs)**.


Then the foundation puts them at vote.

Only the miners can vote. The votes are included in the mined blocks.

Currently it is required that a proposal gets a **75% approval in the mined blocks** (over some period of time).

Note: **75% of blocks \approx 75% of computing power**.

What needs to be discussed

1. How is the **trusted bulletin-board** maintained?
2. How are the users identified? 
3. Where does the money come from?
4. What is the syntax of the transactions?

User identification

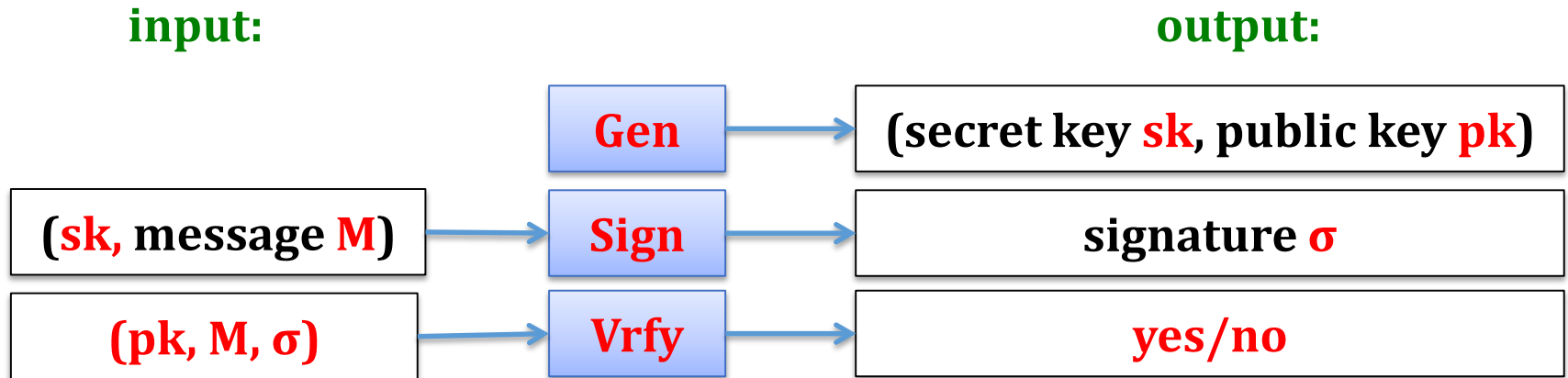
We use the digital signature schemes.



The users are identified by their public keys.

Digital signature schemes

A **digital signature scheme** consists of algorithms **Gen**, **Sign** and **Vrfy**, where:



Correctness:

for every $(sk, pk) := \text{Gen}()$ and every M we have
 $\text{Vrfy}(pk, M, \text{Sign}(sk, M)) = \text{yes}$

Security:

“without knowing sk it is infeasible to compute σ such that
 $\text{Vrfy}(pk, M, \sigma) = \text{yes}$ ”

What needs to be discussed

1. How is the **trusted bulletin-board** maintained?
2. How are the users identified?
3. Where does the money come from?
4. What is the syntax of the transactions?



Where does the money come from?

A miner who finds a new block gets a “reward” in **BTC**:

≈ 4 years

- for the first **210,000** blocks: **50 BTC**
 - for the next **210,000** blocks: **25 BTC**
 - for the next **210,000** blocks: **12.5 BTC**,
- and so on...



current reward

Note: $210,000 \cdot (50 + 25 + 12.5 + \dots) \rightarrow 21,000,000$

More details

Each block contains a transaction that **transfers the reward** to the miner.

Advantages:

1. It provides **incentives** to be a miner.
2. It also makes the miners interested in **broadcasting new block** asap.

this view was challenged in a recent paper:

Ittay Eyal, Emin Gun Sirer

Majority is not Enough: Bitcoin Mining is Vulnerable

(we will discuss it later)

What needs to be discussed

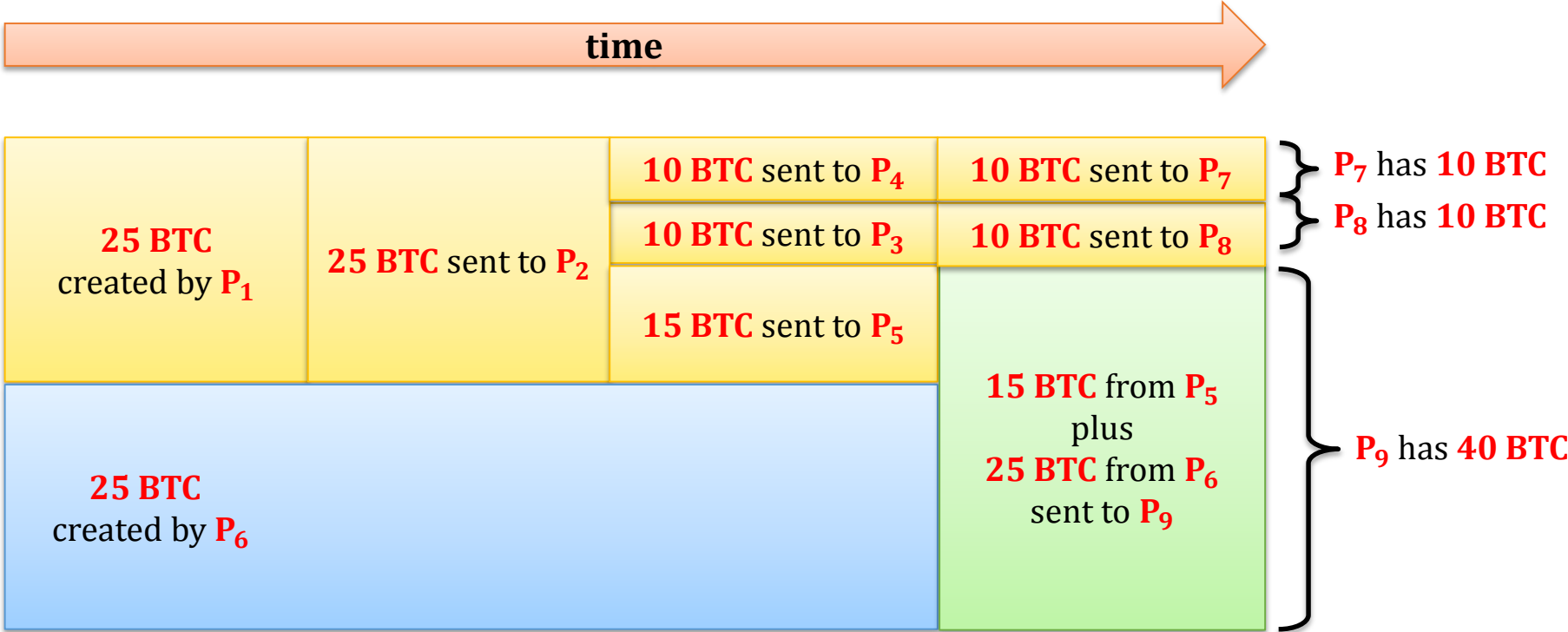
1. How is the **trusted bulletin-board** maintained?
2. How are the users identified?
3. Where does the money come from?
4. What is the syntax of the transactions?



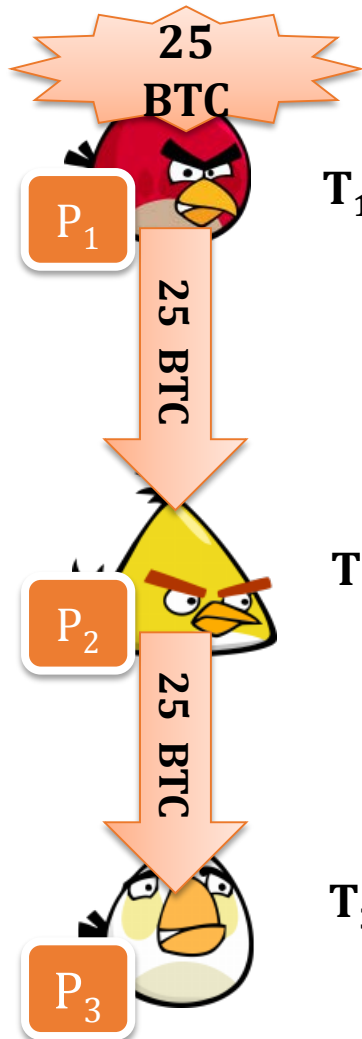
Bitcoin's money mechanics

Bitcoin is “transaction based”.

Technically: there is no notion of a “coin” in Bitcoin.



Transaction syntax – simplified view



in the “mining process”

We say that **T₃**
redeems **T₂**

T₁ =

(User P₁ creates 25 BTC)

[T₂]

“value of T₂”

T₂ =

(User P₁ sends 25 BTC from T₁ to P₂)

signature of P₁ on [T₂]

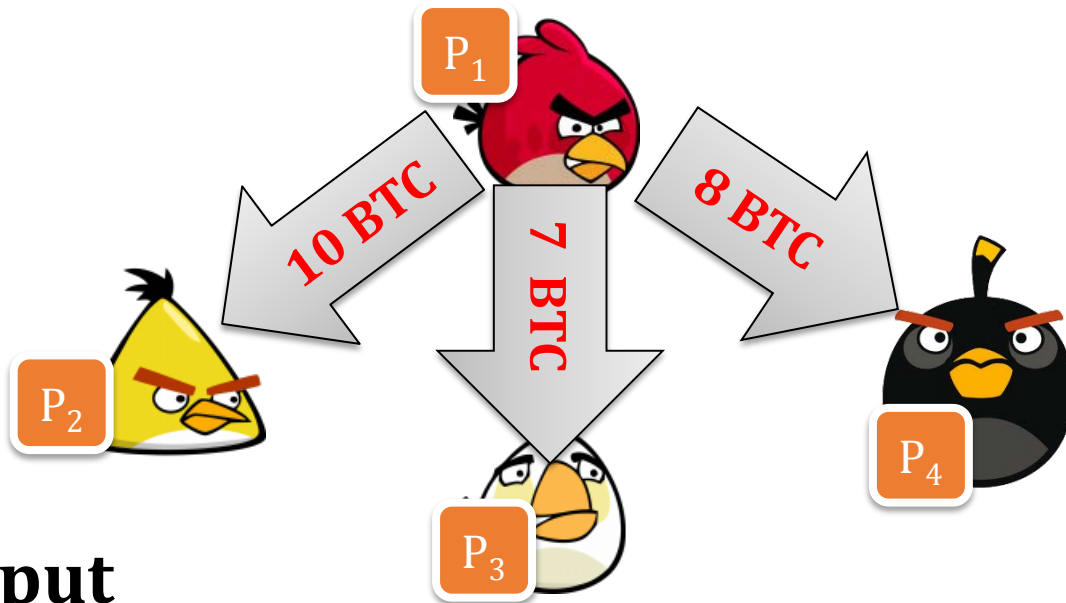
[T₃]

T₃ =

(User P₂ sends 25 BTC from T₂ to P₃)

signature of P₂ on [T₃]

How to “divide money”?



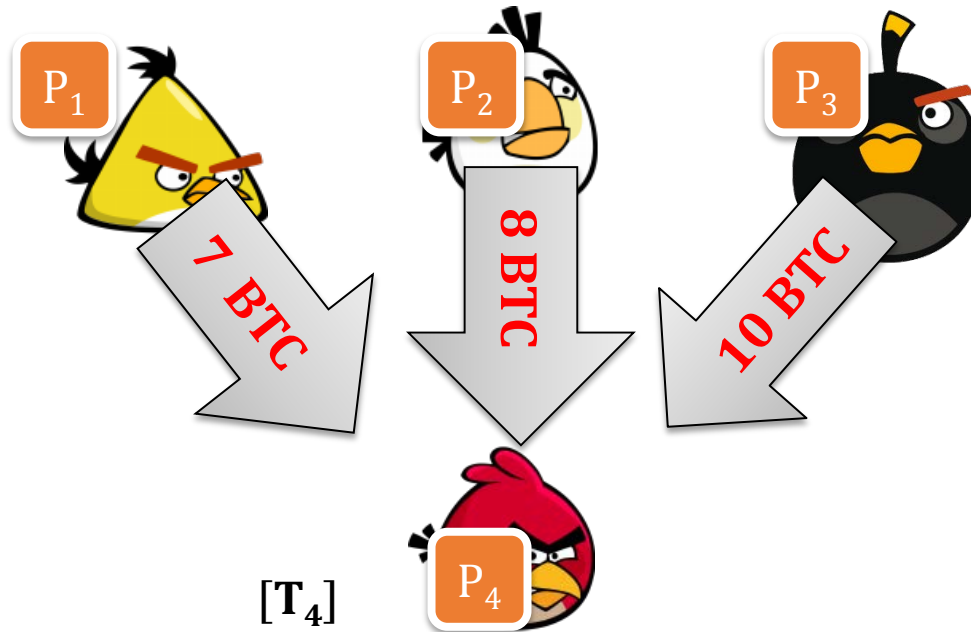
Multi-output
transactions:

$[T_2]$

$T_2 =$ (User P_1 sends 10 BTC from T_1 to user P_2 ,
User P_1 sends 7 BTC from T_1 to user P_3 ,
User P_1 sends 8 BTC from T_1 to user P_4)

signature of P_1 on $[T_2]$

Multiple inputs



$T_4 =$

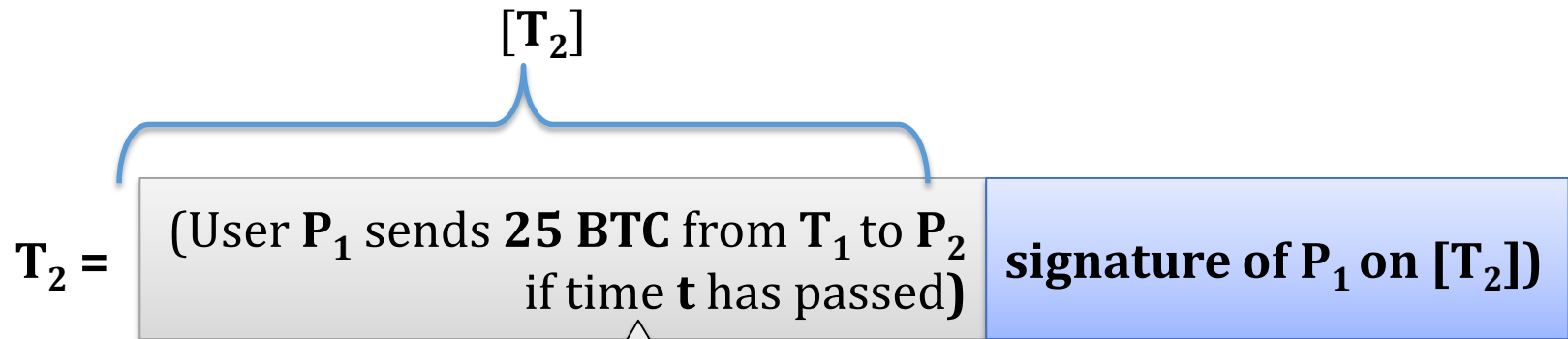
(User P_1 sends 10 BTC from T_1 to user P_4 ,
User P_2 sends 7 BTC from T_2 to user P_4 ,
User P_3 sends 8 BTC from T_3 to user P_4)

signature of P_1 on $[T_4]$,
signature of P_2 on $[T_4]$,
signature of P_3 on $[T_4]$)

all signatures need to be valid!

Time-locks

It is also possible to specify time **t** when a transaction becomes valid.



measured in:

- **real time**, or
- **blocks**.