ACM CCS'15, Denver, Colorado, Oct 15 2015

Based on slides from:

Introduction to Cryptocurrencies

a tutorial

Stefan Dziembowski
University of Warsaw

With some added and removed material

INNOVATIVE ECONOMY
NATIONAL COHESION STRATEGY
FNP
Foundation for Polish Science
EUROPEAN UNION
EUROPEAN REGIONAL DEVELOPMENT FUND

Announcements

- Midterm Upcoming on 3/13
  - Review sheet and practice exam posted on course webpage/Canvas
  - Solutions and Cheat Sheet posted soon on Canvas

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

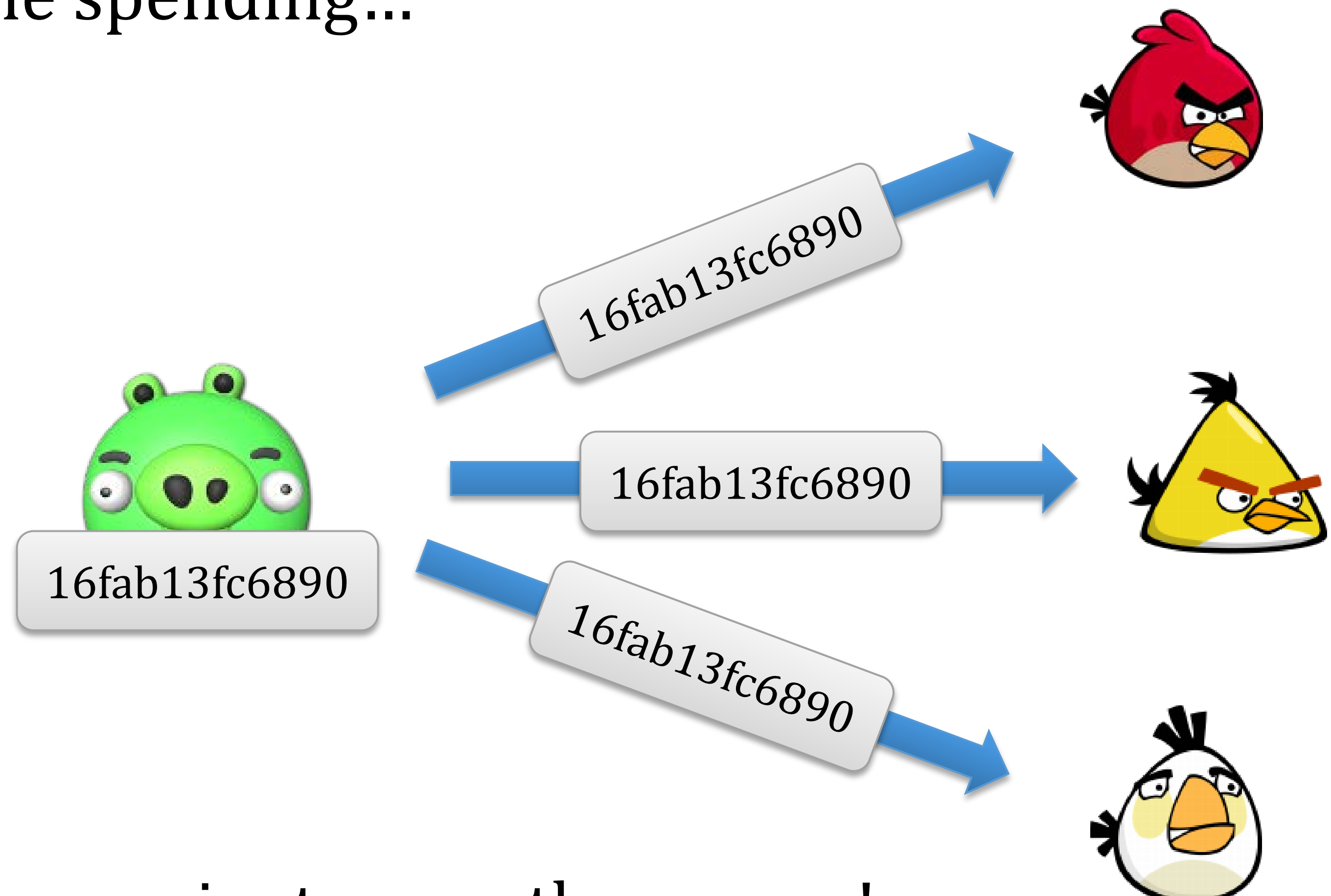
Part I

Introduction to Bitcoin

Main design principles

Main problem with the digital money

Double spending…



Bits are easier to copy than paper!

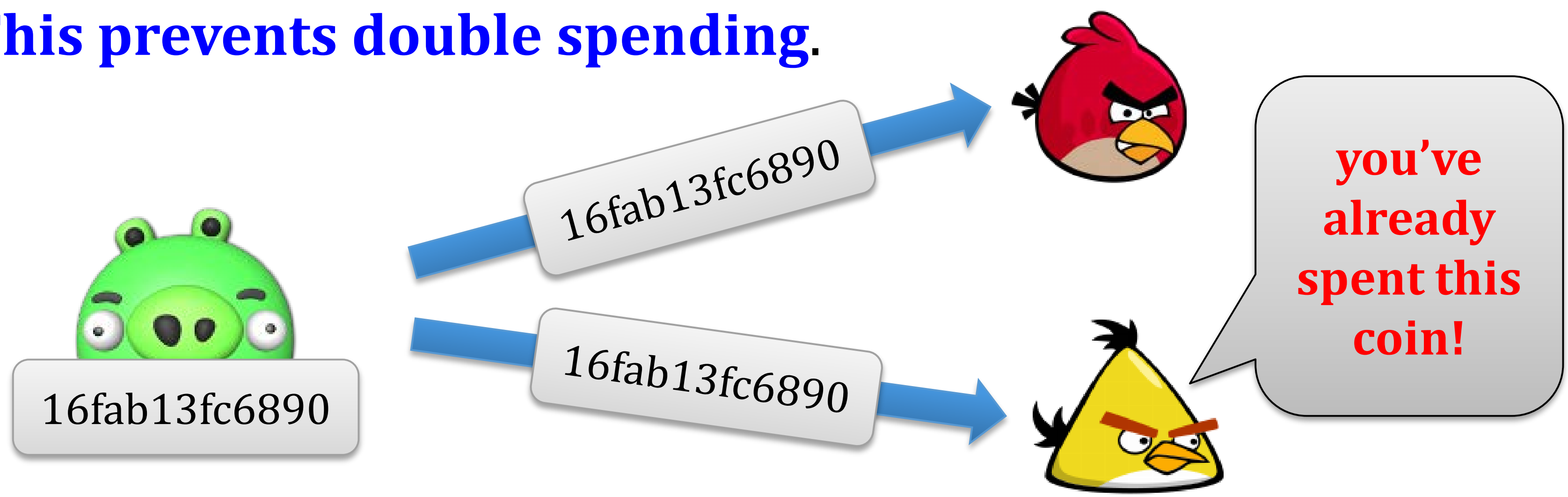
Bitcoin idea (simplified):

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

A transaction is of a form:

"User $\mathbf{P_1}$ transfers a coin #16fab13fc6890 to user $\mathbf{P_2}$"

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-Damgard Transform = Blockchain

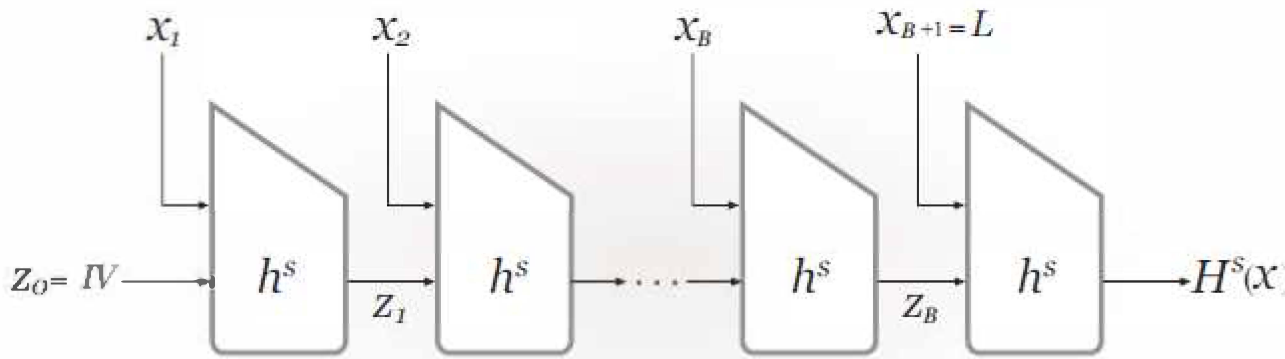


FIGURE 5.1: The Merkle-Damgård transform.

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

First attempt: Majority vote!

Problem

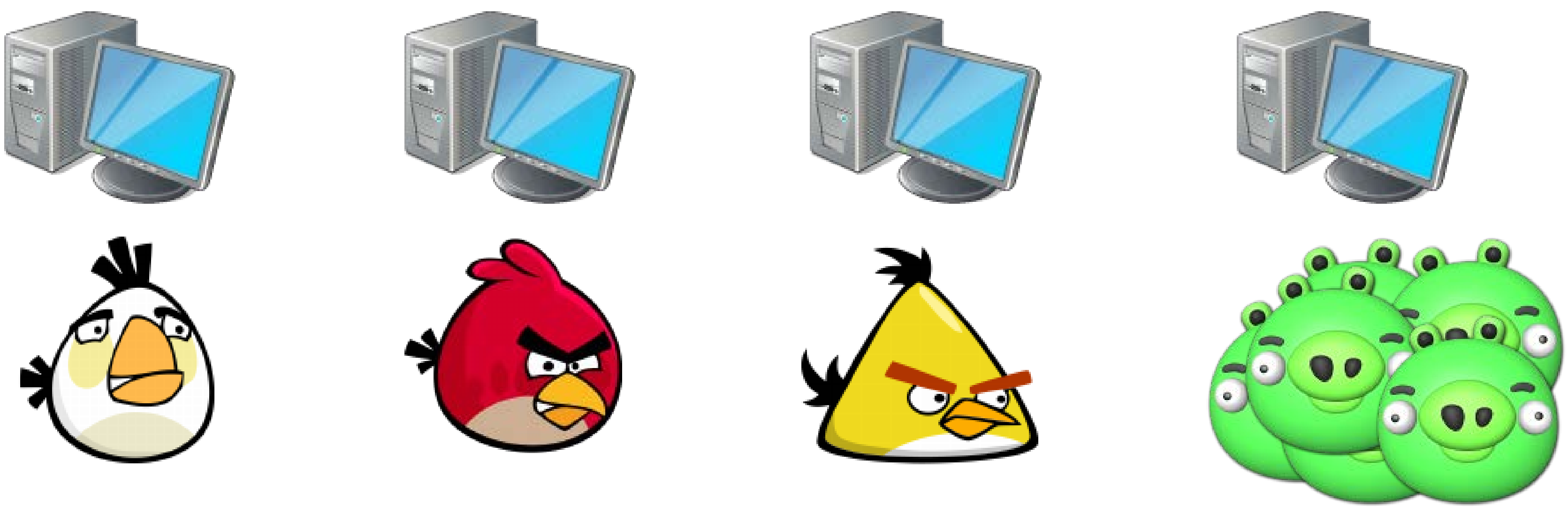
How to define "**majority**" in
a situation where
everybody can join the network?

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**.

<u>**Nowadays**</u>: 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)**

random **x**

s

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

salt | “hardness parameter

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

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

takes time **TIME(H)**

Main idea

The users participating in the scheme are called the "miners".

They maintain a chain of blocks:

block size < 1MB, which translates to max **7 trans./sec**.

the "**genesis block**" created by Satoshi on **03/Jan/2009**

≈ **10 min.**

block$_0$ → **block$_1$** → **block$_2$** → **block$_3$** →

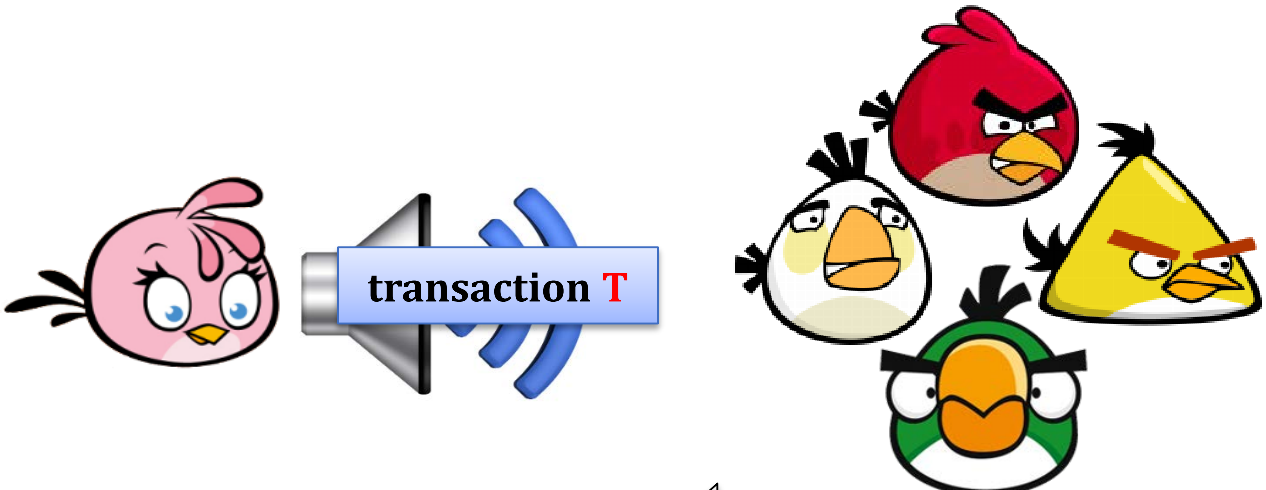
transactions from period **1**

transactions from period **2**

transactions from period **3**

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**.

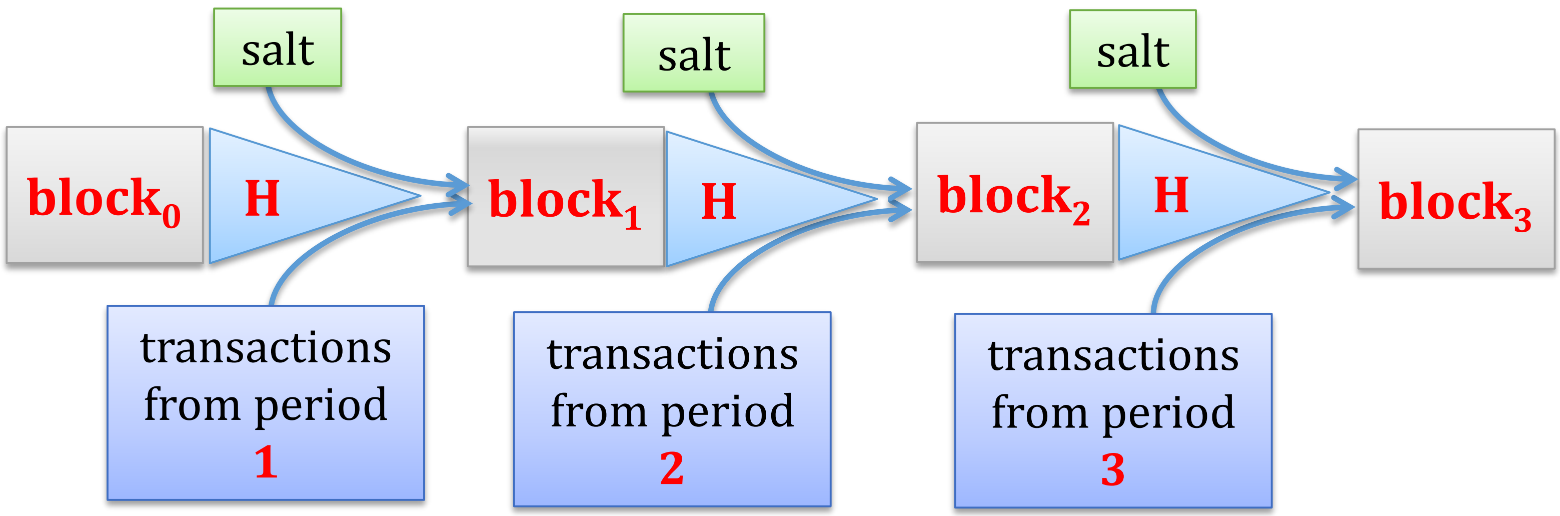
3. The users will always accept "**the longest chain**" as the valid one.

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

"hardness parameter"

“Hashrate” = number of hashes computed per second

total hashrate over the last **2** years:

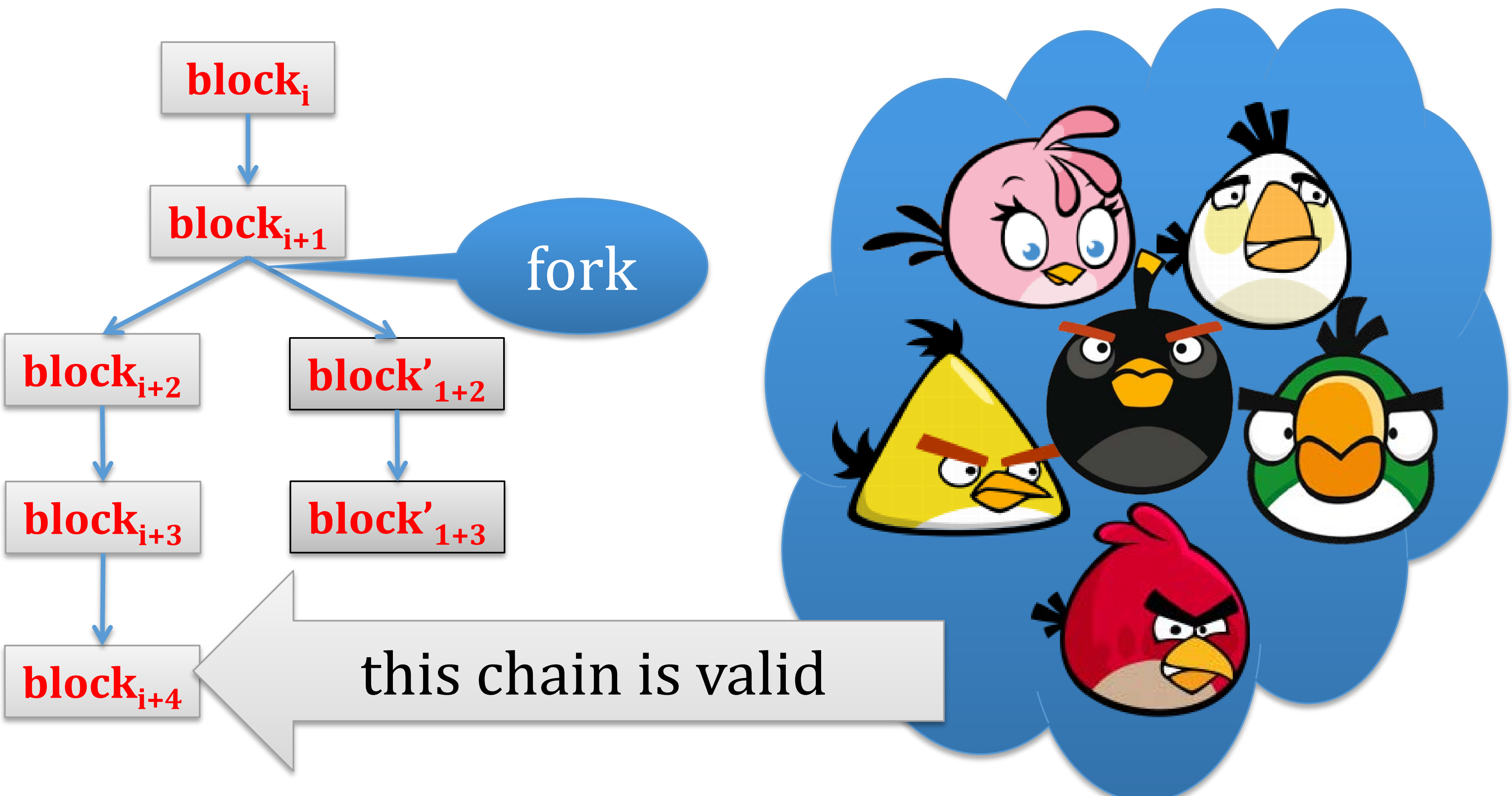
Note:

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

$\approx 2^{58}$ hash / second

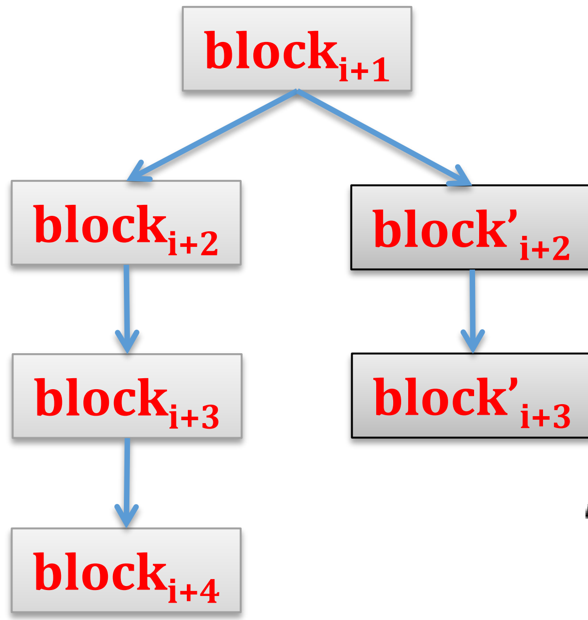
What if there is a “fork”?

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.

How are the miners incentivized to participate in this game?

Short answer: they are paid (in Bitcoins) for this.

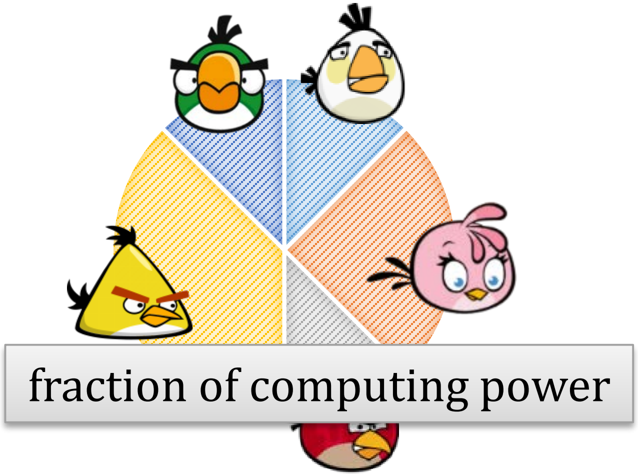
We will discuss it in detail later…

An important feature

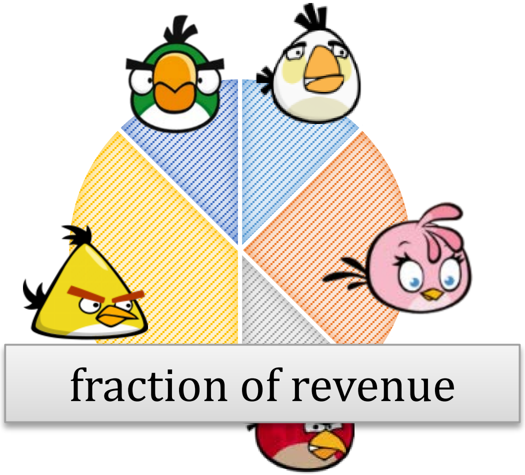
Suppose everybody behaves according to the protocol

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**.



$\approx$



Intuitively this is because:

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

Freshness of the genesis block

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

Here is a heuristic "proof":

Block$_0$ 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]**.

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:

input:		output:
	Gen	**(secret key sk, public key pk)**
(sk, message M)	**Sign**	**signature σ**
(pk, M, σ)	**Vrfy**	**yes/no**

Correctness:

for every **(sk,pk) := Gen()** and every **M** we have
Vrfy(pk,M,Sign(sk,M)) = yes

Security:

"without knowing **sk** it is infeasible to compute **σ** such that
Vrfy(pk,M,σ) = 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**:

$\approx$ **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**,

current reward

and so on…

Note: $\mathbf{210{,}000 \cdot (50 + 25 + 12.5 + \cdots) \to 21{,}000{,}000}$

More details

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

<u>Advantages</u>:

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)

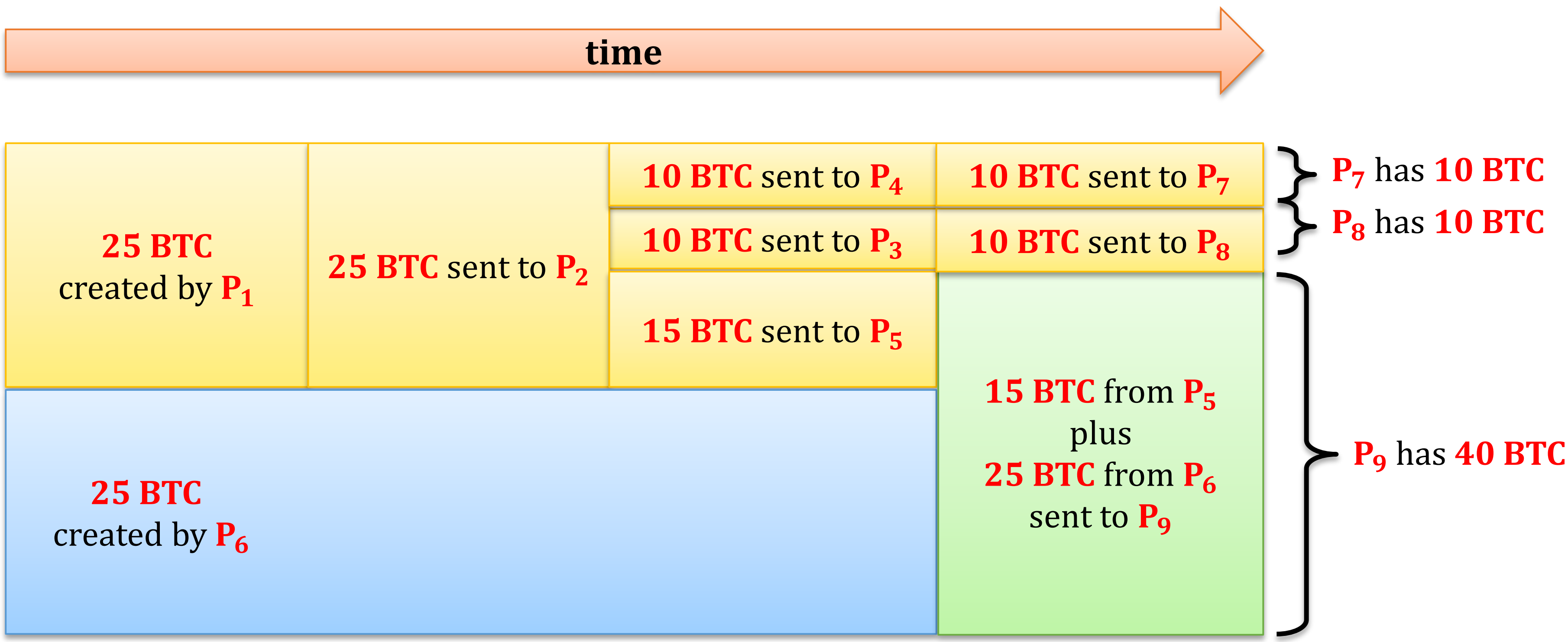
What needs to be discussed

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?

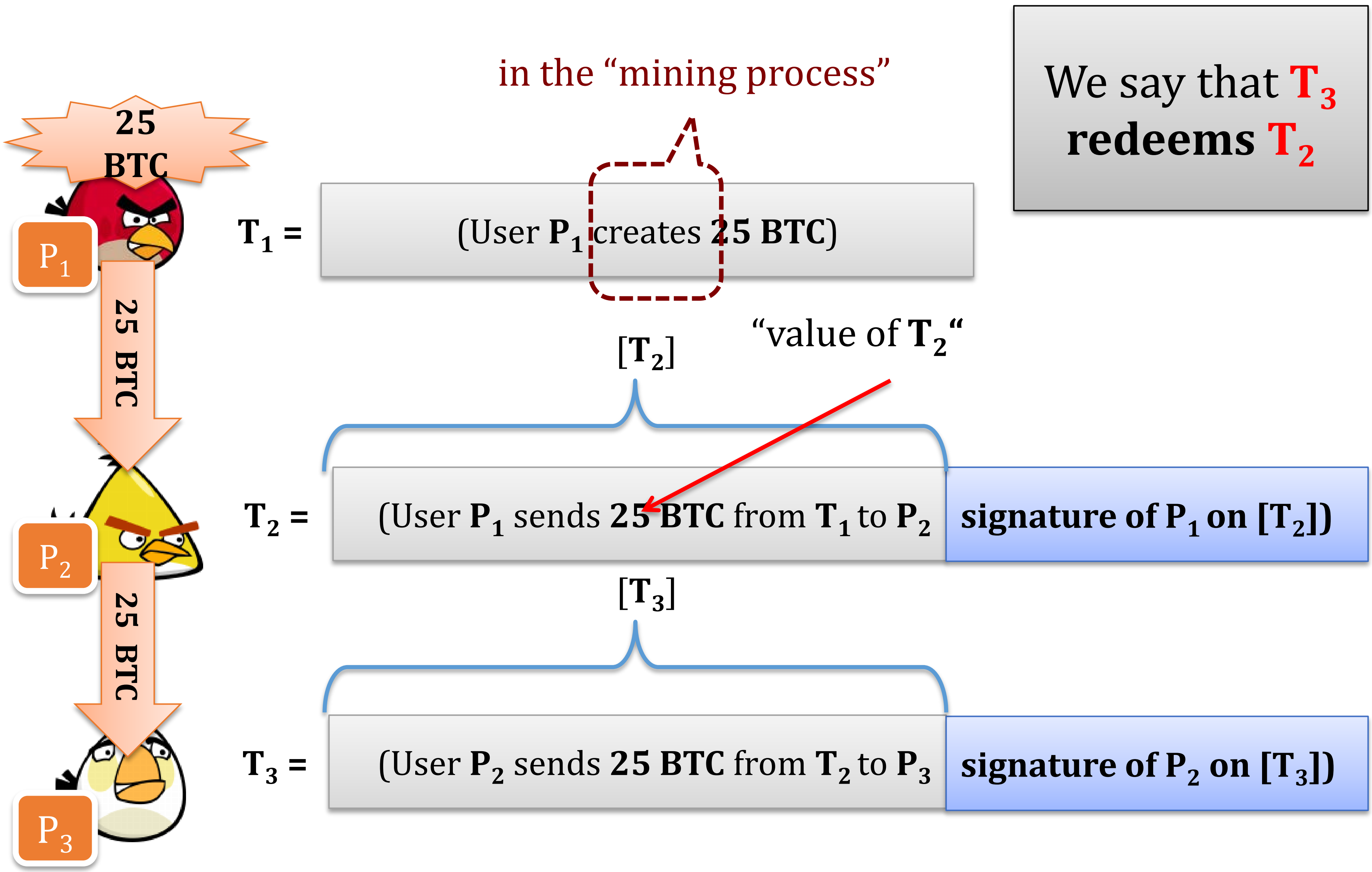
Bitcoin's money mechanics

Bitcoin is "transaction based".

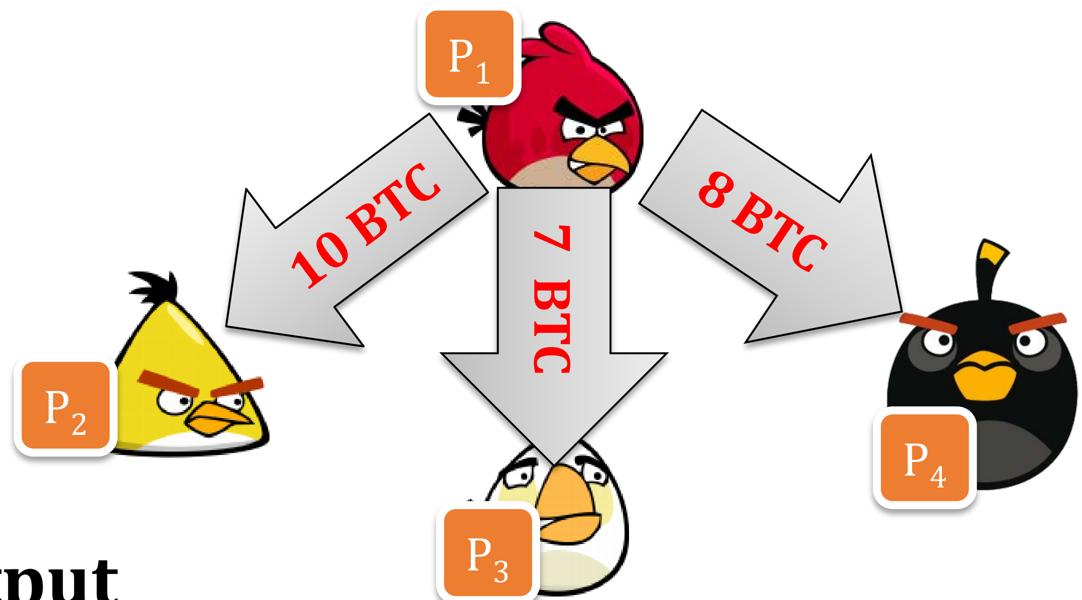
Technically: there is no notion of a "coin" in Bitcoin.



Transaction syntax – simplified view



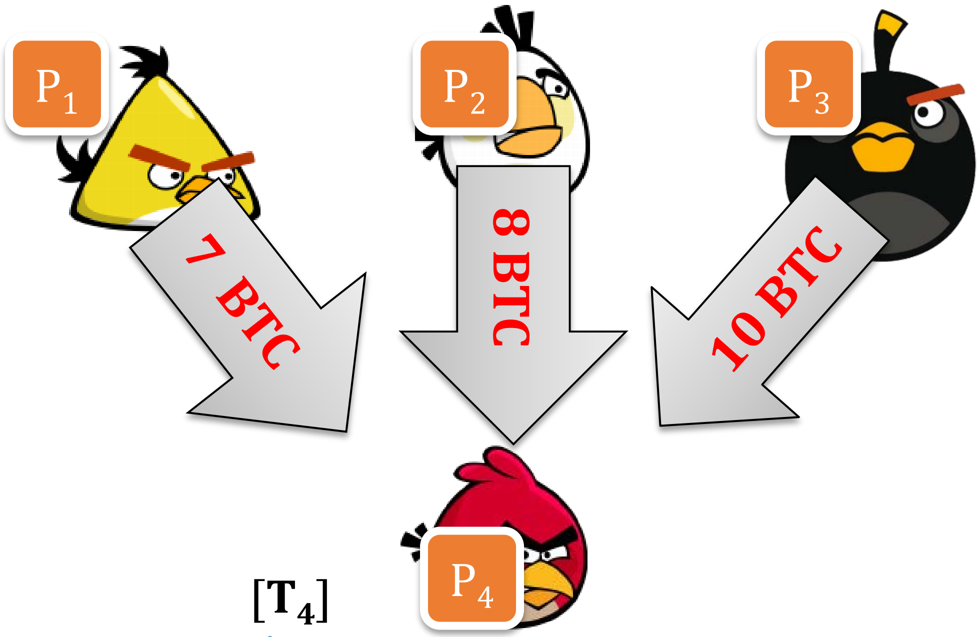
How to “divide money”?



Multi-output
transactions:

$$T_2 = \overbrace{\begin{array}{r}\text{(User } \mathbf{P_1} \text{ sends } \mathbf{10\ BTC} \text{ from } \mathbf{T_1} \text{ to user } \mathbf{P_2}\text{,} \\ \text{User } \mathbf{P_1} \text{ sends } \mathbf{7\ BTC} \text{ from } \mathbf{T_1} \text{ to user } \mathbf{P_3}\text{,} \\ \text{User } \mathbf{P_1} \text{ sends } \mathbf{8\ BTC} \text{ from } \mathbf{T_1} \text{ to user } \mathbf{P_4}\end{array}}^{[T_2]}\ \ \textbf{signature of } \mathbf{P_1} \textbf{ on } \mathbf{[T_2]}\text{)}$$

Multiple inputs



[T_4]

T_4 = (User $\mathbf{P_1}$ sends **10 BTC** from $\mathbf{T_1}$ to user $\mathbf{P_4}$,
User $\mathbf{P_2}$ sends **7 BTC** from $\mathbf{T_2}$ to user $\mathbf{P_4}$,
User $\mathbf{P_3}$ sends **8 BTC** from $\mathbf{T_3}$ to user $\mathbf{P_4}$

signature of $\mathbf{P_1}$ on [$\mathbf{T_4}$],
signature of $\mathbf{P_2}$ on [$\mathbf{T_4}$],
signature of $\mathbf{P_3}$ on [$\mathbf{T_4}$])

all signatures need to be valid!

Time-locks

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

$$\mathbf{T_2} = \overbrace{(\text{User } \mathbf{P_1} \text{ sends } \mathbf{25\ BTC} \text{ from } \mathbf{T_1} \text{ to } \mathbf{P_2} \text{ if time } \mathbf{t} \text{ has passed}}^{[\mathbf{T_2}]} \ \textbf{signature of } \mathbf{P_1} \textbf{ on } [\mathbf{T_2}])$$

measured in:

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