CS 4770: Cryptography

CS 6750: Cryptography and Communication Security

Alina Oprea
Associate Professor, CCIS
Northeastern University

April 9 2018

Schedule

- HW 4
  - Due on Thu 04/12
- Programming project 3
  - Due on 04/26 (last day it can be accepted)
  - Grading on 04/27
- Final exam
  - 04/23, 1-3pm
- Office hours during the week of 04/16

Bitcoin

- Digital crypto currencies
  - Advantages over paper cash
- Distributed public ledger
  - Blockchain creation and distribution
  - Proof of Work (PoW)
  - Agreement and resilience to adversaries
  - Incentives for users
- Bitcoin security
- Other cryptocurrencies

Resources

- Book: Bitcoin and Cryptocurrency Technologies
  - http://bitcoinbook.cs.princeton.edu/
- Bonneau et al. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies.
  - https://eprint.iacr.org/2015/261.pdf
- Bitcoin and Cryptocurrency Technologies Course
  - https://www.coursera.org/learn/cryptocurrency
  - https://piazza.com/princeton/spring2015/btctech/resources

Bitcoin – a “digital analogue” of the paper money

A digital currency introduced by “Satoshi Nakamoto” in 2008

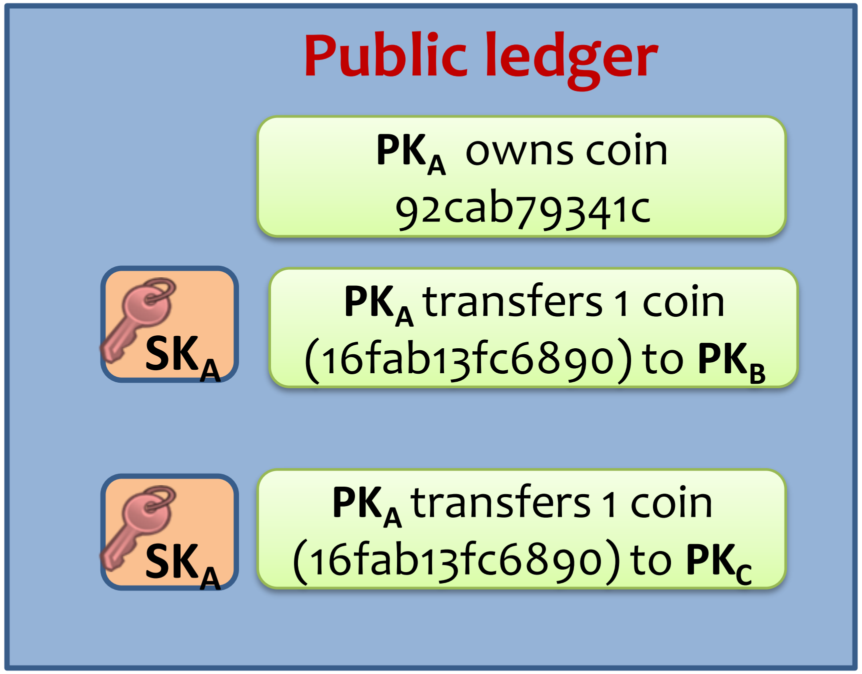
- First e-cash without a centralized issuing authority
  - Store and transfer value without reliance on central banks
  - Anyone can join the system and make transactions
  - Transactions are publicly verifiable
- Built on top of an unstructured P2P system
  - Participants validate transactions and mint currency
  - System works as long as the *majority of users are honest*

Currency unit: **Bitcoin (BTC) 1 BTC = 10^8 Satoshi;** value ≈ $6800

Bitcoin idea

Public trusted bulletin-board (public ledger or DB)

- Includes list of all transactions
- Verifiable by all users
- How to maintain it in decentralized fashion?

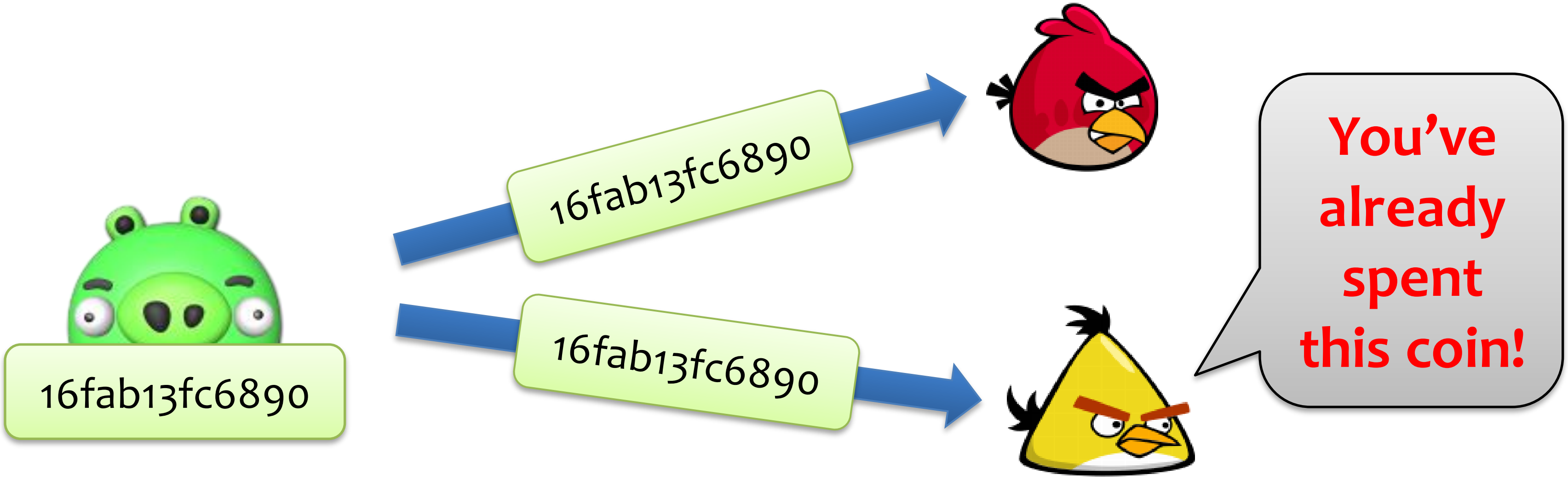


Public ledger

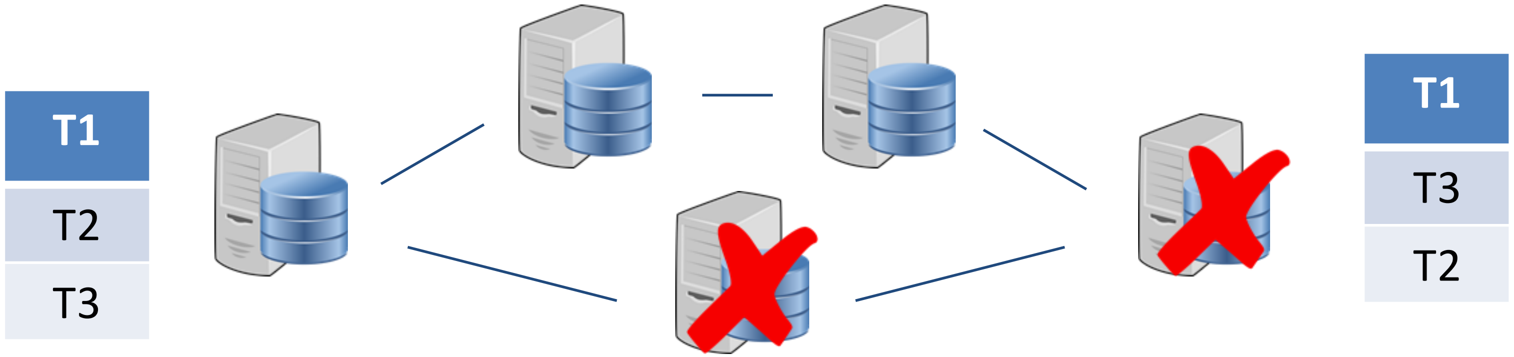
PK_A owns coin 92cab79341c

SK_A **PK_A** transfers 1 coin (16fab13fc6890) to **PK_B**

SK_A **PK_A** transfers 1 coin (16fab13fc6890) to **PK_C**



Challenges in designing public ledger



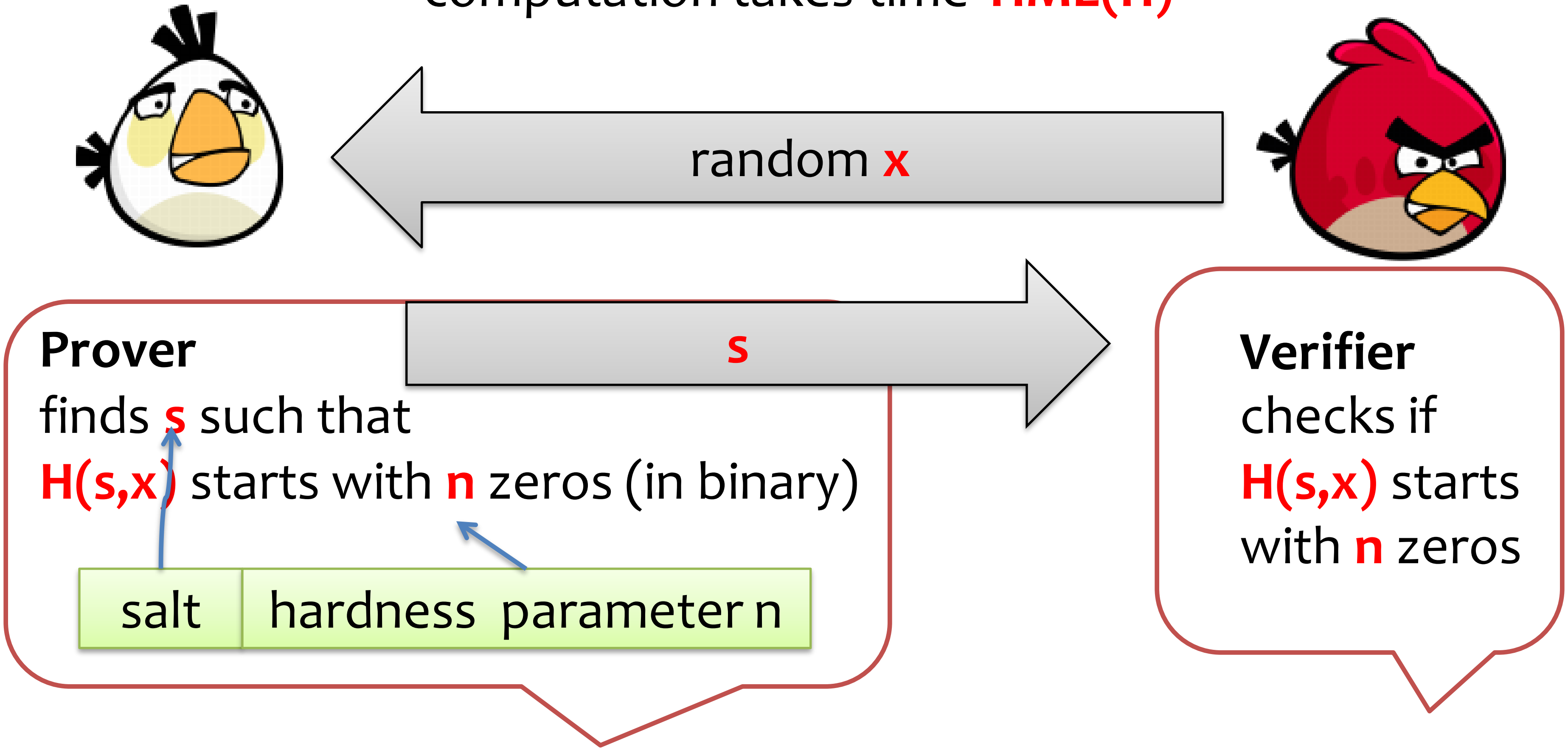
- **Decentralized ledger**
  - Each user maintains a list of transactions ordered across time
  - His own transactions and transactions received from other users
- **Main challenges: Obtain consensus**
  - Order of transactions is the same at all nodes
- **Attack models**
  - Network failures (messages might not be delivered timely)
  - Offline participants (nodes leave and re-connect to the network)
  - Malicious nodes (nodes try to double spend)

Distributing transactions

- New transactions are *broadcast* to all nodes
  - P2P network
- Each node *collects new transactions* into a block
- In each round (e.g., every 10 minutes)
  - A random node *creates the next block* (includes outstanding transactions) and broadcasts it to the network
- Other nodes accept the block only if *all transactions in it are valid*
  - Valid signatures
  - Coins not spent before
- Nodes accept the block by including it in their local ledger

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 n

Verifier

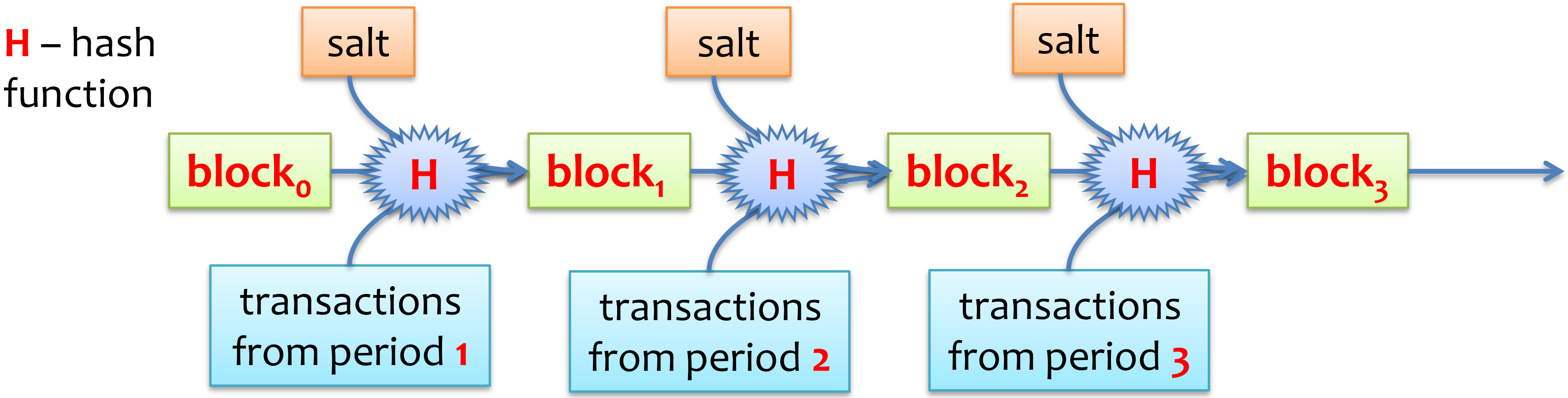
checks if

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

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

takes time **TIME(H)**

How are the PoWs used?



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

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

Process is called block mining

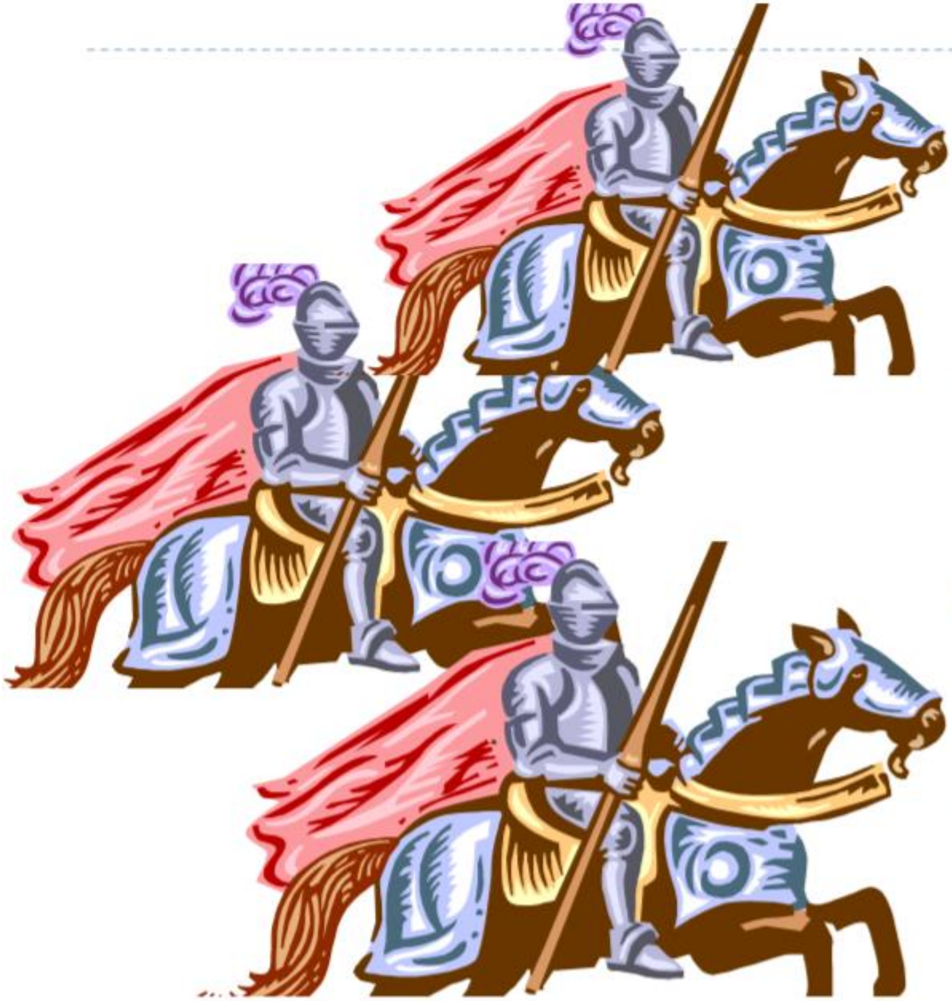
Public ledger

- Tamper-evident log
  - Record and order all transactions
  - Valid transactions can not be modified
  - New transactions are appended after being validated
- How to design it?
  - What data structure and crypto primitives to use?
- How to prevent attackers controlling majority of transactions?
- How to reach agreement?
- How to incentivize users?

Distributed consensus

- Traditional application
  - Reliability in distributed systems
- Fixed number of nodes, each has an input
- Requirements
  - The protocol terminates and all correct nodes output same value
  - The value they output is input to a correct node
- In Bitcoin nodes need to reach consensus on order of transactions
  - Adversary can create arbitrary transactions

Byzantine Generals Problem



- Several Byzantine generals are laying siege to an enemy city
- They can only communicate by messenger
- They have to agree on a common strategy, *attack* or *retreat*.
- Some general may be traitors (their identity is not known)

Why consensus is hard

- Many reasons
  - Nodes may crash (benign failures)
  - Nodes may be malicious (adversarial failures)
  - Network is imperfect
    - Faults, dis-connections, variable latency
  - No notion of global time
- Impossibility results
  - Asynchronous setting (messages might be delayed indefinitely)
  - FLP theorem: It is impossible to achieve consensus with single node failing in asynchronous networks

Consensus in Bitcoin

- New transactions are *broadcast* to all nodes
- Nodes *collect new transactions* into a block
- In each round the node **solving the puzzle** *generates and sends the next block*
  - Other nodes accept the block only if *all transactions in it are valid*
  - They all broadcast the new block to all nodes
- Nodes accept the block by including it in their local ledger **and mining on the new block**

Probabilistic guarantee to get around impossibility results

Eventual consistency

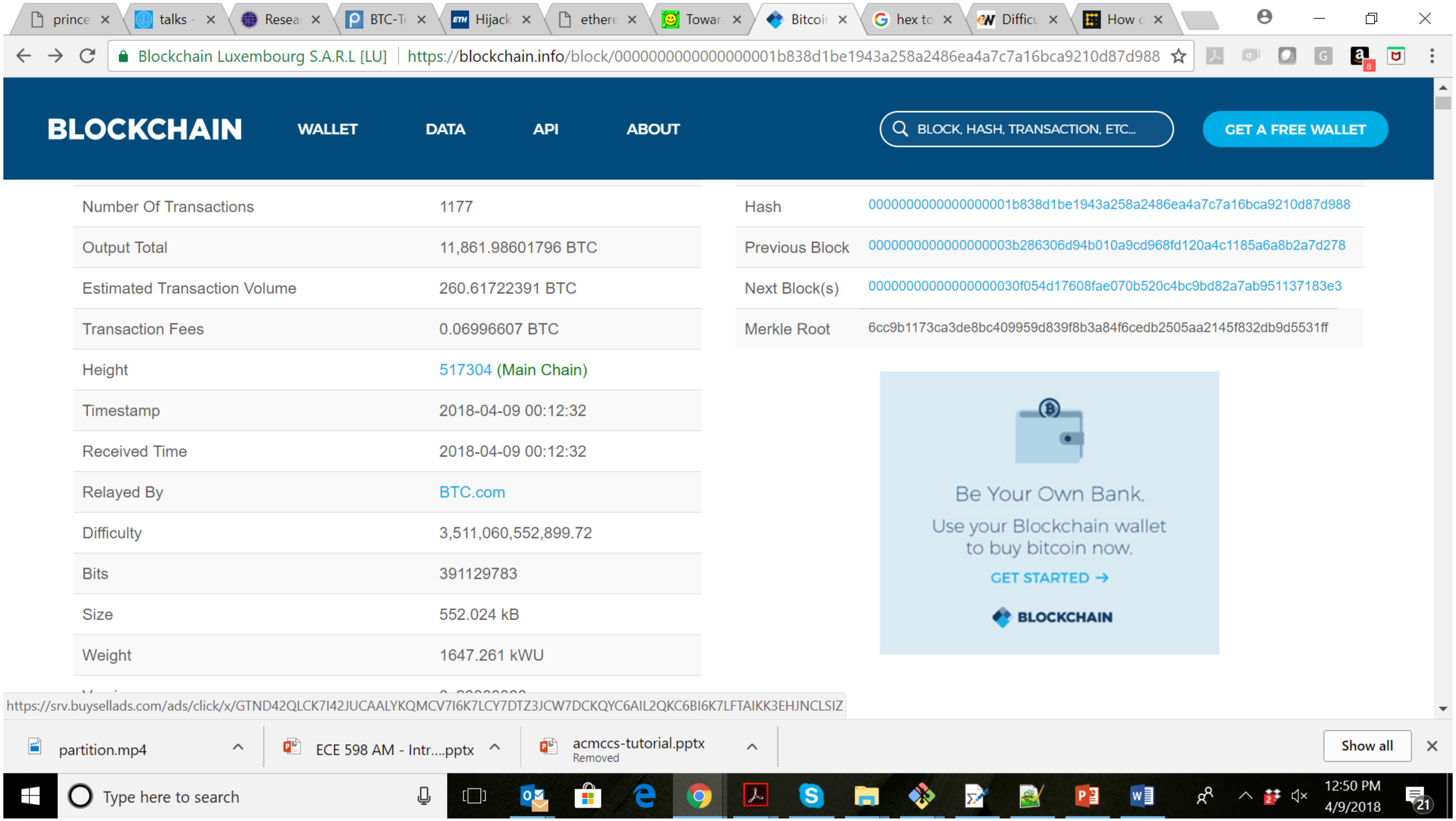
- Consensus doesn’t happen right away
- At least 10 mins to verify a transaction
  - Agree to pay
  - Wait for one block (10 mins) for the transaction to go through
  - But, for a large transaction ($$$) wait longer.
    - If you wait longer there will be more blocks mined and higher probability that your transaction is on the consensus chain
    - For large $$$, you wait for six blocks (1 hour) or longer
    - E.g., if a vendor requires 6 confirmations and an attacker controls 10% of the CPU power, the attack will succeed 0.02428% of the time

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**.
- For example the block generated on 2014-03-17 18:52:10 looked like this:

```
00000000000000006d8733e03fa9f5e5
2ec912fa82c9adfed09fbca9563cb4ce
```

Bitcoin Snapshot



Number Of Transactions	1177
Output Total	11,861.98601796 BTC
Estimated Transaction Volume	260.61722391 BTC
Transaction Fees	0.06996607 BTC
Height	517304 (Main Chain)
Timestamp	2018-04-09 00:12:32
Received Time	2018-04-09 00:12:32
Relayed By	BTC.com
Difficulty	3,511,060,552,899.72
Bits	391129783
Size	552.024 kB
Weight	1647.261 kWU

Hash	0000000000000000001b838d1be1943a258a2486ea4a7c7a16bca9210d87d988
Previous Block	0000000000000000003b286306d94b010a9cd968fd120a4c1185a6a8b2a7d278
Next Block(s)	00000000000000000030f054d17608fae070b520c4bc9bd82a7ab951137183e3
Merkle Root	6cc9b1173ca3de8bc409959d839f8b3a84f6cedb2505aa2145f832db9d5531ff

https://srv.buysellads.com/ads/click/x/GTND42QLCK7I42JUCAALYKQMCV7I6K7LCY7DTZ3JCW7DCKQYC6AIL2QKC6BI6K7LFTAIKK3EHJNCLSIZ

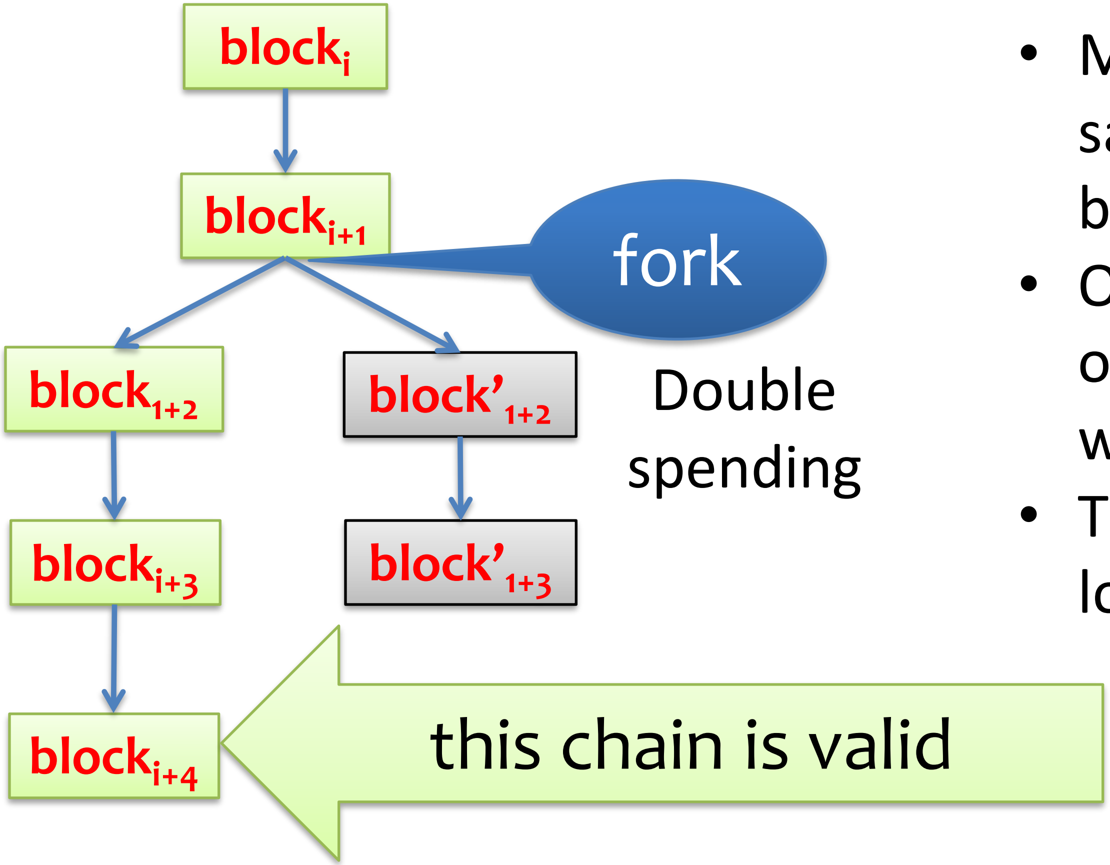
Attacks

- Adversary generates new blocks
  - Can happen proportional to his computational power (less than 50%)
- A transaction generated by honest node is not included by adversary in his generated blocks
  - Eventually honest nodes will solve the puzzle and include all outstanding transactions
  - As long the honest node is connected to other honest nodes, his transactions will be included in a block
    - It might take some time!

Double spending: “forks”

The “**longest**” chain counts.

- It includes “more work”



block_i

block_{i+1}

fork

block_{1+2}

block'_{1+2}

Double spending

block_{i+3}

block'_{1+3}

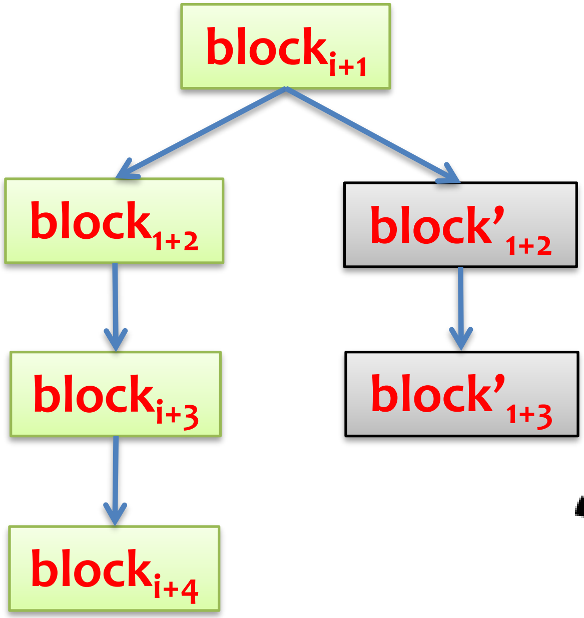
block_{i+4}

this chain is valid

- Malicious nodes spend same coin on different branches
- Once network reconciles, only one of the transaction will be accepted
- The transaction on the longest chain

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.

Bitcoin Protocol

- Each P2P node runs the following algorithm:
  - New transactions are broadcast to all nodes.
  - Each node (miner) collects new transactions into a block.
  - Each node works on solving proof-of-work (PoW) for its block
    - Use computational resources
  - When a node finds a solution, it broadcasts the block to all nodes.
  - Nodes accept the block only if all transactions are valid (digital signature checking) and coins not already spent (check transactions from public ledger).
  - Nodes express their acceptance by working on creating the next block in the chain
    - If multiple valid blocks are available, choose the longest chain and include transactions from discarded blocks in the queue
    - Include the hash of the accepted block as the previous hash.

Nodes eventually reach global consensus on all transactions

Main principles

1. It is **computationally hard** to extend the chain (solve puzzle)

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

4. **Wait longer** to perform action according to the value of the transaction

Public ledger

- Tamper-evident log
  - Record and order all transactions
  - Valid transactions can not be modified
  - New transactions are appended after being validated
- How to design it?
  - What data structure and crypto primitives to use?
- How to prevent attackers controlling majority of transactions?
- How to reach agreement?
- How to incentivize users?

How are the miners incentivized to participate in this game?

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

Incentives

- Transactions may include a transaction fee
  - Paid to whoever mines a block that includes the transaction
- New blocks mint new coins
  - Node who wins “mines” a fixed amount of coins as a prize
  - Called a coinbase transaction
  - The only way to generate new coins in the system

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**,

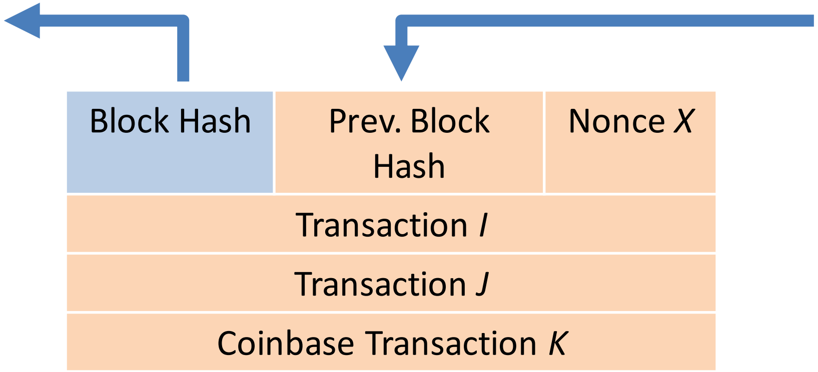
current reward

and so on…

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

Fixed number of blocks in the system

Coinbase Transactions



Block Hash	Prev. Block Hash	Nonce *X*
Transaction *I*		
Transaction *J*		
Coinbase Transaction *K*		

- Generated upon successful mining and included in block chain
- Node will get the reward only if this transaction is on the blockchain
- Elegantly solves several problems
  - Where do bitcoins come from?
  - How are they minted?
  - Who gets newly minted coins?

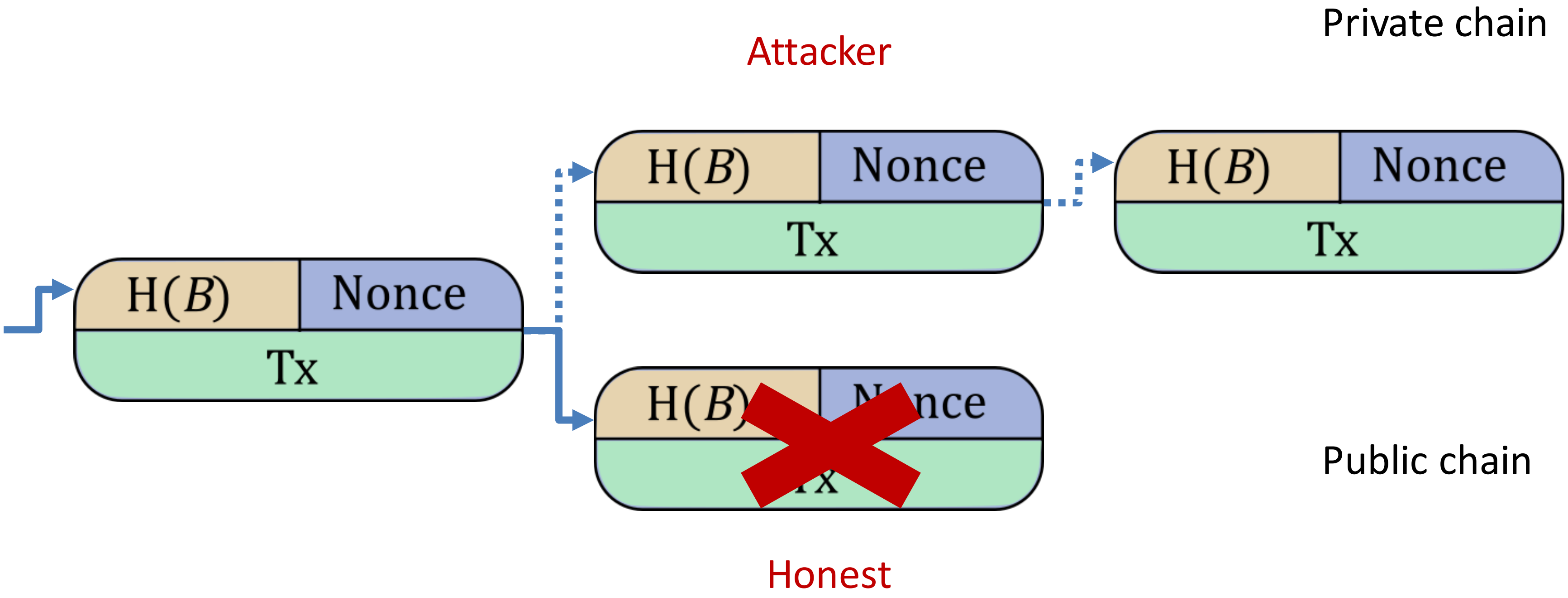
Bitcoin security

- Protection again *invalid transactions* (forgery)
  - Cryptographic (digital signature)
- Protection against *modification of blockchain* (remove or modify old transactions)
  - Cryptography (collision-resistant hash functions and digital signatures)
- *Non-repudiation of transactions*
  - Based on blockchain
- Protection against *double spending*
  - Enforced by consensus (correct majority)
  - One of the transactions (either one) will be eventually accepted
- Protection against *Sybil attacks*
  - PoW cryptographic puzzles
  - Assume that adversary does not control majority of CPU resources

Selfish mining attacks

[Majority is not Enough: Bitcoin Mining is Vulnerable]

Ittay Eyal, and Emin Gün Sirer

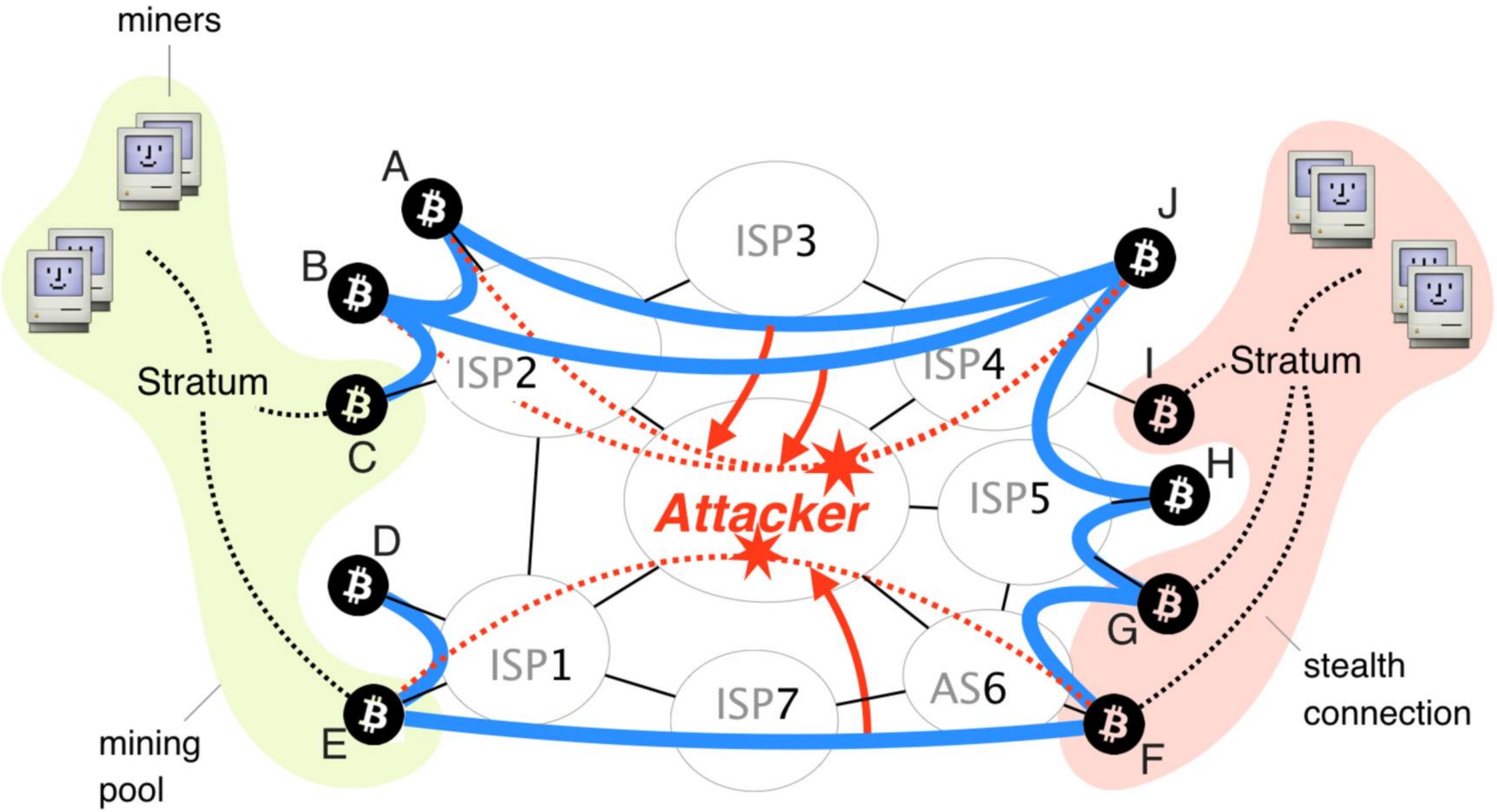


- Attacker: I'll keep these blocks for myself!
  - Create private chain
- When attacker reveal blocks in private chain, honest blocks will switch to longer chain and waste resources
  - Creates incentives for honest users to join attacker coalition

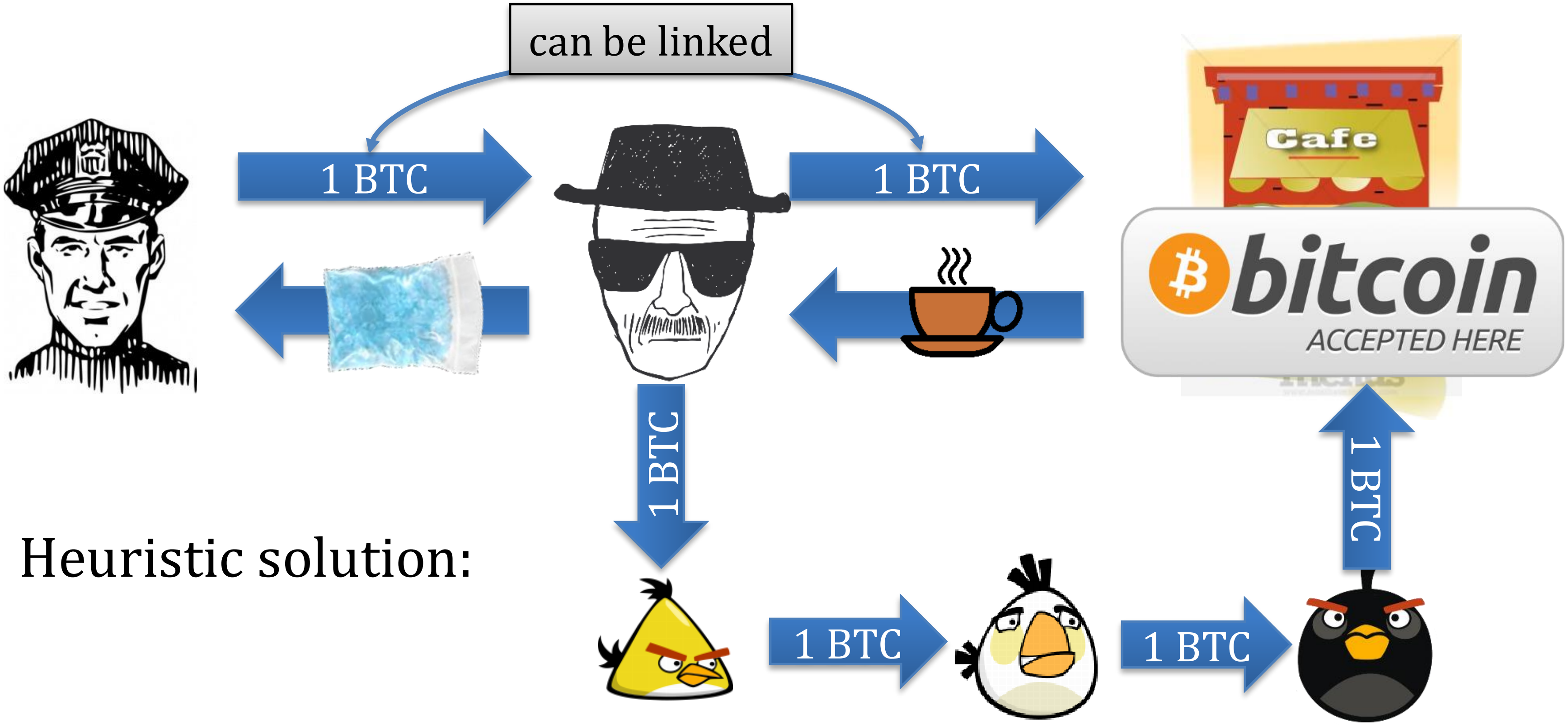
Partitioning attacks

Hijacking Bitcoin: Routing Attacks on Cryptocurrencies

Maria Apostolaki, Aviv Zohar, Laurent Vanbever



One obvious problem: lack of anonymity

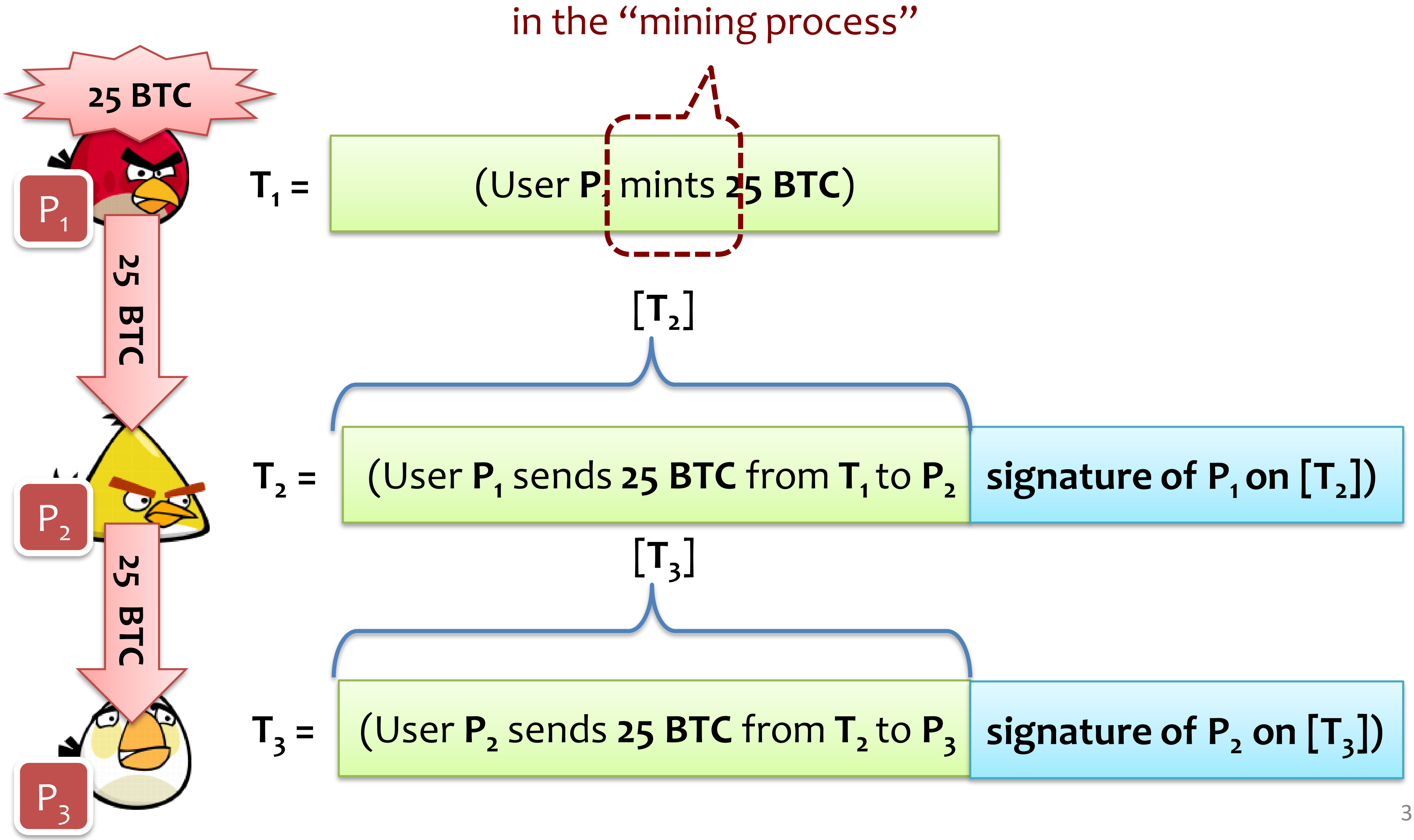


Heuristic solution:

Can sometimes be de-anonymized:

[Meiklejohn et al., *A fistful of bitcoins: characterizing payments among men with no names*, 2013]

Transaction syntax – simplified view



in the “mining process”

T_1 = (User **P_1** mints **25 BTC**)

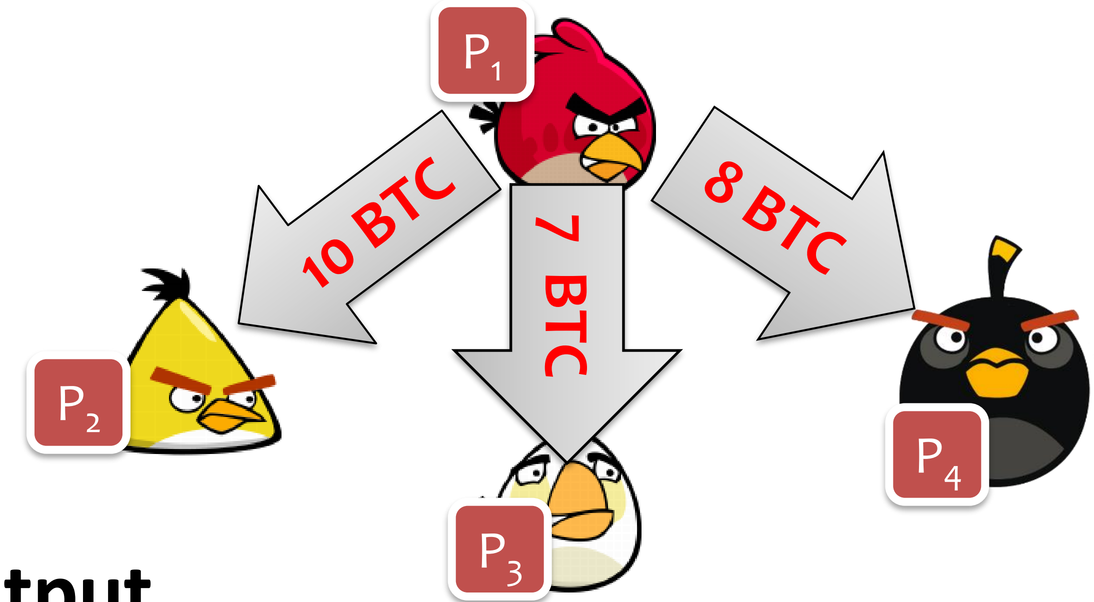
[T_2]

T_2 = (User **P_1** sends **25 BTC** from **T_1** to **P_2** | **signature of P_1 on [T_2]**)

[T_3]

T_3 = (User **P_2** sends **25 BTC** from **T_2** to **P_3** | **signature of P_2 on [T_3]**)

How to “divide money”?

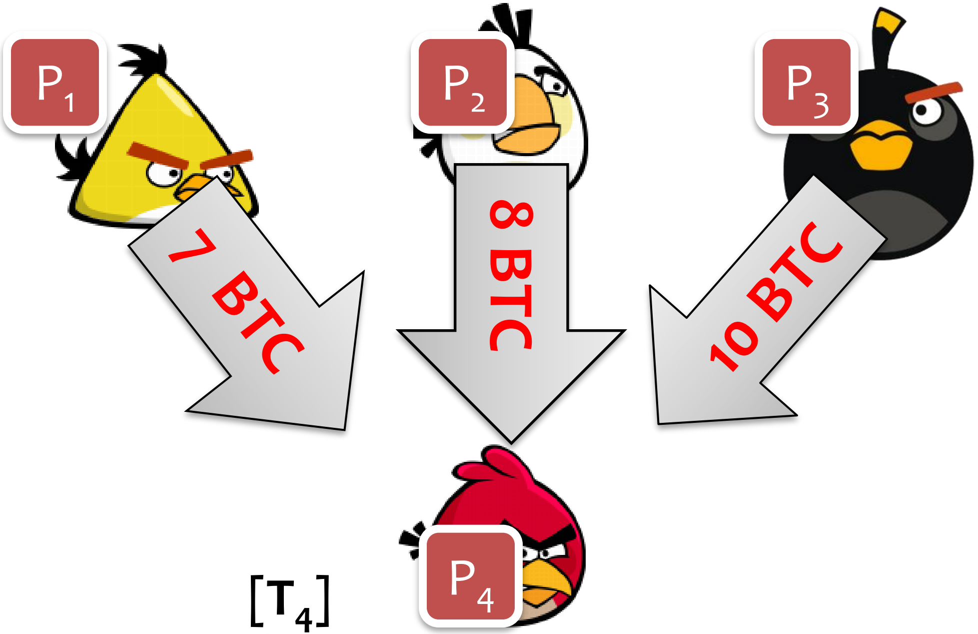


Multi-output
transactions:

[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]

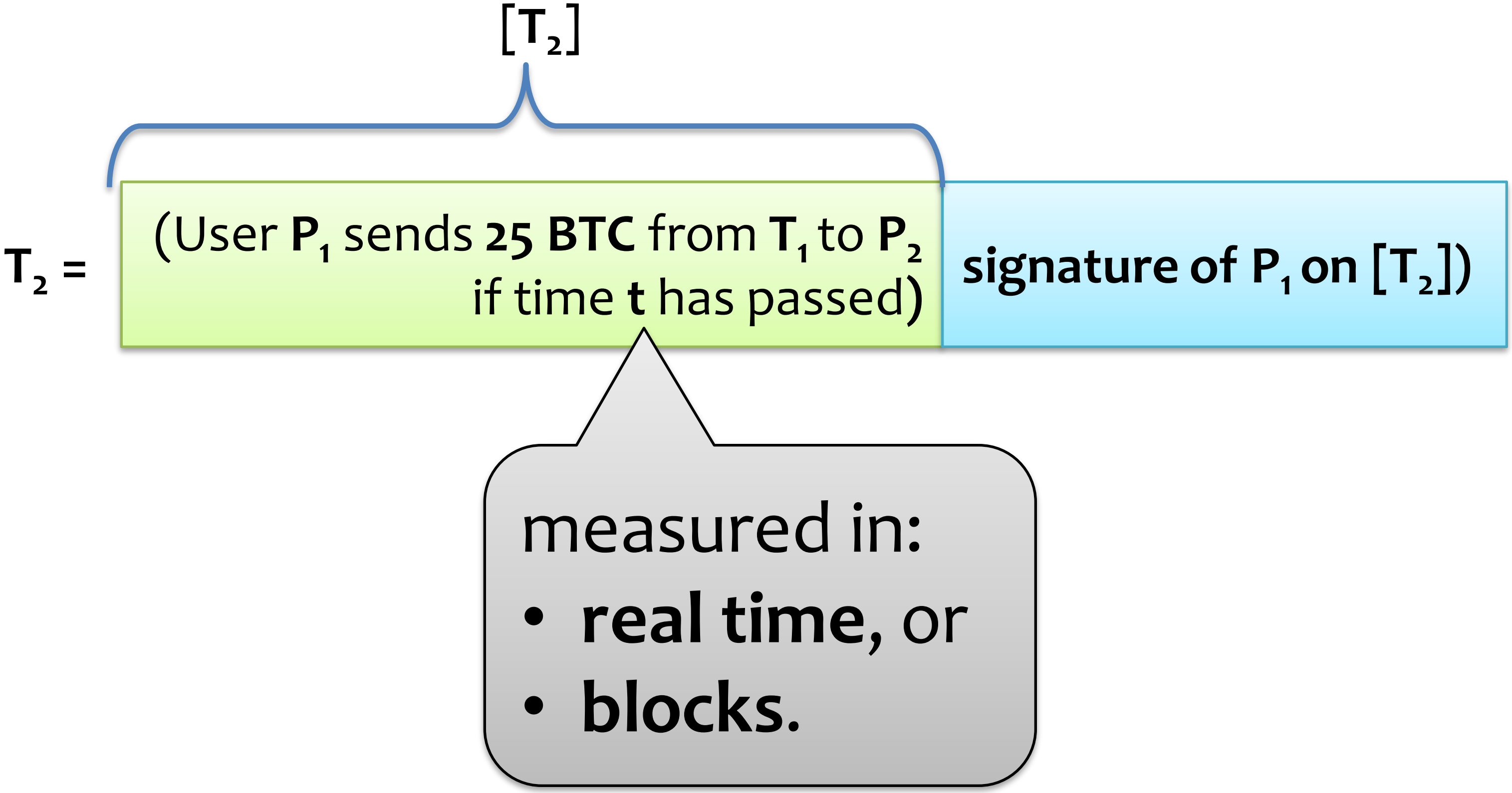
T_4 = (User **P_1** sends **10 BTC** from **T_1** to user **P_4**,
User **P_2** sends **7 BTC** from **T_2** to user **P_4**,
User **P_3** sends **8 BTC** from **T_3** to user **P_4**

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.



[T_2]

T_2 = (User **P_1** sends **25 BTC** from **T_1** to **P_2** if time **t** has passed) **signature of P_1 on [T_2])**

measured in:

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

Generalizations

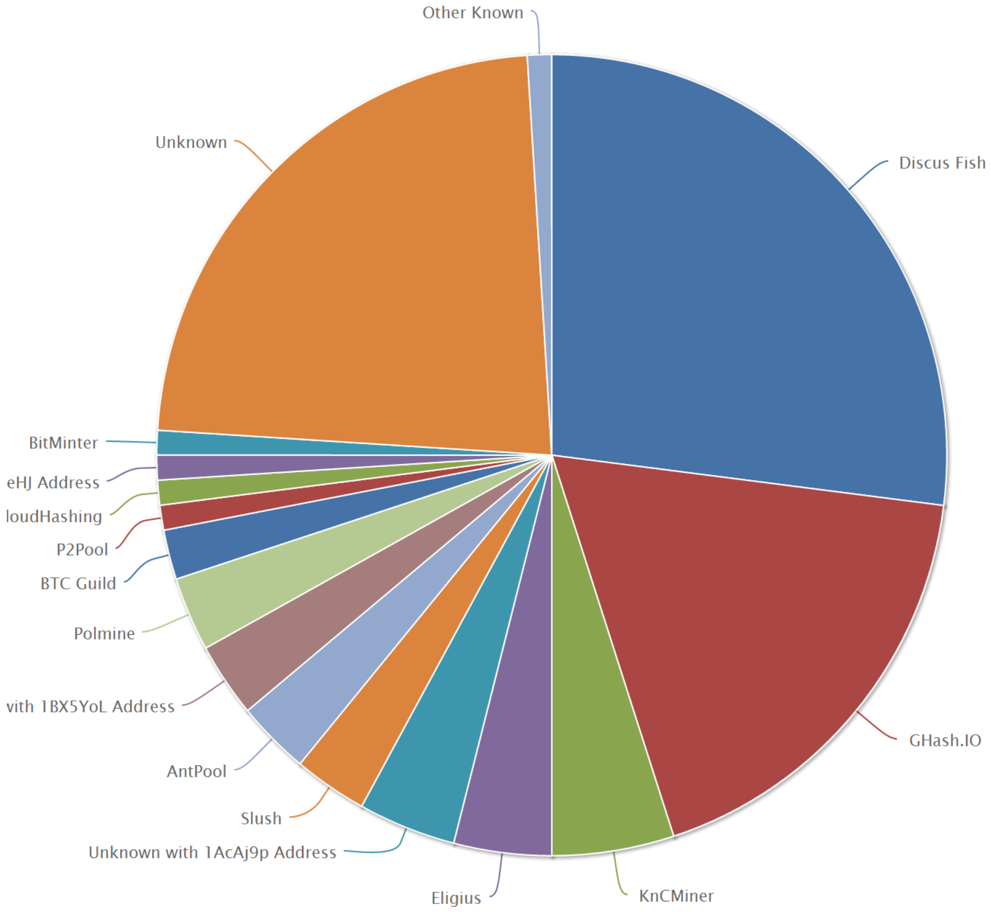
1. All these features can be combined.
2. The total value of **in-coming transactions** can be larger that the value of the **out-going transactions**.

   (the difference is called a "**transaction fee**" **and goes to the miner**)
3. Second mechanism to incentivize nodes to be honest

Popular mining pools

Miners create cartels called **mining pools**

This allows them to reduce the variance of their income



The general picture

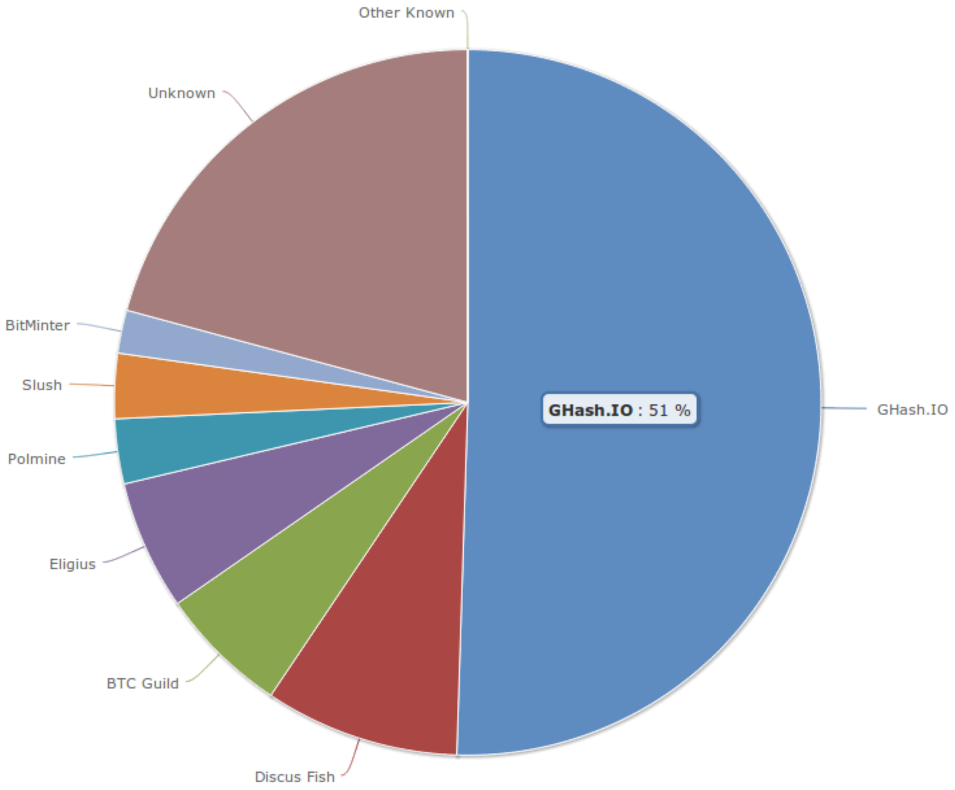
The mining pool is **operated centrally**.

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

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

June 2014

Ghash.io got > 50% of the total hashpower.



Then this percentage went down…

Alternative cryptocurrencies

a) **Litecoin** – a currency where hardware mining is (supposedly) harder

b) **Spacecoin** – a currency based on the Proofs of Space

c) Currencies based on the **Proofs of Stake**

d) Currencies doing **some useful work** (Primecoin, Permacoin)

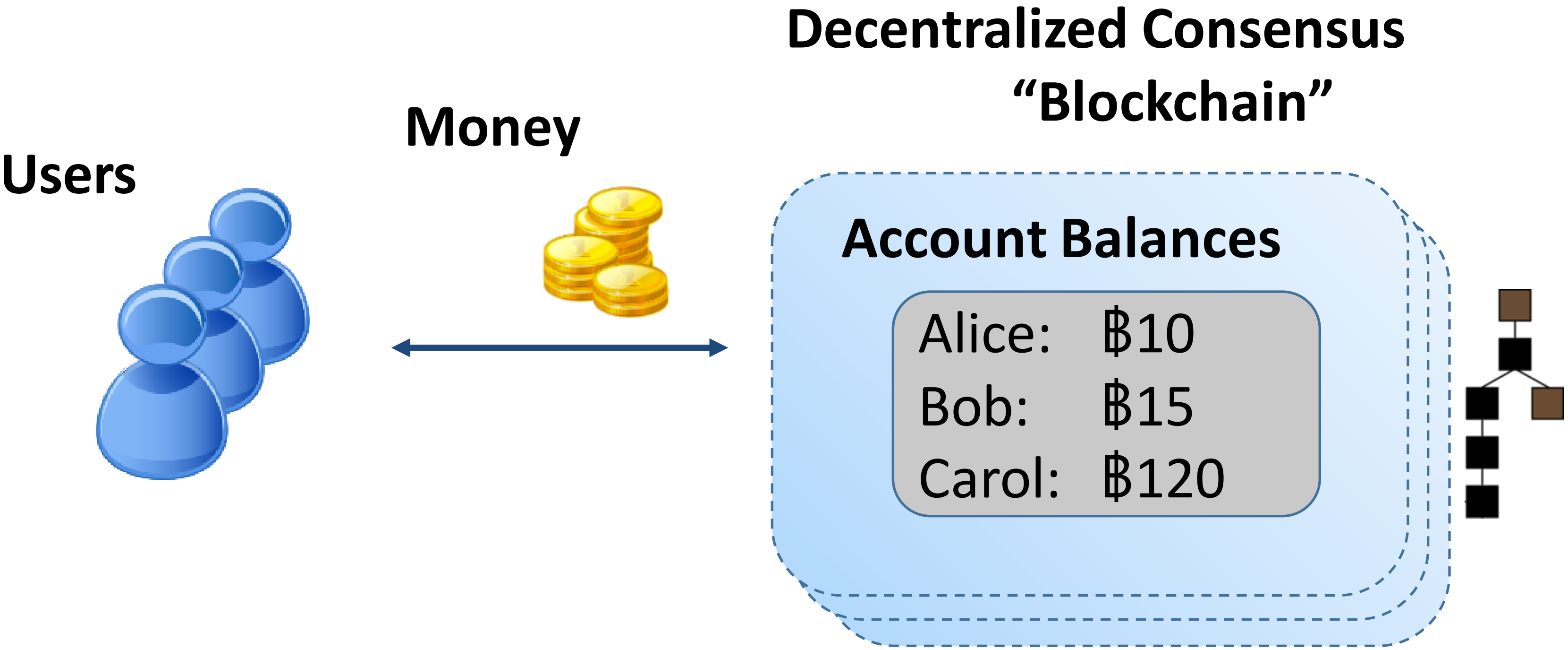
e) **Zerocash** – a currency with true anonymity

f) **Ethereum** – a currency with Turing-complete scripts

g) Other uses of the Blockchain technology

Disclaimers: (a) some of them are just **academic proposals**, (b) this order is **not chronologic**.

Digital currency is just one application on top of a blockchain



Users

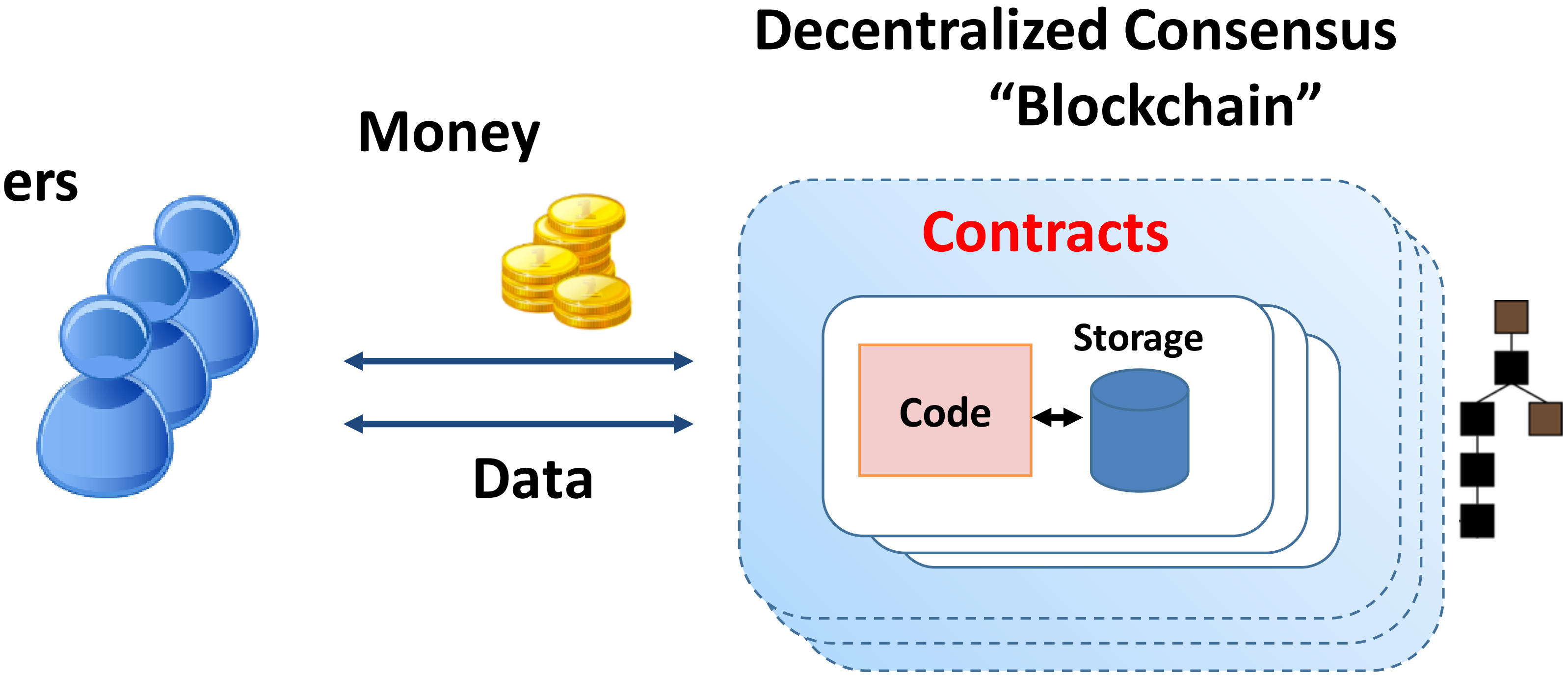
Money

Decentralized Consensus
“Blockchain”

Account Balances

Alice:	฿10
Bob:	฿15
Carol:	฿120

Smart Contracts: user-defined programs running on top of a blockchain



Smart Contract Example (very high level)

If GOOG rises to $1,000 by 30 June 2015, assign 10 shares from Alice to Bob and pay Alice $10,000

Other examples abound:

Auctions, elections, lotteries, escrow, ...

Zerocash

[Ben-Sasson, Chiesa, Garman, Green, Miers, Tromer, Virza, *Zerocash: Decentralized Anonymous Payments from Bitcoin*, 2014]

Main idea:

instead of showing

“I spend some unspent transaction Tx from the past”

show:

“There exists an unspent transaction Tx from the past that I now spend”

This is done using zero-knowledge proofs (ZKP)

Technical challenge: how to do it without executing ZKPs on the entire blockchain?

Acknowledgement

Some of the slides and slide contents are taken from
http://www.crypto.edu.pl/Dziembowski/teaching
and fall under the following:

We have also used slides from Prof. Dan Boneh online cryptography course at Stanford University:

http://crypto.stanford.edu/~dabo/courses/OnlineCrypto/