



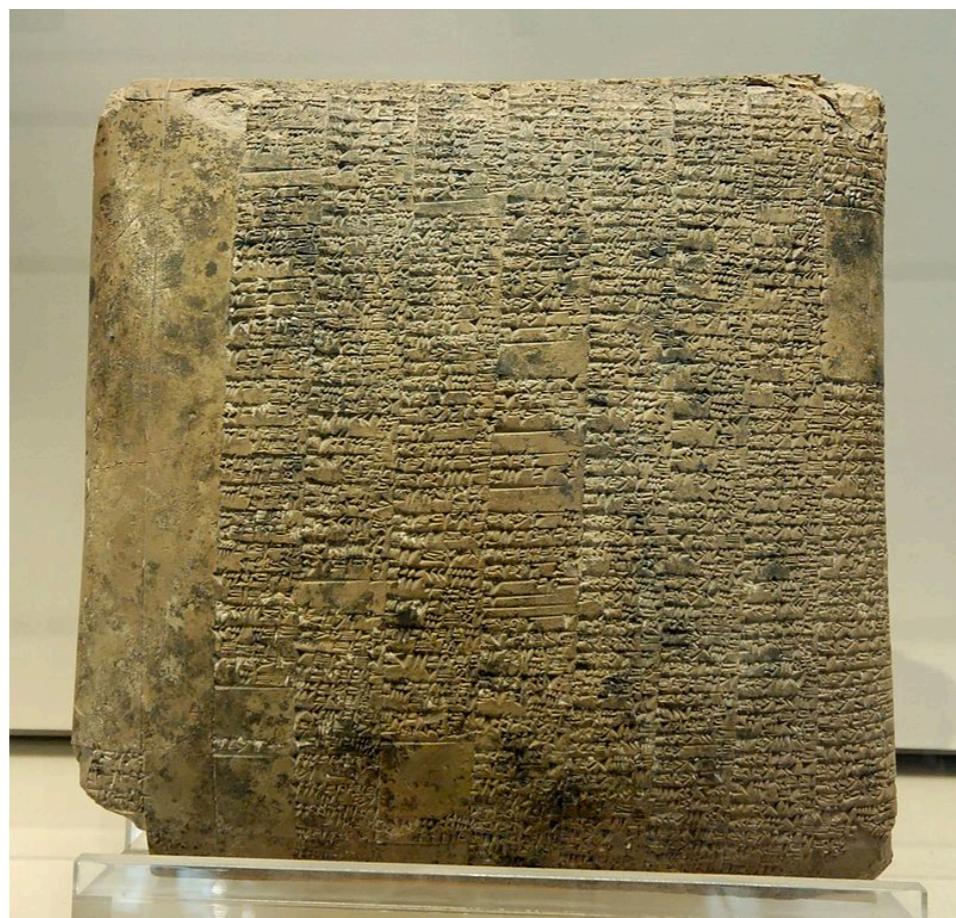
# Principles of Distributed Ledgers

Arthur Gervais



# Introduction

# 2040 BC



Balance sheet - Mesopotamia

Balance sheet



Mesopotamia



HSBC



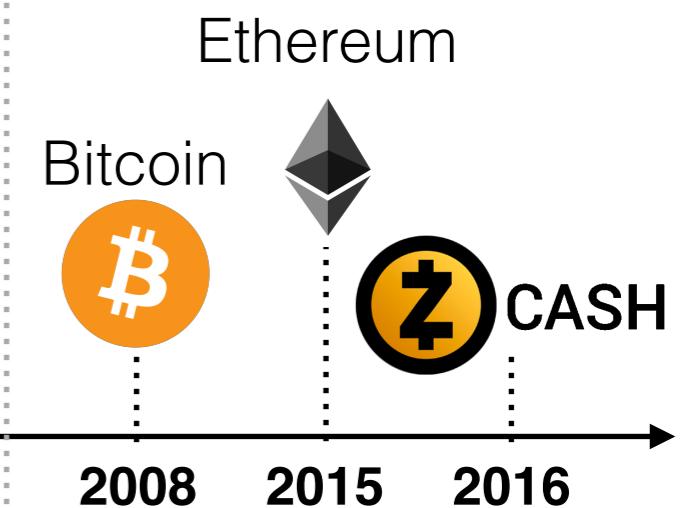
Turing

VISA

ECash



Chaum

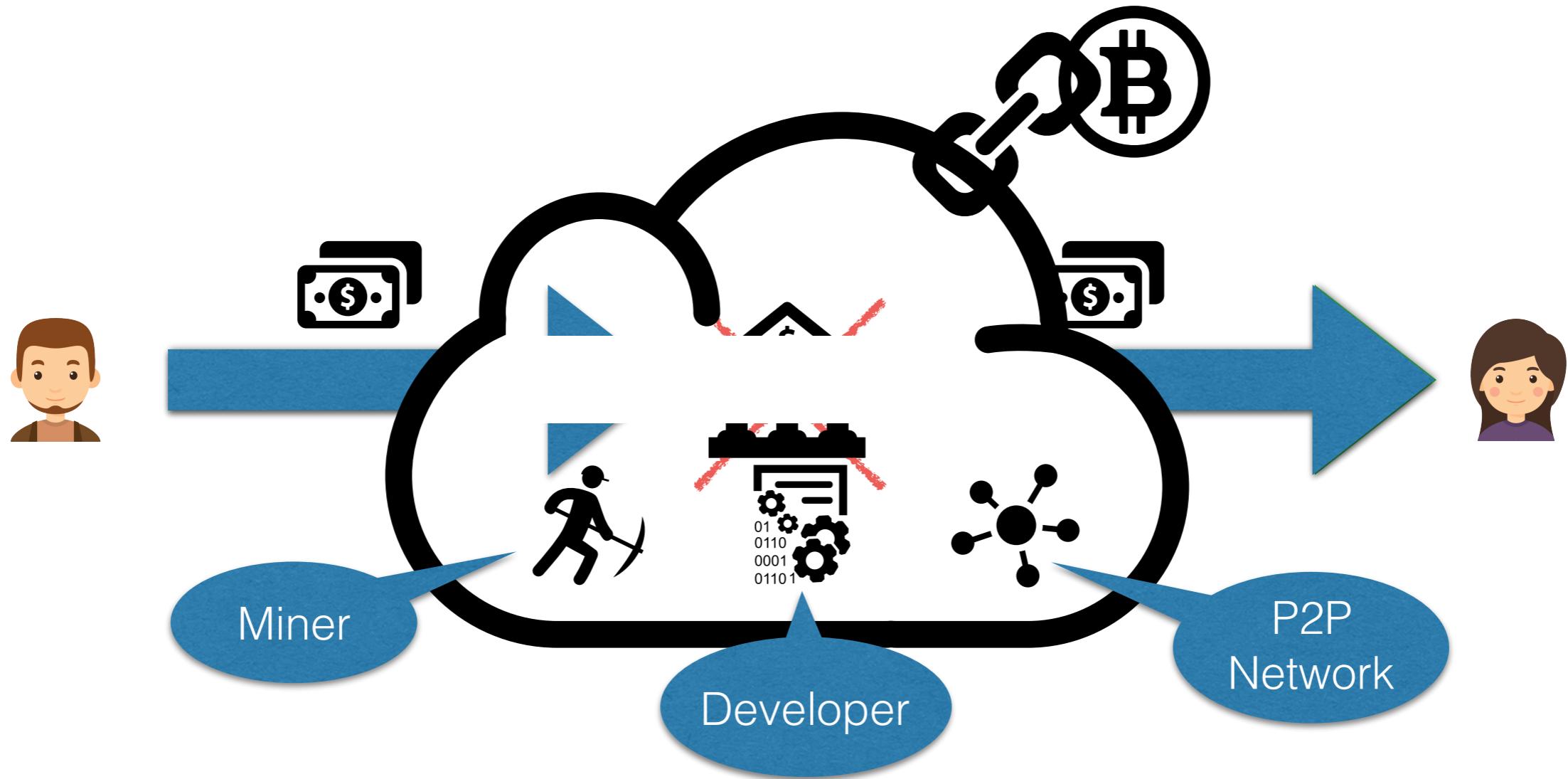


Centralized  
Banking

Privacy-Preserving  
Banking

Decentralized  
Banking

# From Centralized to Decentralized Payment Systems



How to perform secure **decentralized** payments?



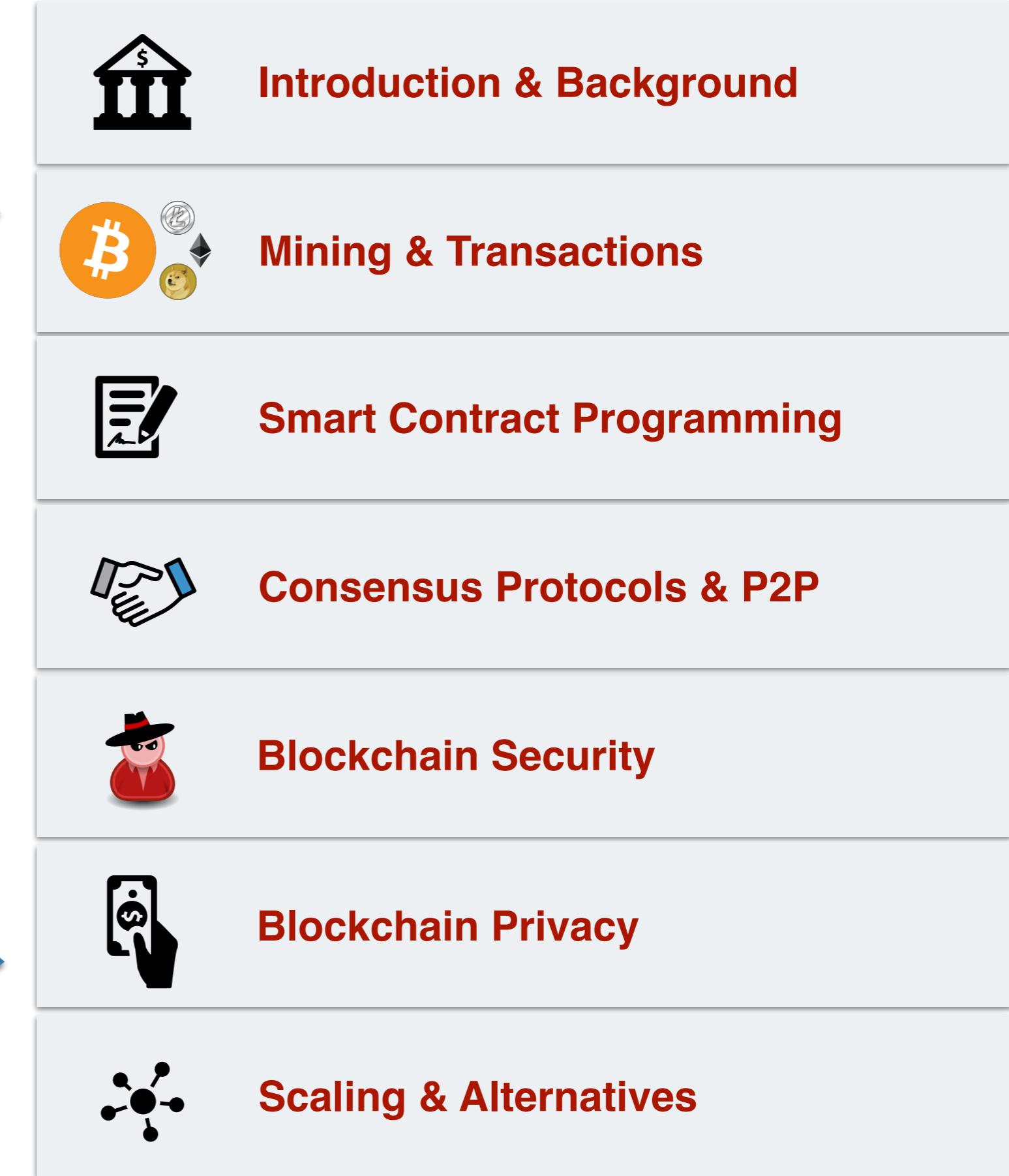
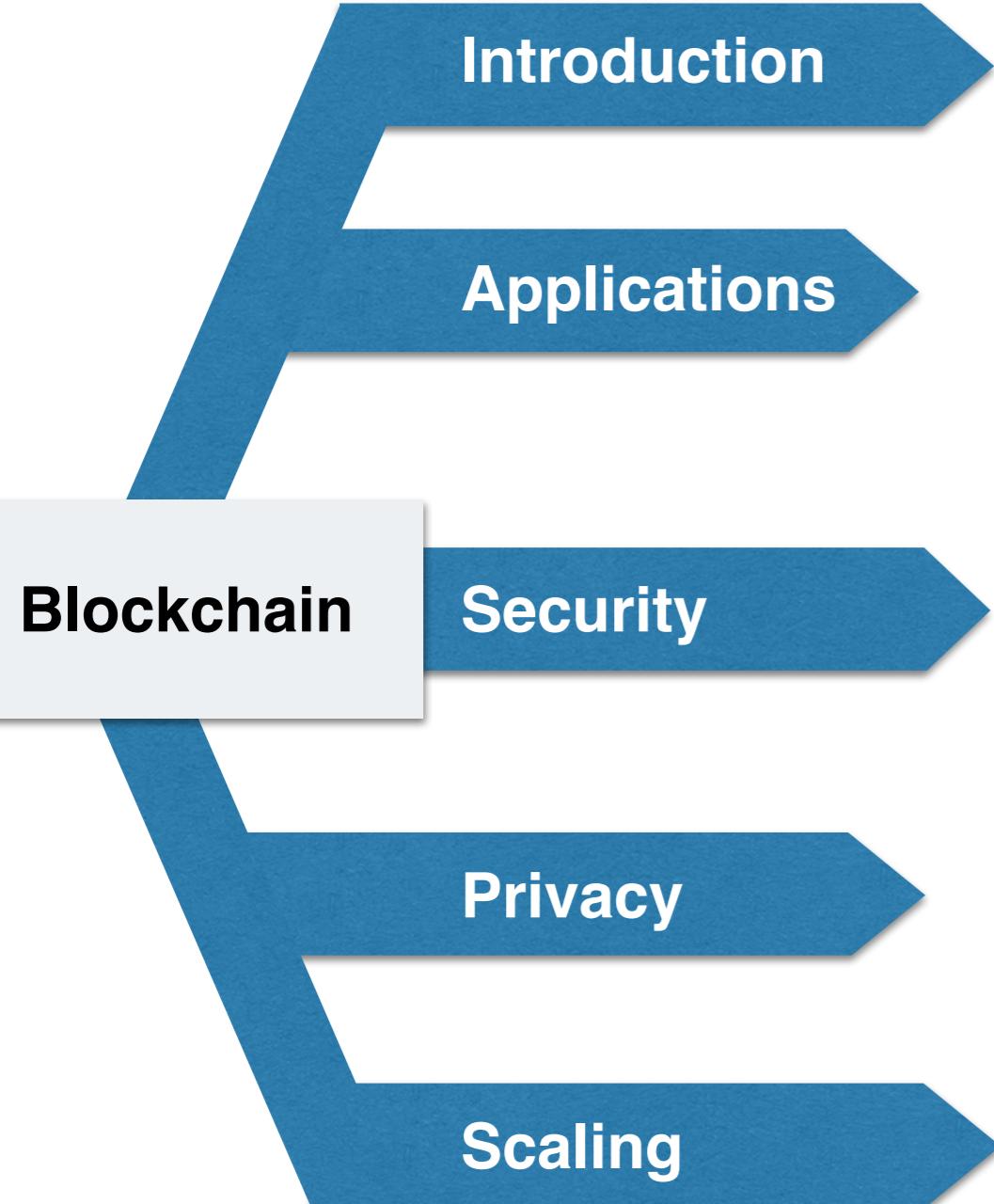
How to exchange **privacy-preserving** payments?



How to make decentralized systems **efficient**?



# Course Outline



## Course Goals

- Understanding the foundational principles of distributed ledgers (e.g., how authenticated data structures allow to protect integrity)
- Being able to program secure decentralized applications and to independently start research projects related to distributed ledgers.
- Understanding and developing new security attacks and possible defences in distributed ledgers.

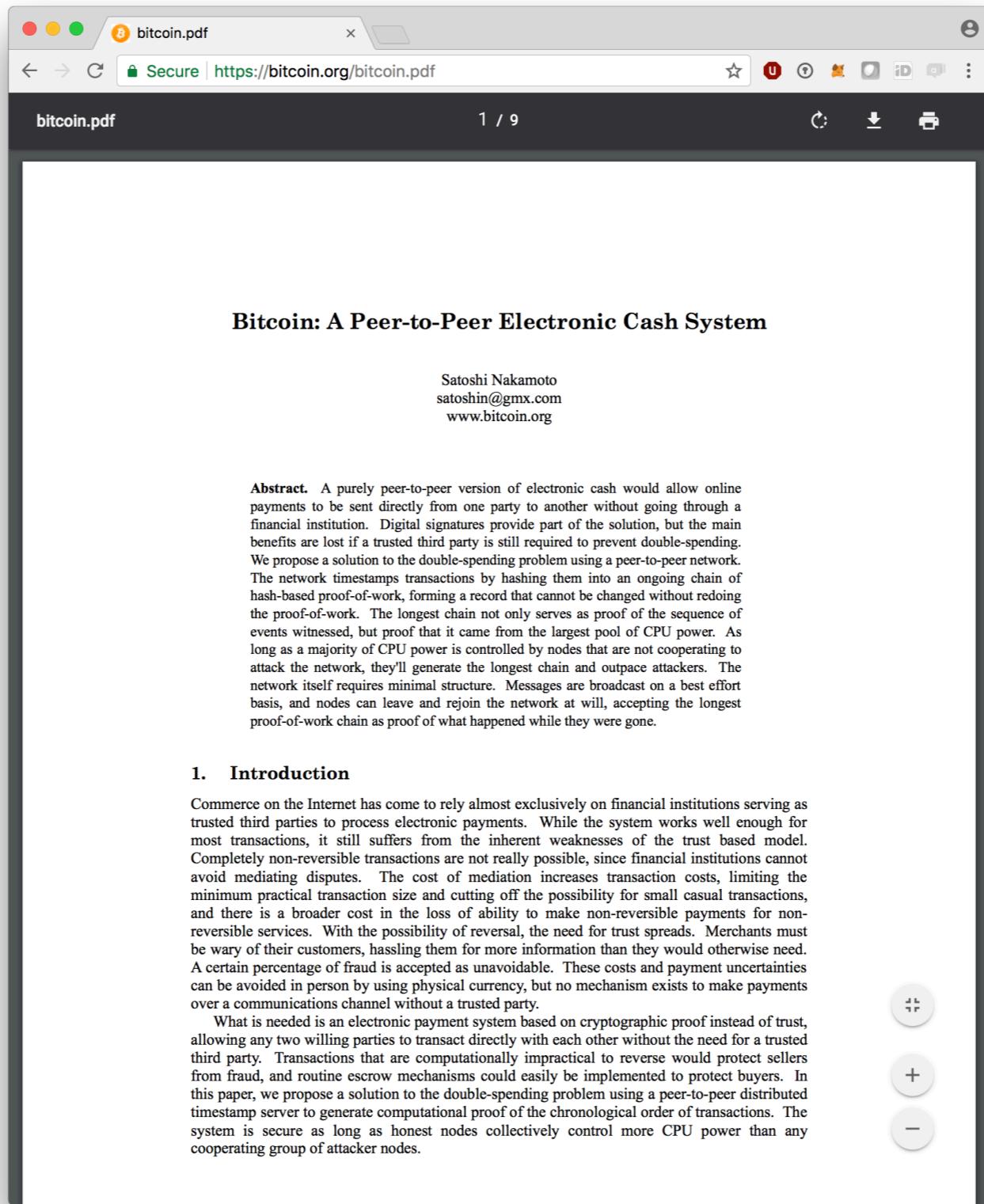
## Resources

- Piazza for discussions and paper reading  
[piazza.com/imperial.ac.uk/spring2019/467/home](https://piazza.com/imperial.ac.uk/spring2019/467/home)
- Bitcoin and Cryptocurrency Technologies  
[https://lopp.net/pdf/princeton\\_bitcoin\\_book.pdf](https://lopp.net/pdf/princeton_bitcoin_book.pdf)
- Mentimeter  
For questions and live interactions
- GTA  
Sam Werner, Dominik Harz, Rami Khalil, Alexei Zamyatin, Lewis Gudgeon

## Paper Reading

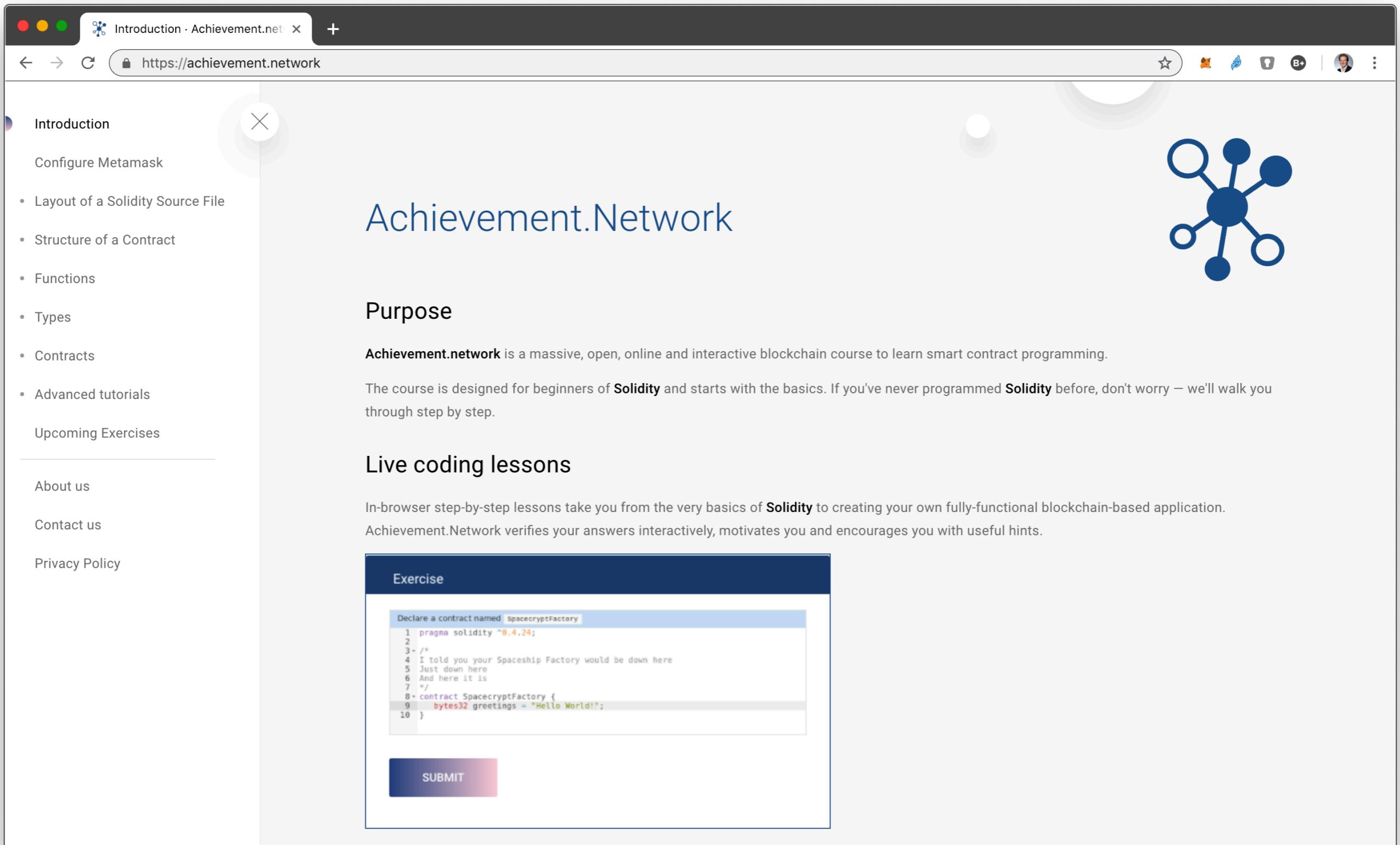
- We will average 1 paper per week
- Reading list is provided on Piazza
- Everybody must have read the paper. I'll choose 1 random student to present it during 5 minutes (without slides), graded
- Submit 1 unique question to Piazza about the paper, graded

# Paper Reading



- <https://bitcoin.org/bitcoin.pdf>

# CW1 and 3: Create an exercise on Achievement.Network



The screenshot shows a web browser window for 'Introduction · Achievement.net'. The page title is 'Achievement.Network' with a subtitle 'Purpose'. On the left, a sidebar lists navigation items: 'Introduction' (selected), 'Configure Metamask', 'Layout of a Solidity Source File', 'Structure of a Contract', 'Functions', 'Types', 'Contracts', 'Advanced tutorials', 'Upcoming Exercises', 'About us', 'Contact us', and 'Privacy Policy'. A blue network icon is on the right. The main content area displays a code editor titled 'Exercise' containing Solidity code for a 'SpacecryptFactory' contract. The code includes a comment about a spaceship factory and a 'greetings' function. A 'SUBMIT' button is at the bottom.

```
Exercise

Declare a contract named SpacecryptFactory
1 pragma solidity ^0.4.24;
2
3 /*
4 I told you your Spaceship Factory would be down here
5 Just down here
6 And here it is
7 */
8 contract SpacecryptFactory {
9     bytes32 greetings = "Hello World!";
10 }
```

# CW1 and 3: Create an exercise on Achievement.Network

The screenshot shows a web browser window with the title 'How to create exercises · Achiev' and the URL 'https://achievement.network/HowTo/createExercises.html'. The page content is as follows:

**How to create exercises**

**Introduction**

**Disclaimer:** This page will be moved to its own dedicated documentation in the future, along with a public git repo as a tutorial

**Introduction**

You've finished all exercises on the website and are wondering what I have for you next. Guess what? I will present what's behind the scenes. The whole tutorial you've just completed is written in [Markdown](#). Because this documentation is not aimed at learning you how to markdown, I won't spend time explaining all this language details, while it should still be understandable.

**Pro tip:** To better follow this tutorial, create an empty file on your machine and update it as you go through this webpage

**Writing text**

When you are reading a book or a website, you usually don't interact with it. You are reading text. As a content creator, writing text is quite easy. For example, this page you're viewing is written in Markdown. Like this:

```
# How to create exercises
> **Disclaimer**: This page will be moved to its own dedicated documentation in the future

## Introduction

You've finished all exercises on the website
```

## CW2: Smart Contract Programming Assignment

- Implement a smart contract on Ethereum that can be used by different users
- The users should not trust each other
- Choose your own or one from the suggested list
- Implement and test using [ethfiddle.com](http://ethfiddle.com) or/and the pyethereum simulator
- Write a report about the project

## CW4: Write a scalable Blockchain Application

- Choose a use-case
- Choose a technology to scale
  - 2-party channels
  - Commit chains
- Write a report



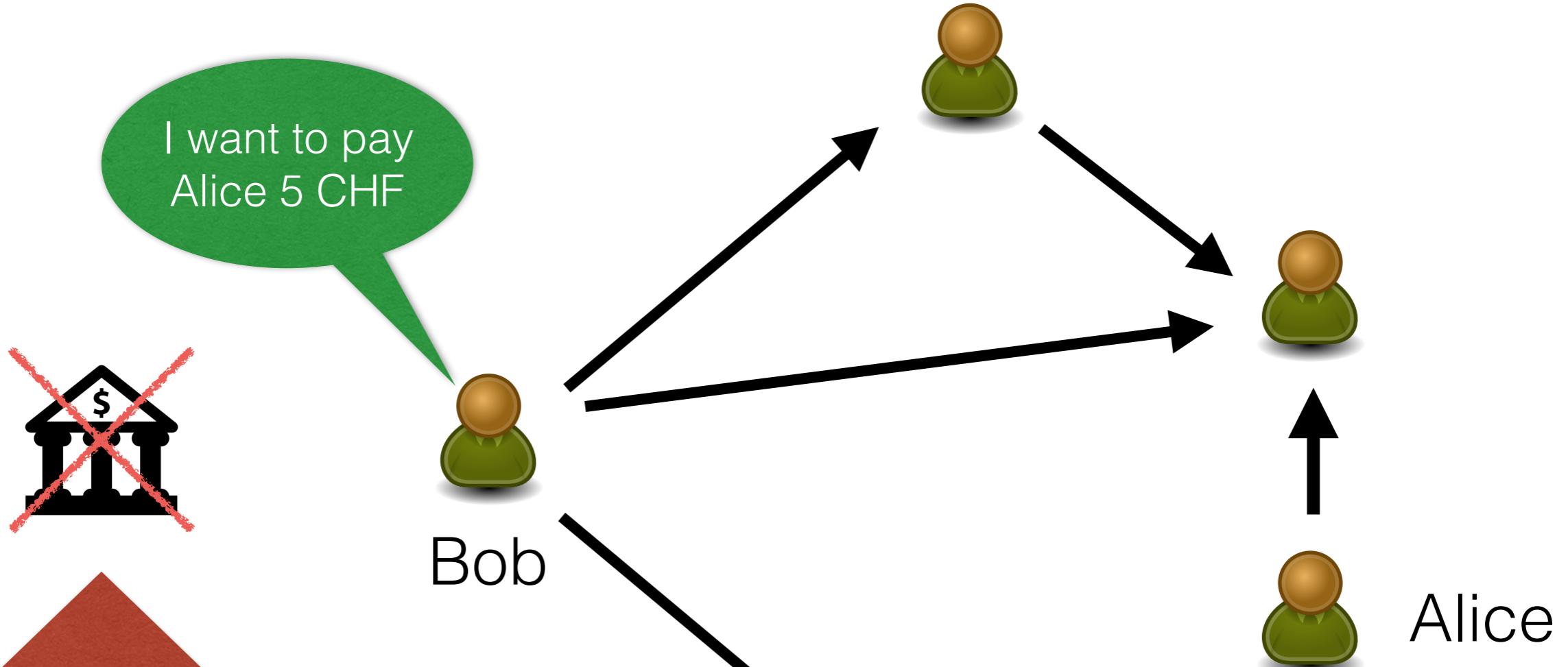
# Blockchain Background

# Digital Payment Systems

- With a centralized entity
  - Banks
  - Digital Signatures are the main double-spending resistance mechanism
- Examples
  - Pepper Micropayments [Rivest]
  - ECash [David Chaum] (privacy preserving)



# Decentralized Digital Payments



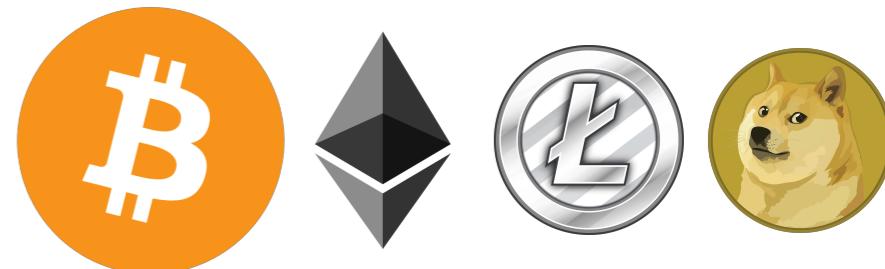
**Every  
node should  
have the same  
view of the  
ledger**

| No.    | Cash   | No.    | Cash           |
|--------|--------|--------|----------------|
| (1)    | 10,000 | (5)    | 500            |
| (6)    | 500    | (8)    | 200            |
| (12)   | 2,000  | (9)    | 500            |
| (14)   | 25,000 | (10)   | 500            |
|        |        | (11)   | 1,000          |
|        |        | (13)   | 1,000          |
|        |        | (15)   | 1,000          |
| Totals |        | 37,500 | Totals (4,700) |

# Blockchain != Blockchain

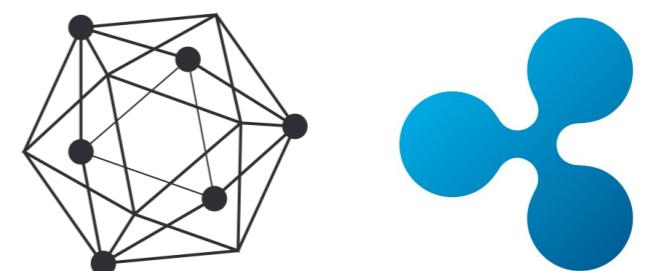
## Open and Decentralised Blockchains

- Ethereum
- Bitcoin



## Permission-based Blockchains

- Hyperledger
- Ripple
- Stellar



# Open and Decentralised Blockchains



# Bitcoin

- First introduced in 2009
- Pseudonymous
- Peer-to-peer
- No trusted third party
- Blockchain
- ▶ Transactions
- ▶ Blocks

The screenshot shows a web browser displaying a research paper from the International Association for Cryptologic Research (IACR). The title of the paper is "Is Bitcoin a Decentralized Currency?" by Arthur Gervais\*, Ghassan O. Karam\*\* Srdjan Capkun\* Vedran Capkun\*\*\*. The authors are affiliated with ETH Zurich, NEC Laboratories Europe, and HEC Paris. The paper's abstract discusses the decentralized nature of Bitcoin, noting that while it promises decentralization, recent incidents and observations reveal significant centralization. It highlights that a limited set of entities control vital operations like mining and incident resolution. The paper also explores ways to enhance decentralization. The keywords listed are Bitcoin and Decentralized decision process.

Is Bitcoin a Decentralized Currency?

Arthur Gervais\* Ghassan O. Karam\*\* Srdjan Capkun\*  
Vedran Capkun\*\*\*  
\*ETH Zurich, 8092 Zuerich, Switzerland.  
\*\*NEC Laboratories Europe, 69115 Heidelberg, Germany.  
\*\*\*HEC Paris, France.

**Abstract**

Bitcoin has achieved large-scale acceptance and popularity by promising its users a fully decentralized and low-cost virtual currency system. However, recent incidents and observations are revealing the true limits of decentralization in the Bitcoin system. In this article, we show that the vital operations and decisions that Bitcoin is currently undertaking are not decentralized. More specifically, we show that a limited set of entities currently control the services, decision making, mining, and the incident resolution processes in Bitcoin. We also show that third-party entities can unilaterally decide to “devalue” any specific set of Bitcoin addresses pertaining to any entity participating in the system. Finally, we explore possible avenues to enhance the decentralization in the Bitcoin system.

**Keywords:** Bitcoin, Decentralized decision process

## 1 Introduction

Bitcoin has witnessed a wider adoption and attention than any other digital currency proposed to date. One reason for such a broad adoption of Bitcoin has been a promise of a low-cost and decentralized currency that is inherently independent of governments and of any centralized authority [1]. In this work, we analyze the (de-)centralized nature of Bitcoin and show that—contrary to widespread belief—Bitcoin is not a truly decentralized system as it is deployed and implemented today.

Namely, in Bitcoin, the users “vote” with their computing power to prevent double-spending (i.e., by *power-voting*) which effectively limits the power of individual users and makes Sybil attacks difficult. Given the huge computing power harnessed in the Bitcoin system (currently around 30,000 Tera hashes per second), users believe that it is unlikely for any entity to acquire such power alone. However, even a quick look at the distribution of computing power in Bitcoin reveals that the power of dedicated “miners” far exceeds the power that individual users dedicate to mining, allowing few parties to effectively control the currency; currently the top-three (centrally managed) mining pools control more than 50% of the computing power in Bitcoin. Indeed, while mining and block generation in Bitcoin was originally designed to be decentralized, these processes are currently largely centralized.

On the other hand, other Bitcoin operations, like protocol updates and incident resolution are not designed to be decentralized, and are controlled by a small number of administrators whose

1



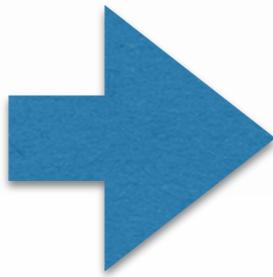
# Data Types



## • Cryptographic Hash Functions

- SHA256
- RIPEMD160

Arbitrarily long  
data



Fixed sized  
hash/digest



# Cryptographic Hash Function



- Cryptographic Hash Functions
  - Takes any byte sequence as input
  - Fixed size output
  - Efficiently computable
- Security Properties:
  - Collision-resistance
  - Second pre-image resistance
  - Pre-image resistance
  - Hiding
  - Puzzle-friendly

Example: <https://www.pelock.com/products/hash-calculator>



## Pre-image Resistance

- For any given  $h$  in the output space of the hash function, it is hard to find  $x$ , s.t.  $H(x)=h$

## Second Pre-image Resistance

- For a given message  $x$ , it is hard to find  $y$  s.t.  $x \neq y$  and  $H(x) = H(y)$

## Collision Resistance

- It is hard to find a pair of values,  $x \neq y$  and  $H(x) = H(y)$



## Hiding

- A hash function  $H$  is hiding when a secret value  $r$  is chosen from a high min-entropy probability distribution, then given  $H(r \parallel x)$ , it is hard to find  $x$ .

## Puzzle-friendly

- A hash function  $H$  is puzzle friendly if for every possible  $n$ -bit output value  $h$ , if  $k$  is chosen from a distribution with high min-entropy, then it is infeasible to find  $x$  such that  $H(k \parallel x) = h$  in time significantly less than  $2^n$ .



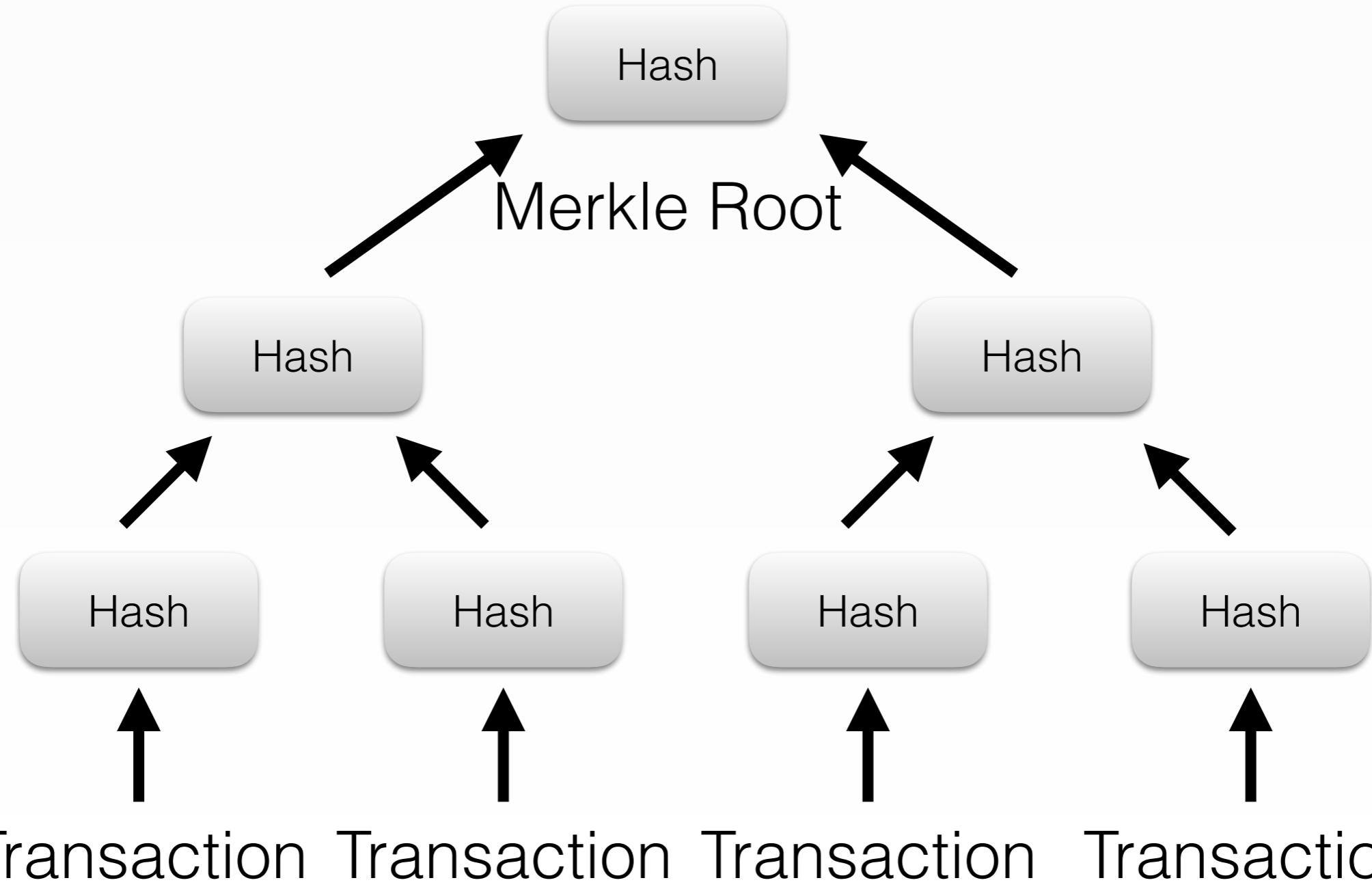
## Search puzzle

- A hash function  $H$
- A value,  $id$ , chosen from a high min-entropy distribution
- A target set  $Y$

A solution to the puzzle is a value  $x$ , s.t.

$$H(id||x) \in Y$$

# Data Types



## Data Types



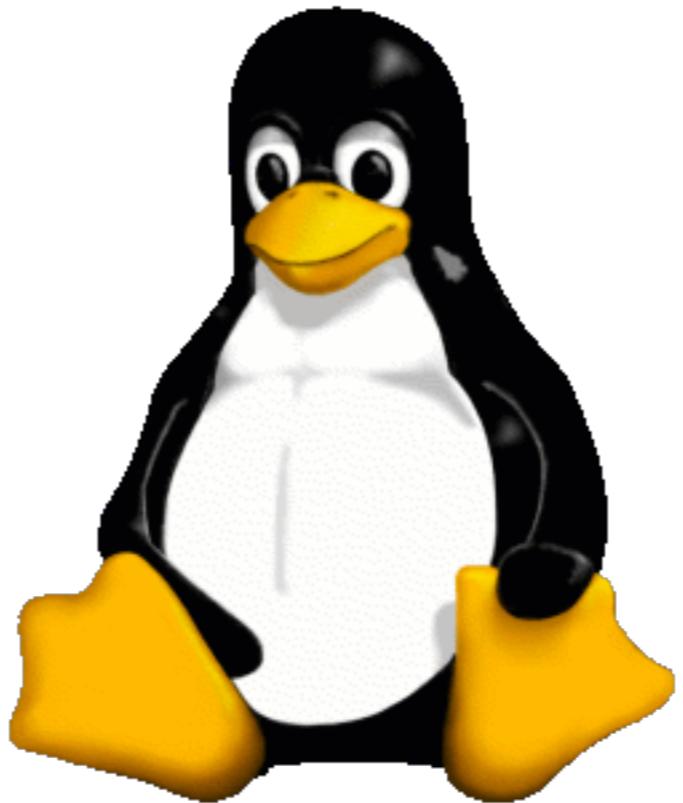
- ECDSA (secp256k1 curve) is used to
  - Sign transactions
  - Verify the signature of transactions
  - Nothing in Bitcoin is encrypted
- **Elliptic Curve Signature Algorithm (ECDSA)**



# Digital Signatures for Security



- <https://www.kernel.org/signature.html>



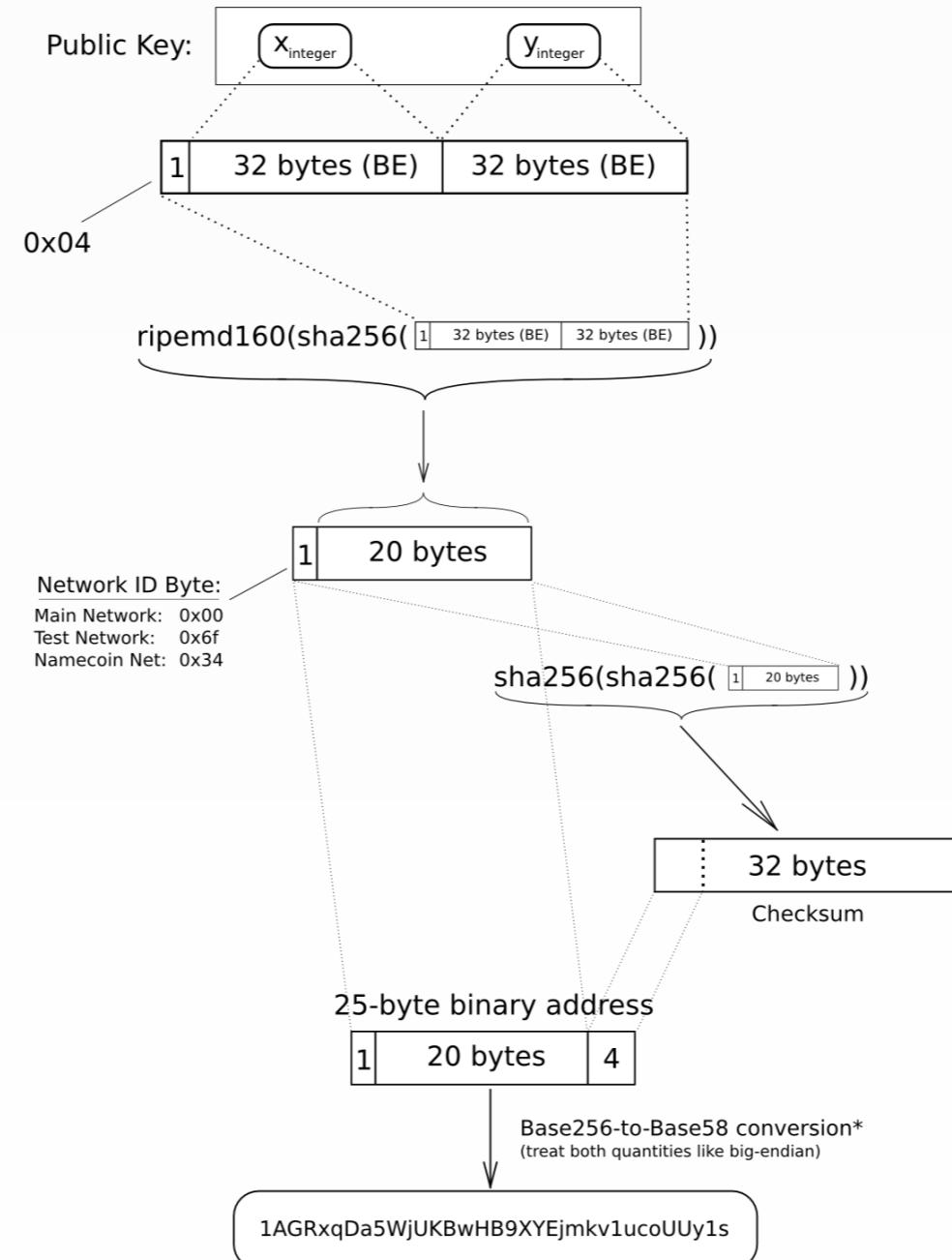
# Addresses

- Unique identifier
- Hash of a public key
- 0 < Balance
- 1 Satoshi

1EGal...

3J98t...

Elliptic-Curve Public Key to BTC Address conversion



etotheipi@gmail.com / 1Gffm7LKXcNPrtxy6yF4JBoe5rVka4sn1



1EGal...

@Eve

@ETH

4YTEr...

AVNLy...

# Transactions



# Transactions in Bitcoin



Alice



Transaction

Bob



Wallet



Wallet



Sender address

0.25 BTC

@Bob

Sender address

1 BTC

@Alice

0.75 BTC

@Alice

Recipient address

Change address

Inputs

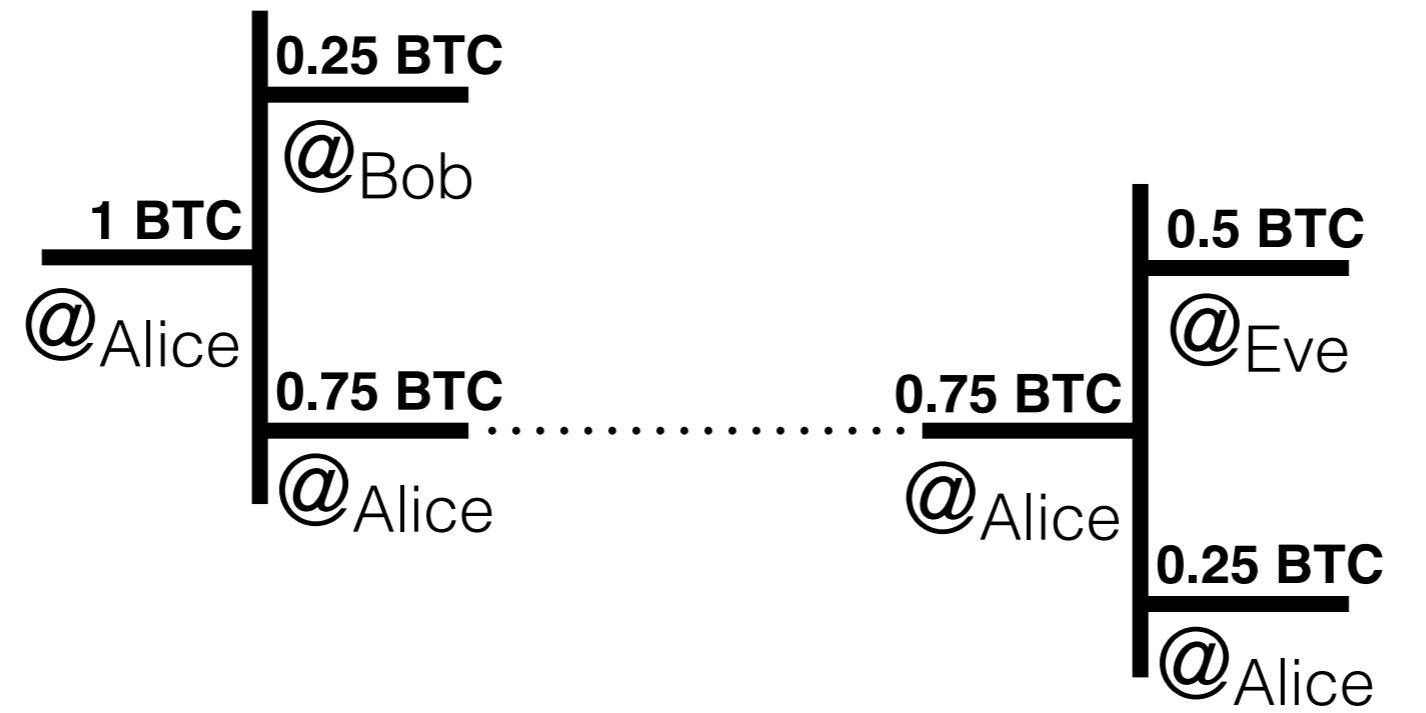
Outputs

Transaction Fees

$$\sum \text{inputs} \geq \sum \text{outputs}$$

Difference are fees

# Transactions in Bitcoin



Transaction 1

Transaction 2

# Script



- Stack based programming language
- If evals to *true* —> Bitcoin transaction is valid
- Many opcodes
- Execution time is critical to prevent DoS attacks

## Example Script

<signature><publicKey> OP\_CHECKSIG

Constants  
are pushed onto the  
stack

Operation  
executes on stack  
values

|                 |                |
|-----------------|----------------|
| <PubKey>        |                |
| <Sig>           | OP_DUP         |
| Execution Stack | Execution Code |



|                 |                |
|-----------------|----------------|
| <PubKeyHash>    |                |
| <PubKeyHash>    |                |
| <PubKey>        |                |
| <Sig>           | OP_EQUALVERIFY |
| Execution Stack | Execution Code |



|                 |                |
|-----------------|----------------|
| <PubKey>        |                |
| <PubKey>        |                |
| <Sig>           | OP_HASH160     |
| Execution Stack | Execution Code |



|                 |                |
|-----------------|----------------|
| <PubKey>        |                |
| <Sig>           | OP_CHECKSIG    |
| Execution Stack | Execution Code |

Constants are pushed onto the stack

<Sig> <PubKey> OP\_DUP OP\_HASH160 <PubKeyHash> OP\_EQUALVERIFY OP\_CHECKSIG

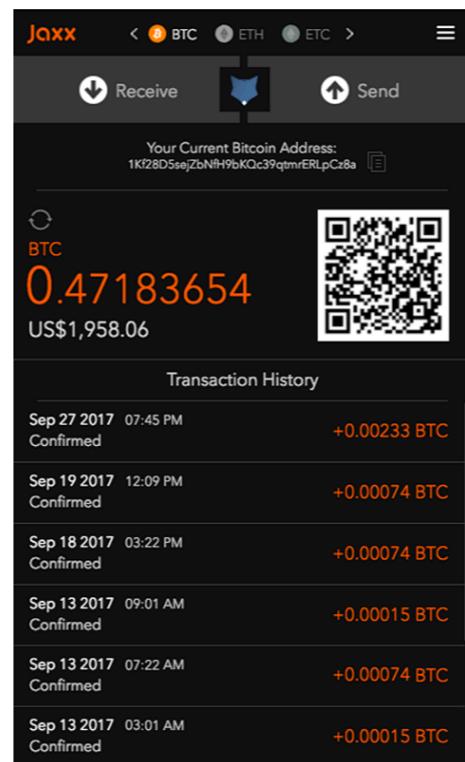
# Transaction Types



- P2PKH - Pay to Public Key Hash
  - Redeemer needs a public key and signature
- P2SH - Pay to Script Hash
  - Redeemer needs a script that matches a pre-defined hash
- Multisignature (m-n)
  - Requires multiples signatures to be redeemable
  - **m** out of **n** signatures required

# Blockchain Wallets

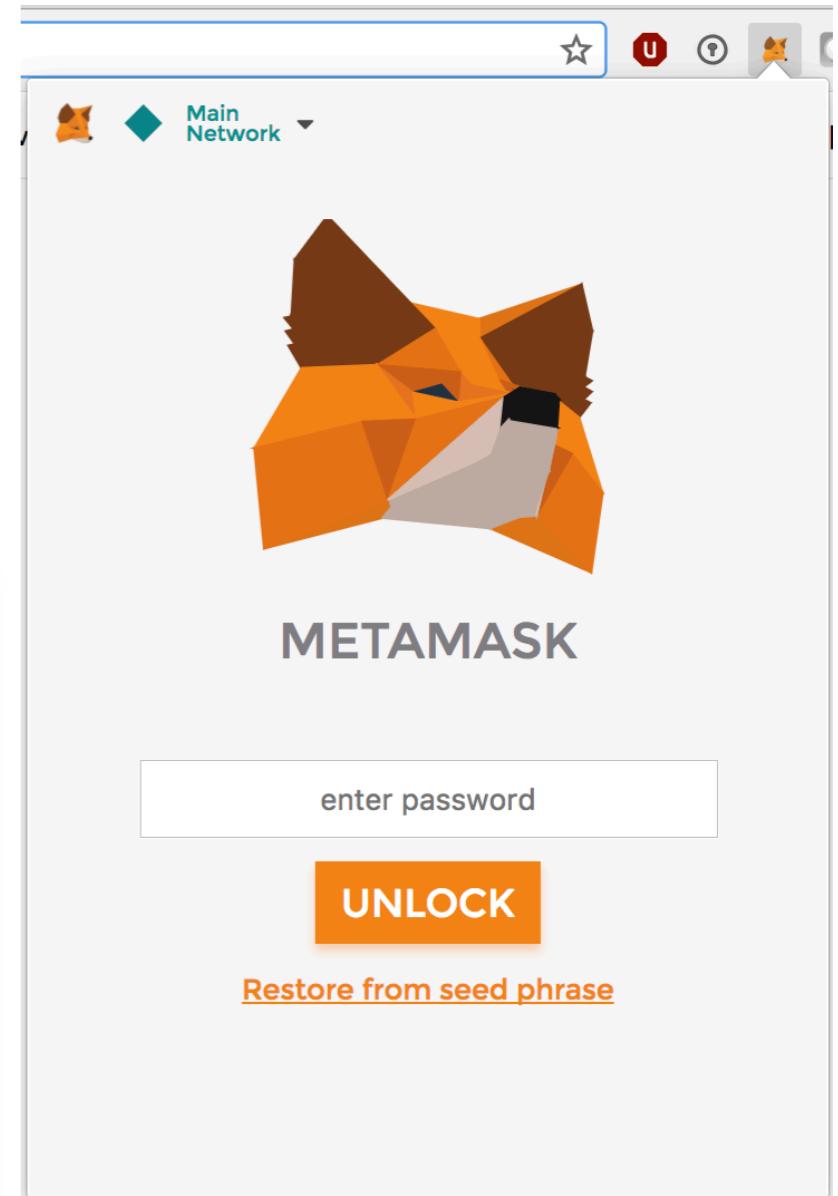
- Bitcoin Core  
Full node, downloads all transactions
- Bitcoin Wallet  
Android
- Jaxx  
Multi-chain Wallet
- Ledger Nano S



# Blockchain Wallets

- MetaMask
- MyEtherWallet

The screenshot shows the 'Create New Wallet' page of MyEtherWallet.com. At the top, there's a red banner with a warning: 'DON'T GET PHISHED, please! > Thank you!' followed by instructions: '1. BOOKMARK MYETHERWALLET.COM | 2. INSTALL EAL or MetaMask or Cryptonite'. Below the banner, the header includes the MyEtherWallet logo, version 3.11.2.3, language selection (English), gas price (41 Gwei), and network (ETH). A note says 'The network is really full right now. Check Eth Gas Station for gas price to use.' The main form has a title 'Create New Wallet' and a password input field with placeholder 'Enter a password'. Below it is a note: 'Do NOT forget to save this!' with a copy icon. A blue button says 'Create New Wallet'. To the right, a sidebar titled 'Already have a wallet somewhere?' lists several options: Ledger / TREZOR / Digital Bitbox (hardware wallet), MetaMask Connect (extension), Jaxx / imToken (Mnemonic Phrase), and Mist / Geth / Parity (Keystore File). At the bottom, a note states: 'This password encrypts your private key. This does not act as a seed to generate your keys. You will need this password + your private key to unlock your wallet.' Links 'How to Create a Wallet' and 'Getting Started' are at the very bottom.



## For next class..

- Create a wallet on MyEtherWallet
- Get test Ether for the Rinkeby network
- Send to address  
“0xD0bE22D14C3C7f4b754DBd1838b36fdd0E23c72”, 1 Rinkeby Test Ether and your student ID as data attached.

# Ethereum Testnet Transaction

The screenshot shows the MyEtherWallet.com interface for sending a transaction. The top navigation bar includes links for New Wallet, Send Ether & Tokens (which is active), Swap, Send Offline, Contracts, Check TX Status, View Wallet Info, and Help. The version is 3.11.2.3, the language is English, the gas price is 41 Gwei, and the network is set to Kovan (Etherscan.io). A banner at the top urges users to bookmark the site and install EAL or MetaMask.

**To Address:** 0x0D0bE22D14C3C7f4b754DBd1838b36fdd0E23c72

**Amount to Send:** 0.001 KOVAN ETH

**Gas Limit:** 21084

**Data:** 0x000000001

**Generate Transaction**

**Account Address:** 0x0D0bE22D14C3C7f4b754DBd1838b36fdd0E23c72

**Account Balance:** 0.57405 KOVAN ETH

**Transaction History:** KOVAN ETH (<https://kovan.etherscan.io>)

**Learn more about protecting your funds:** Ledger, TREZOR

**Token Balances:** How to See Your Tokens. You can also view your balances on <https://kovan.etherscan.io>.

Show All Tokens | Add Custom Token

## Checklist

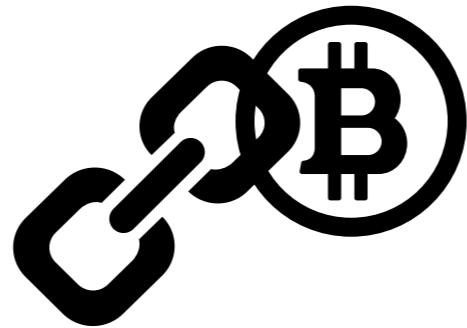
- Generate a keypair
- Give your “address” to someone
- Get someone else’s “address”
- Lookup the history of an “address”
- Use the transaction id/hash to look it up

## What can you do with Crypto?

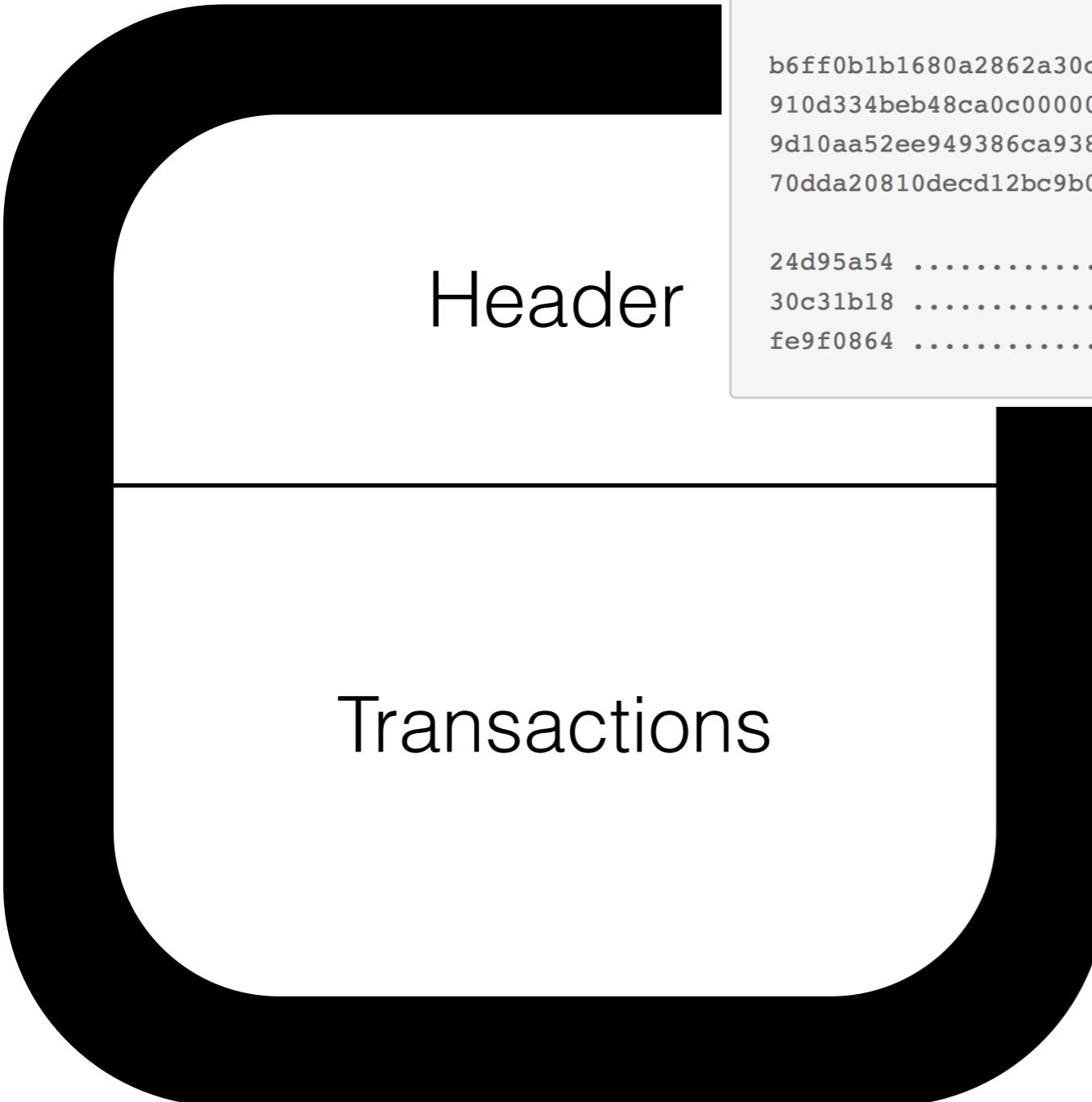
- Trade with other people
- Exchange to other crypto/FIAT
- Buy something online
- Donate
- Build applications/games

... and many more things!

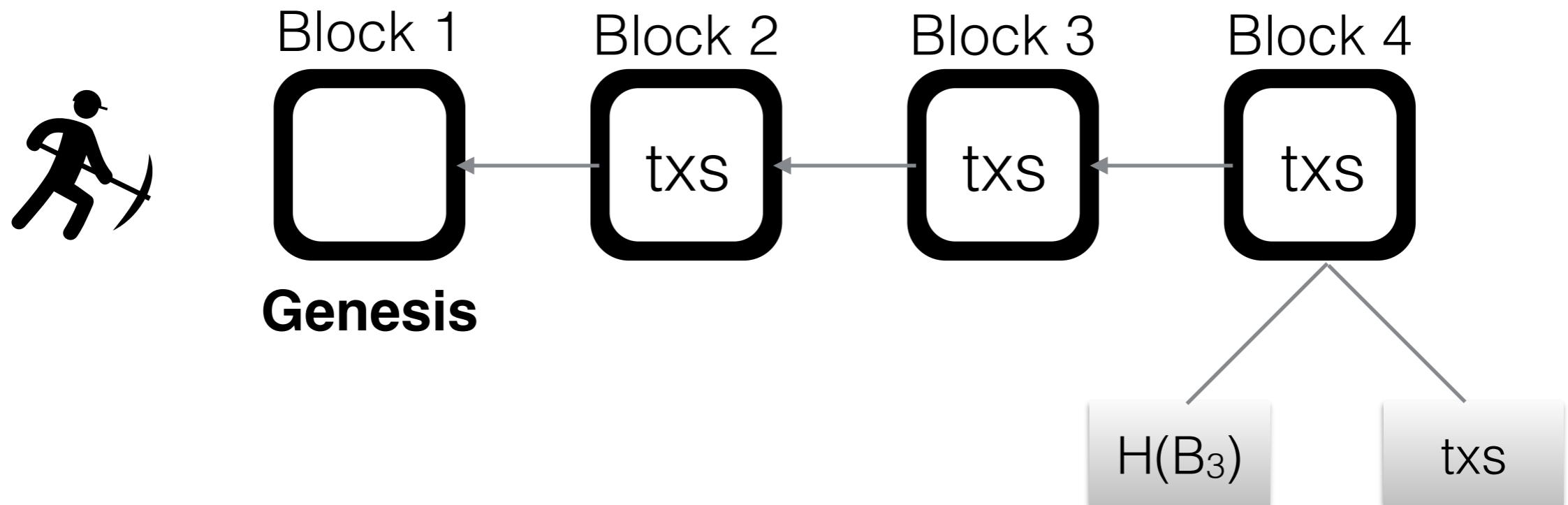
# Blockchain



# Block



# Blockchain

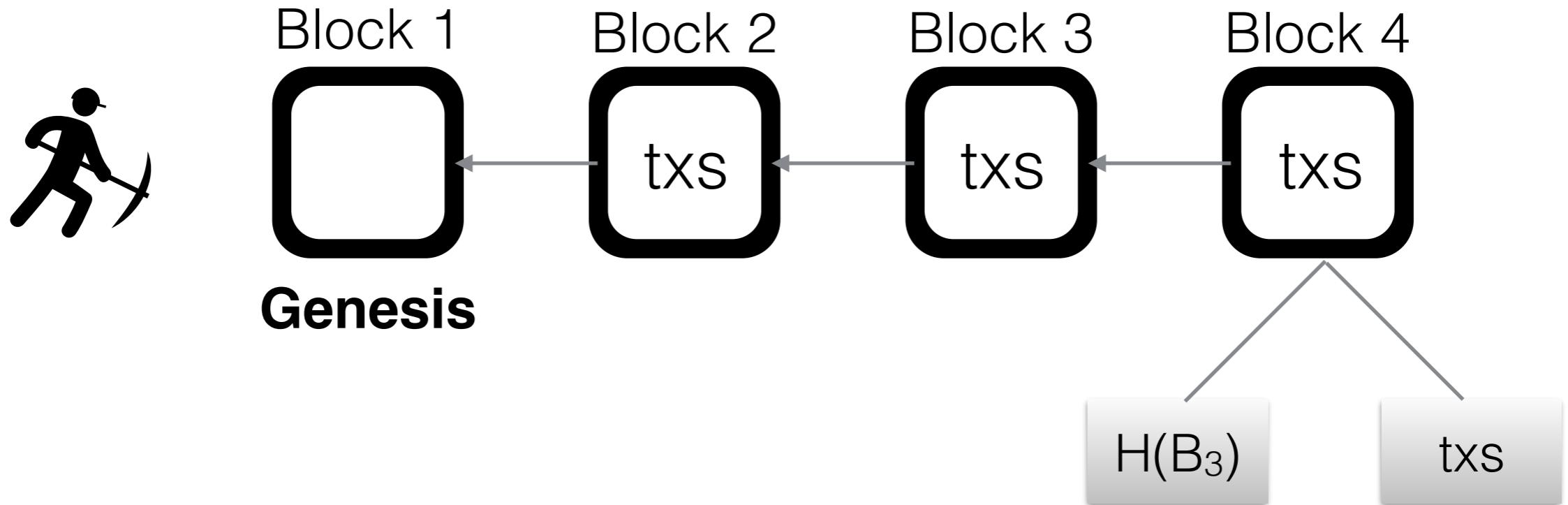


# How to do mining



1. Join the network, listen for transactions  
Validate all incoming transactions
2. Listen for new blocks, build on valid blocks
3. Construct a block template
4. Find a nonce that validates the new block
5. Tell all the other miners
6. Receive reward

# Blockchain



## Mining

- Find Nonce  $N$ , s.t.

$$\text{Hash}(\text{Hash}(B_3)|\text{txs}|N) < \text{target}$$

Best known approach: **Brute Force**

Controls the **difficult**

## Mining Difficulty

$\text{Hash}(\text{Hash}(B_3) | \text{txs} | N) < \text{target} = 0x000^{**}$

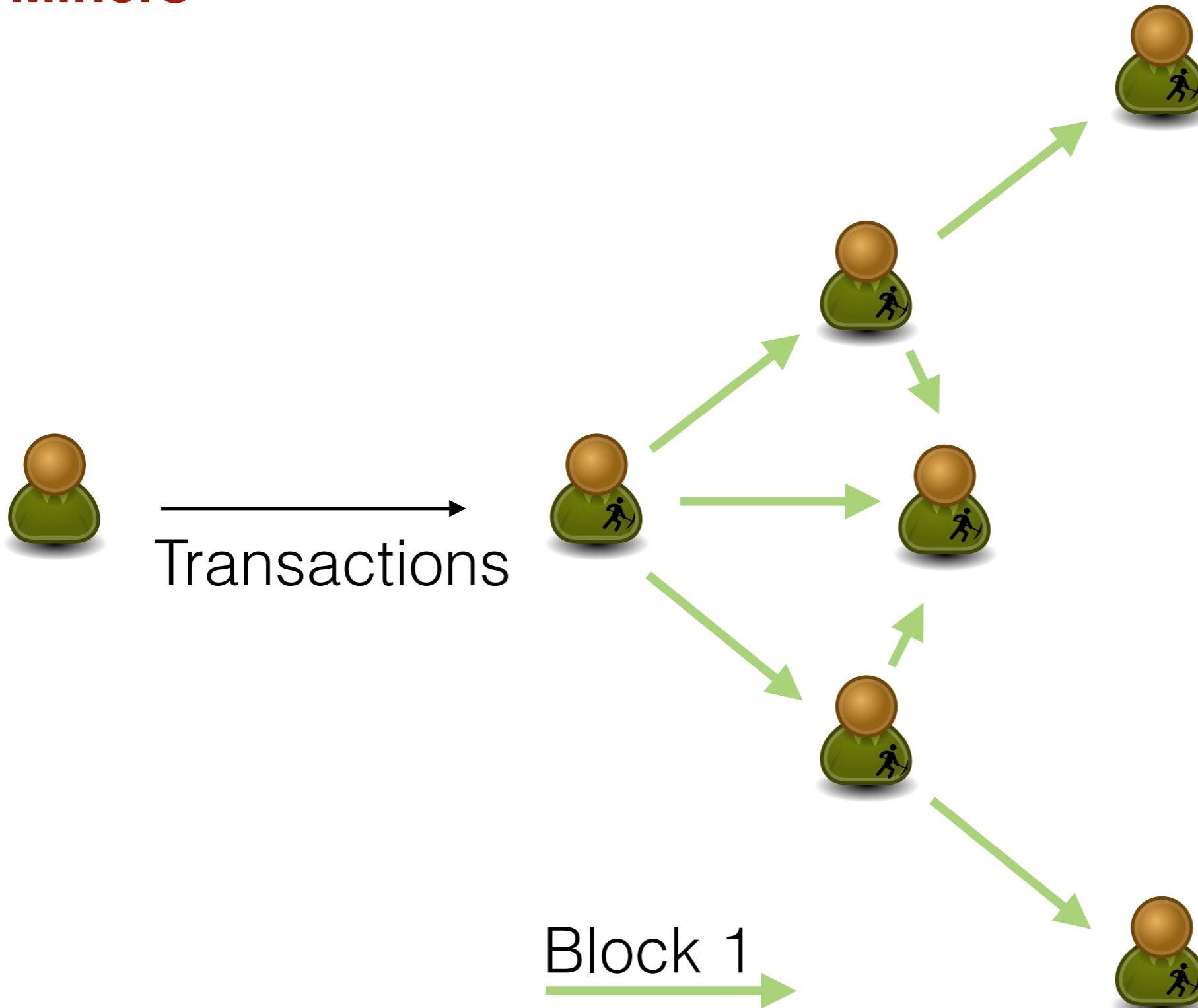
~~$\text{Hash}(\text{Block\_3} | \text{merkle\_root} | 0xbeed) = 0x03ef..$~~

~~$\text{Hash}(\text{Block\_3} | \text{merkle\_root} | 0xbeee) = 0x12ef..$~~

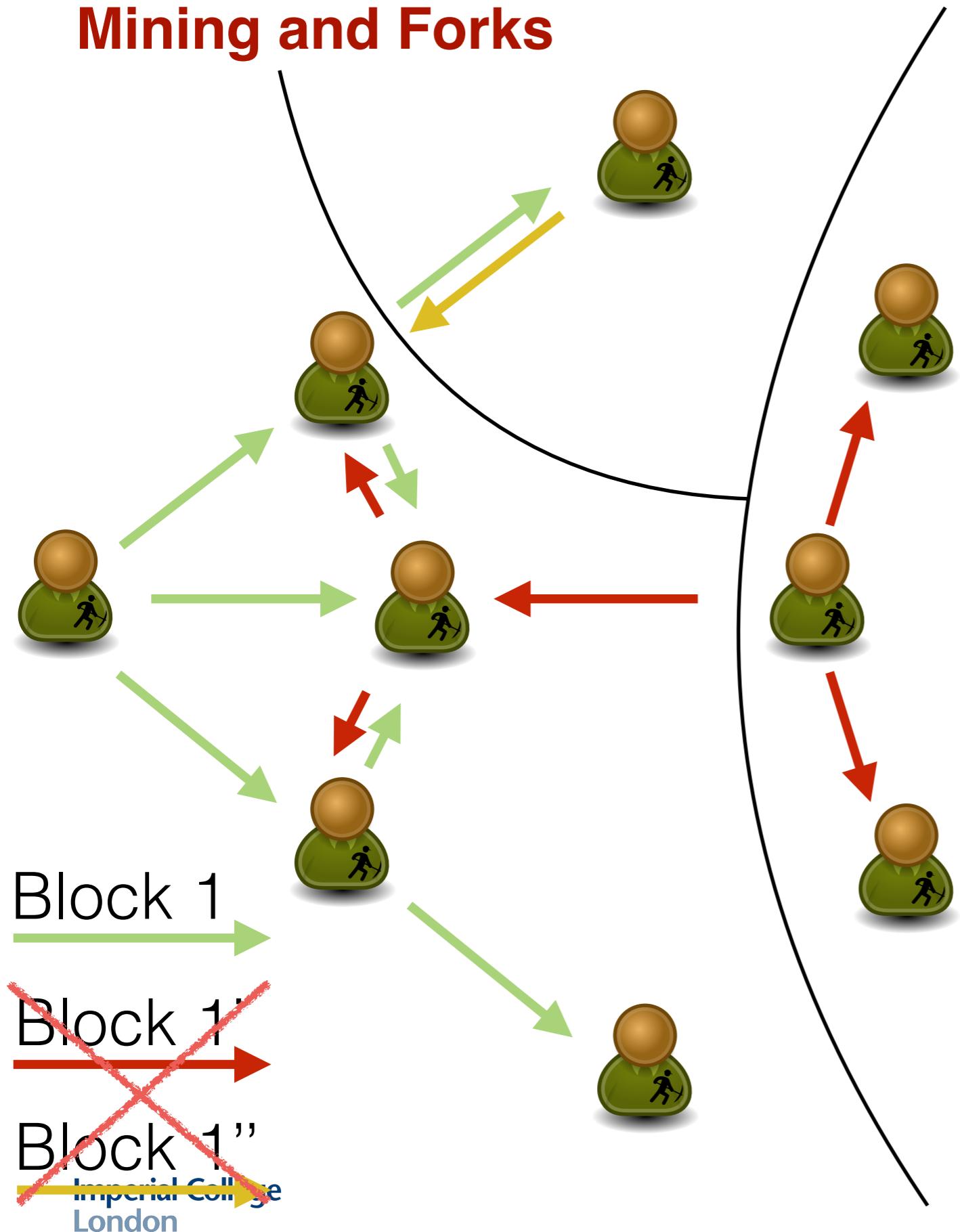
$\text{Hash}(\text{Block\_3} | \text{merkle\_root} | 0xbeef) = 0x000f..$



# Miners



## Mining and Forks



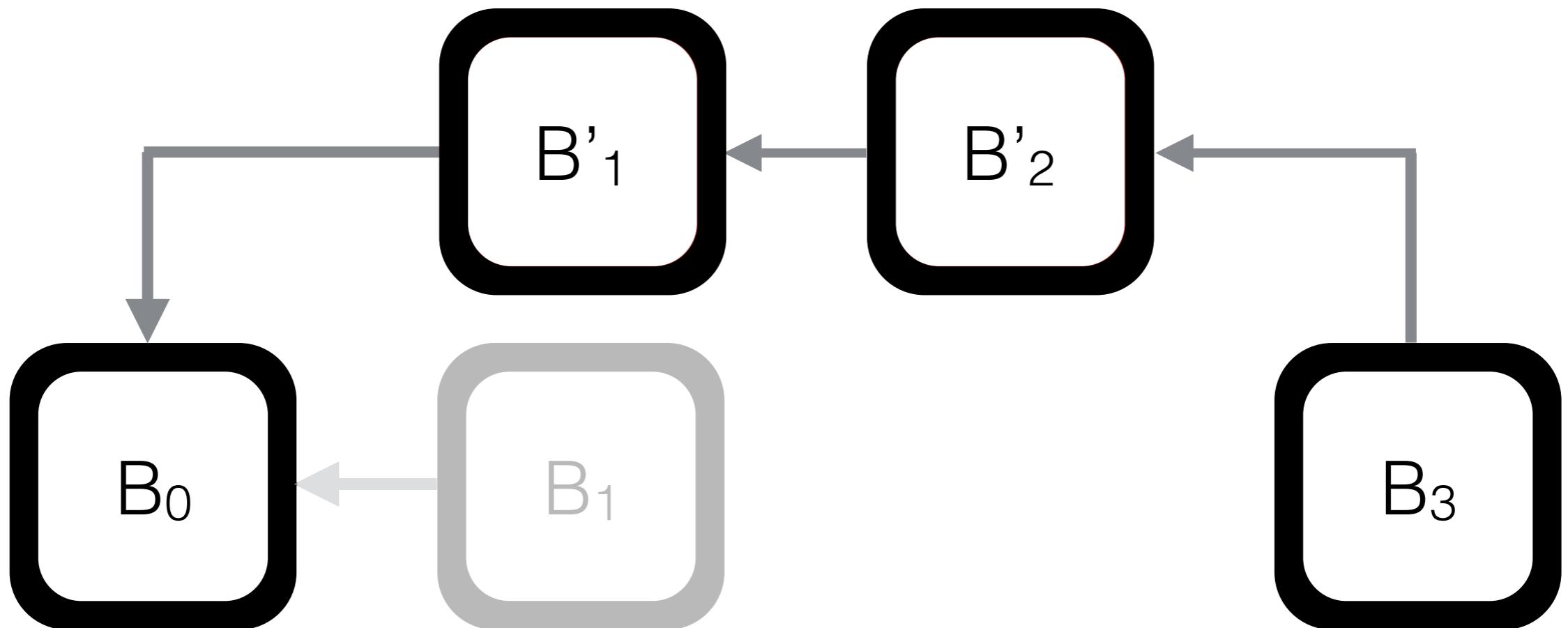
- Network partition
- **Stale blocks = lost efforts**

Selfish Mining

Denial of Service

Double Spending

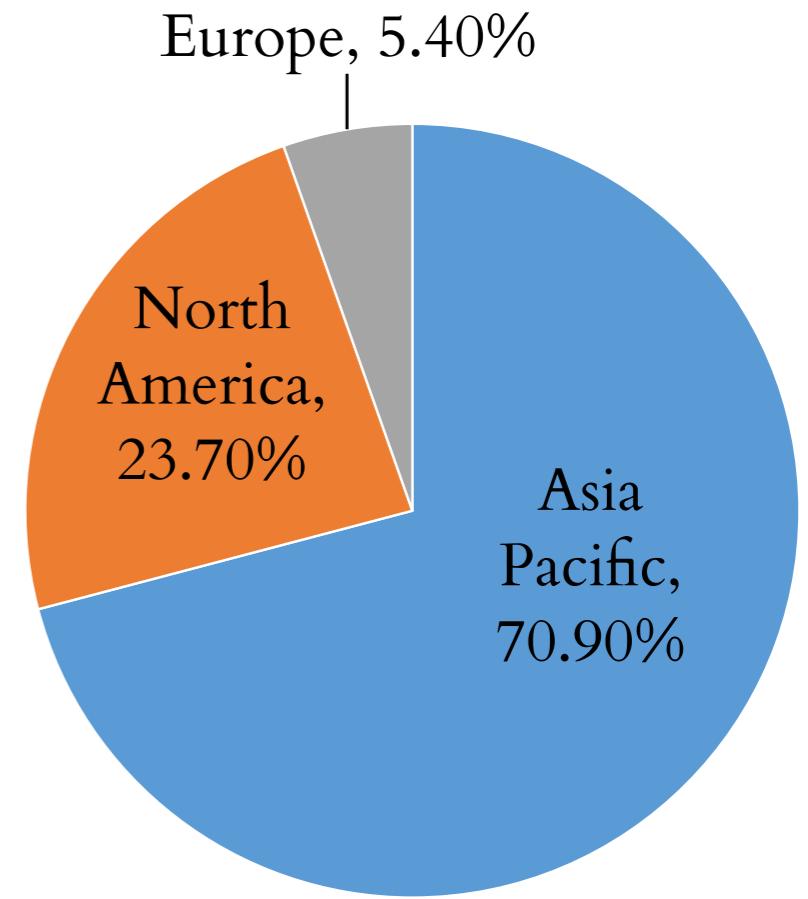
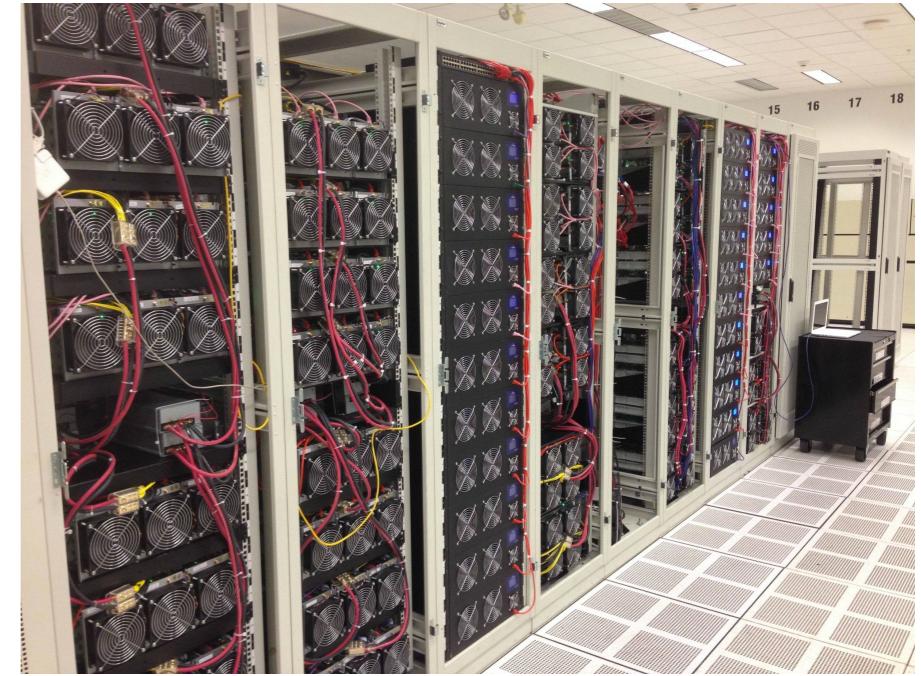
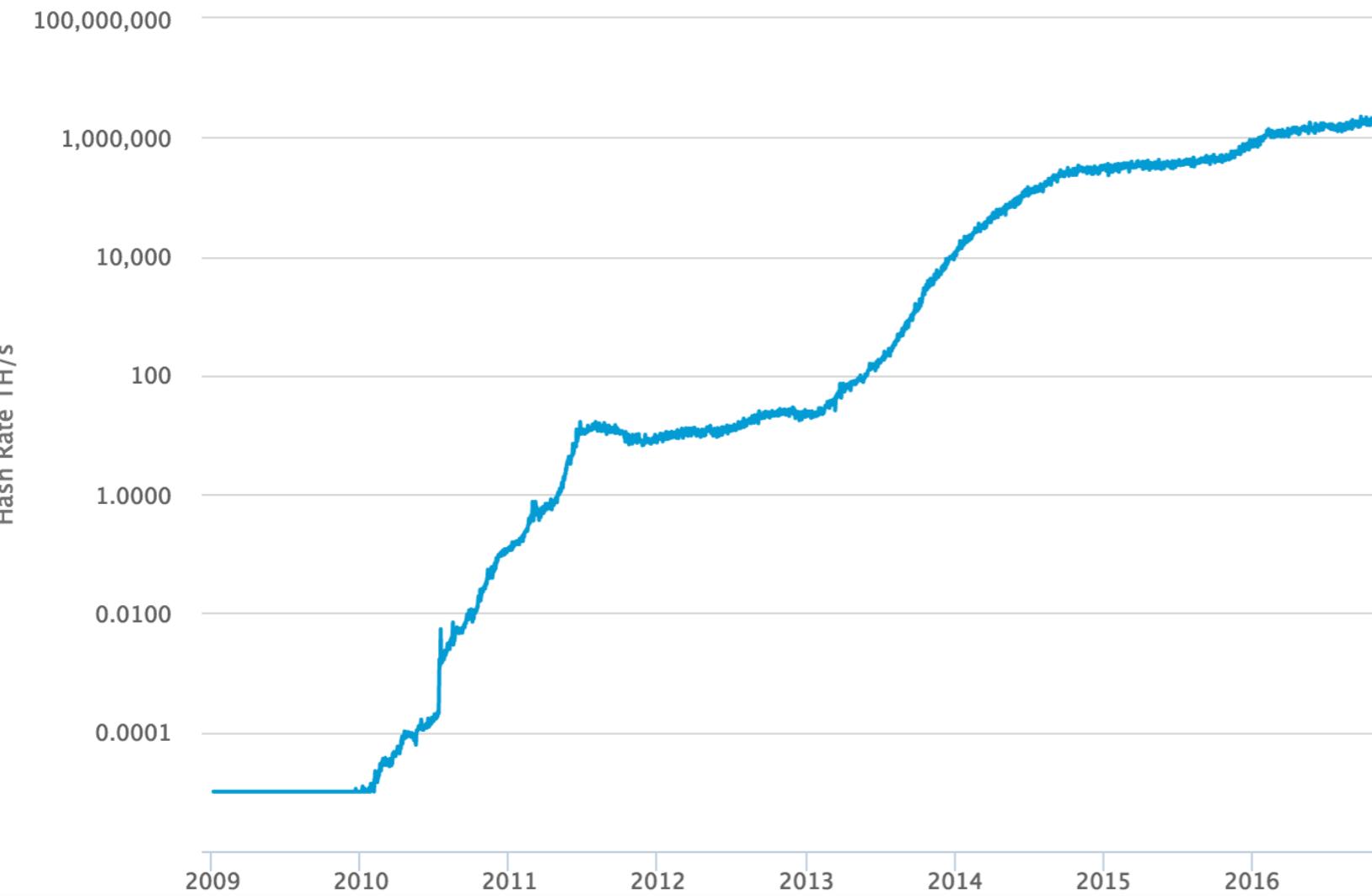
# Blockchain Forks



# Mining



- From CPU to GPU to ASICs



# Mining Pools



- Probability of finding a block alone is very small
- Unite in Mining pools
- Payout is done proportional to the work

