Principles of Distributed Ledgers

Arthur Gervais

Introduction

2040 BC

Balance sheet - Mesopotamia

Balance sheet

Mesopotamia

2040 BC

BNP PARIBAS

1848

HSBC

Bank of America

Turing

1936

VISA

1958

mastercard

ECash

Chaum

1983

Bitcoin

2008

Ethereum

2015

ZCASH

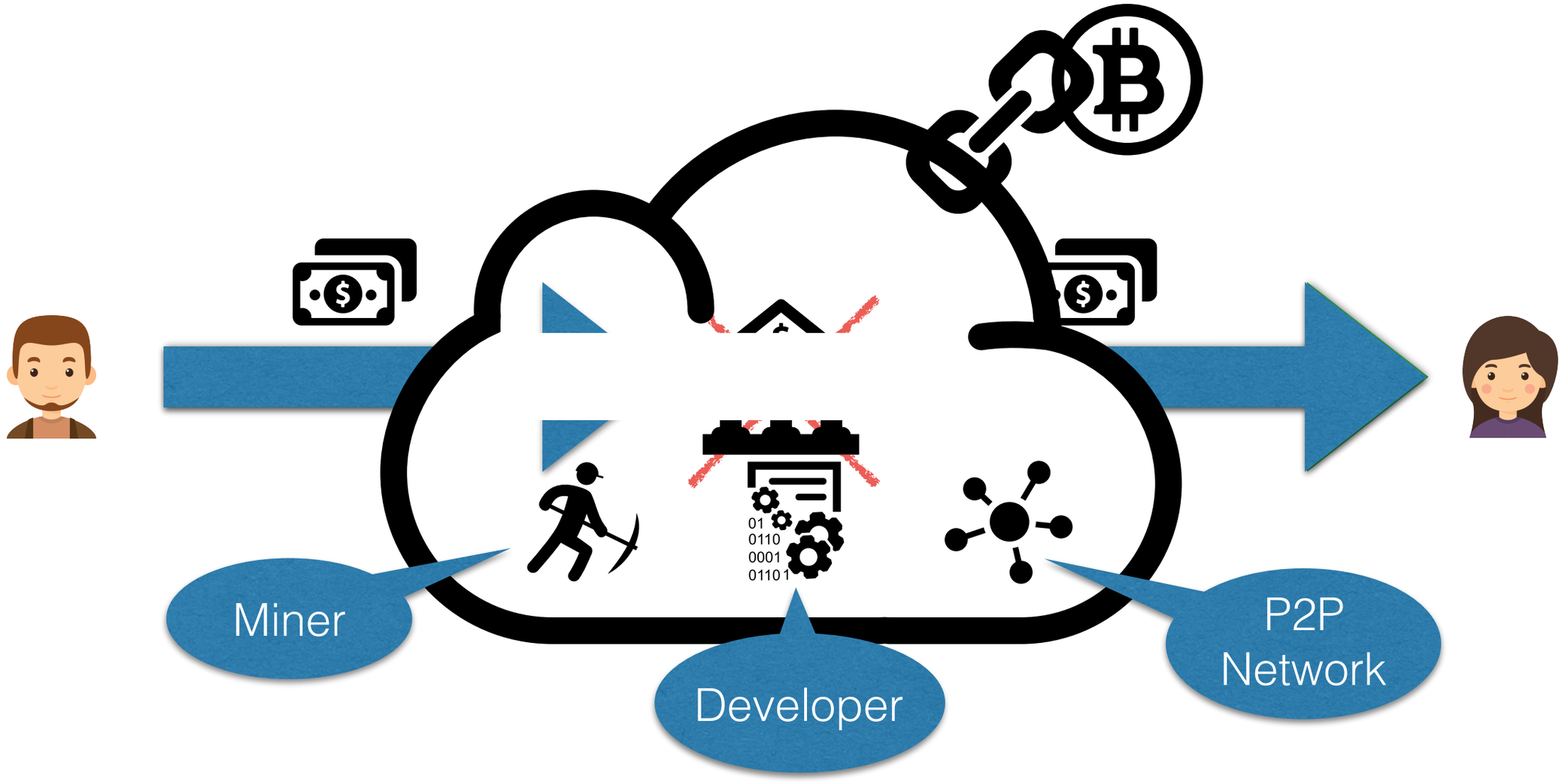
2016

Centralized Banking

Privacy-Preserving Banking

Decentralized Banking

From Centralized to Decentralized Payment Systems

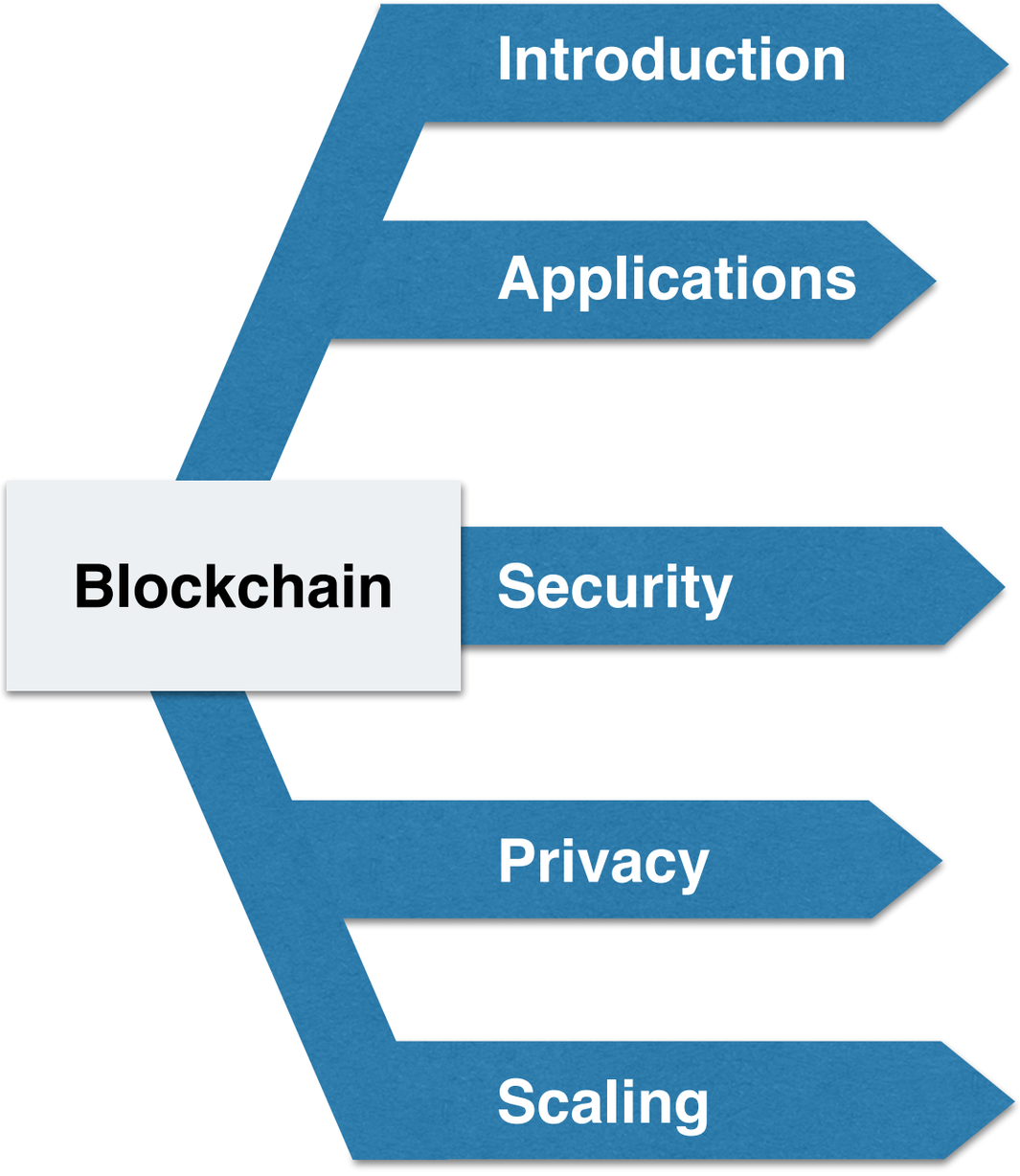


How to perform secure **decentralized** payments?

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

How to make decentralized systems **efficient**?

Course Outline



Introduction & Background

Mining & Transactions

Smart Contract Programming

Consensus Protocols & P2P

Blockchain Security

Blockchain Privacy

Scaling & Alternatives

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

- Bitcoin and Cryptocurrency Technologies
  https://lopp.net/pdf/princeton_bitcoin_book.pdf

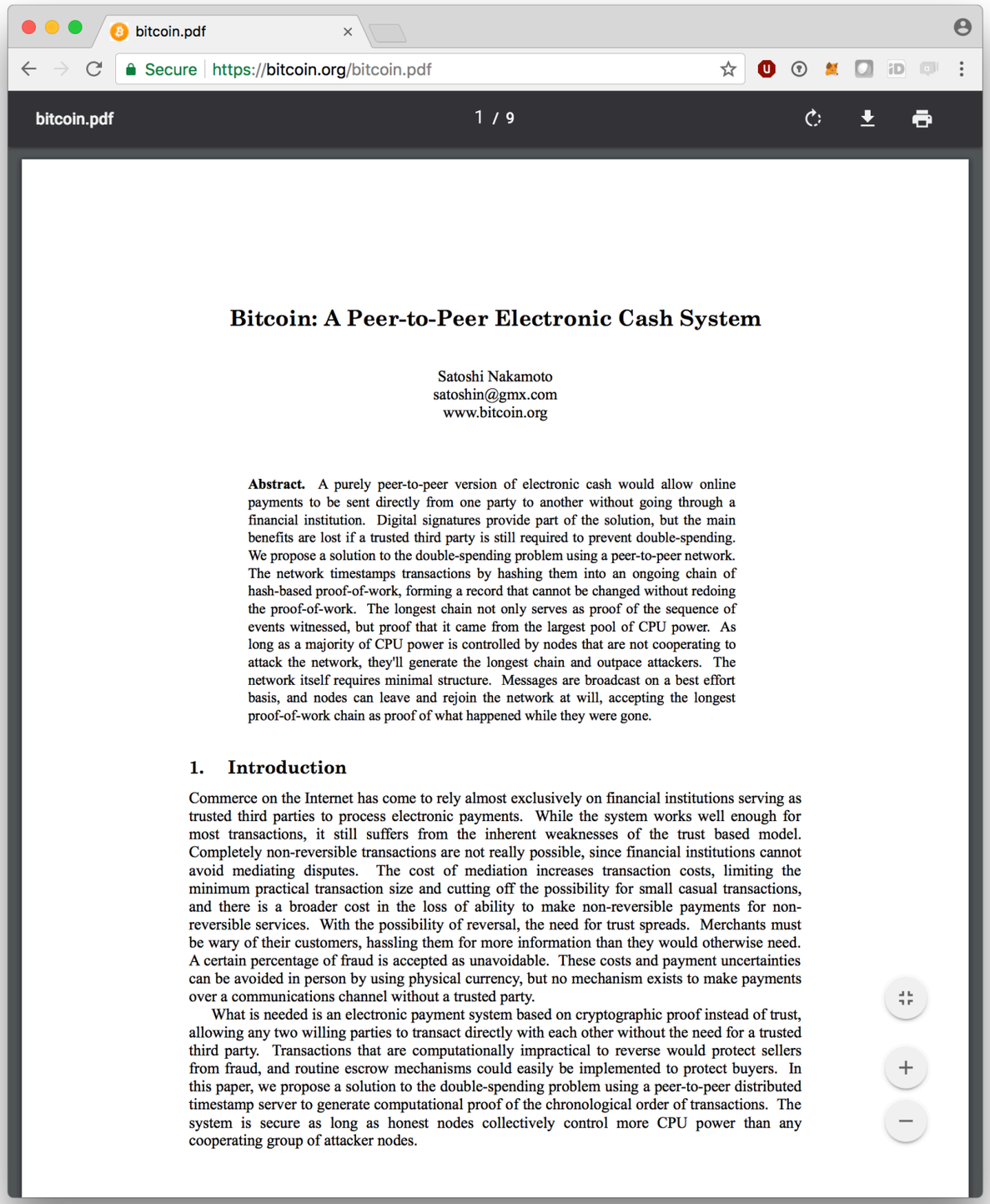
- Mentimeter
  For questions and live interactions

- GTA
  Sam Werner, Dominik Harz, Rami Khalil, Alexei Zamyatin, Lewis Gudgeon

Paper Reading

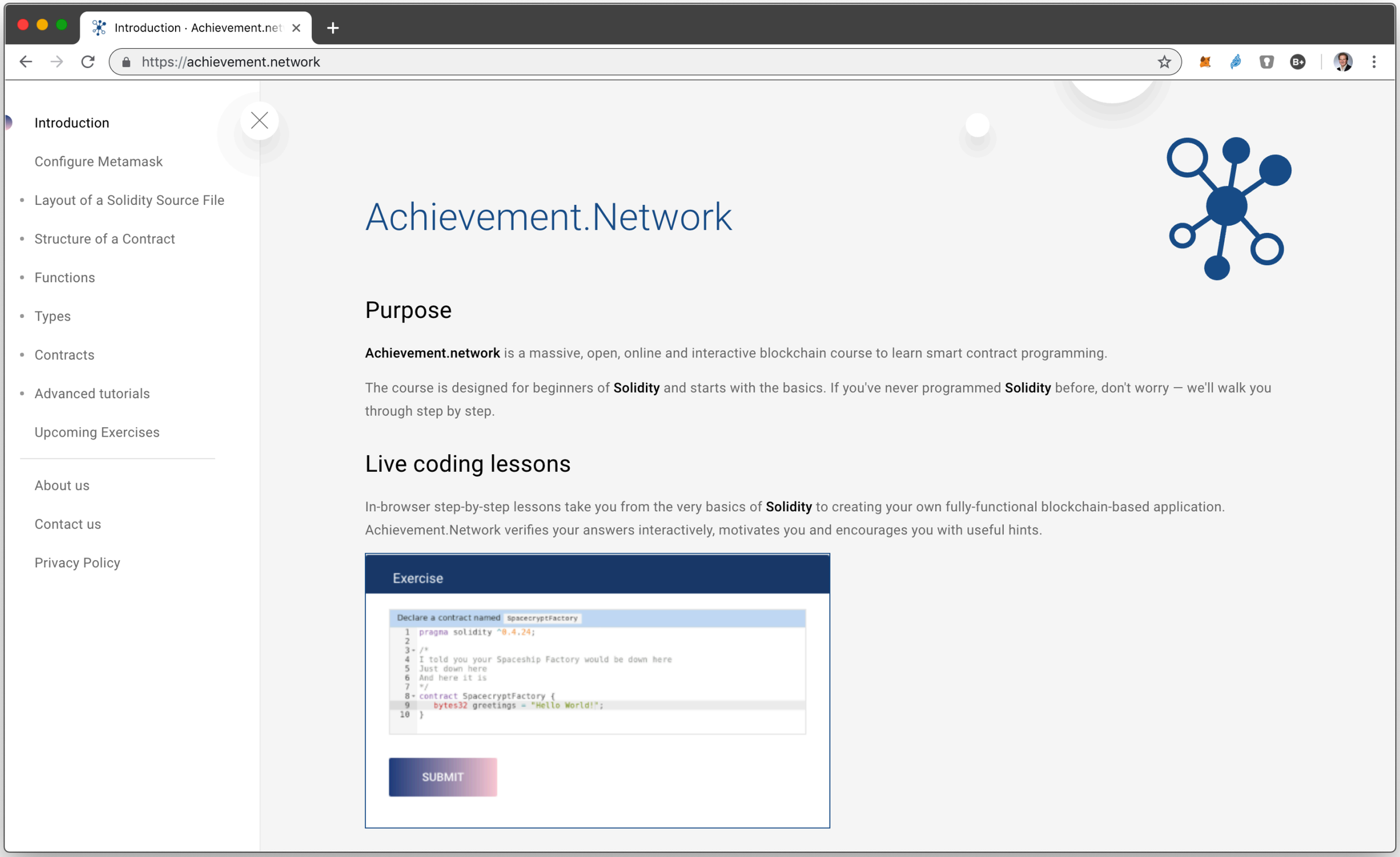
- We will average 1 paper per week
- Reading list is provided on Piazza
- Everybody must have read the paper. I'll choose 1 random student to present it during 5 minutes (without slides), graded
- Submit 1 unique question to Piazza about the paper, graded

Paper Reading

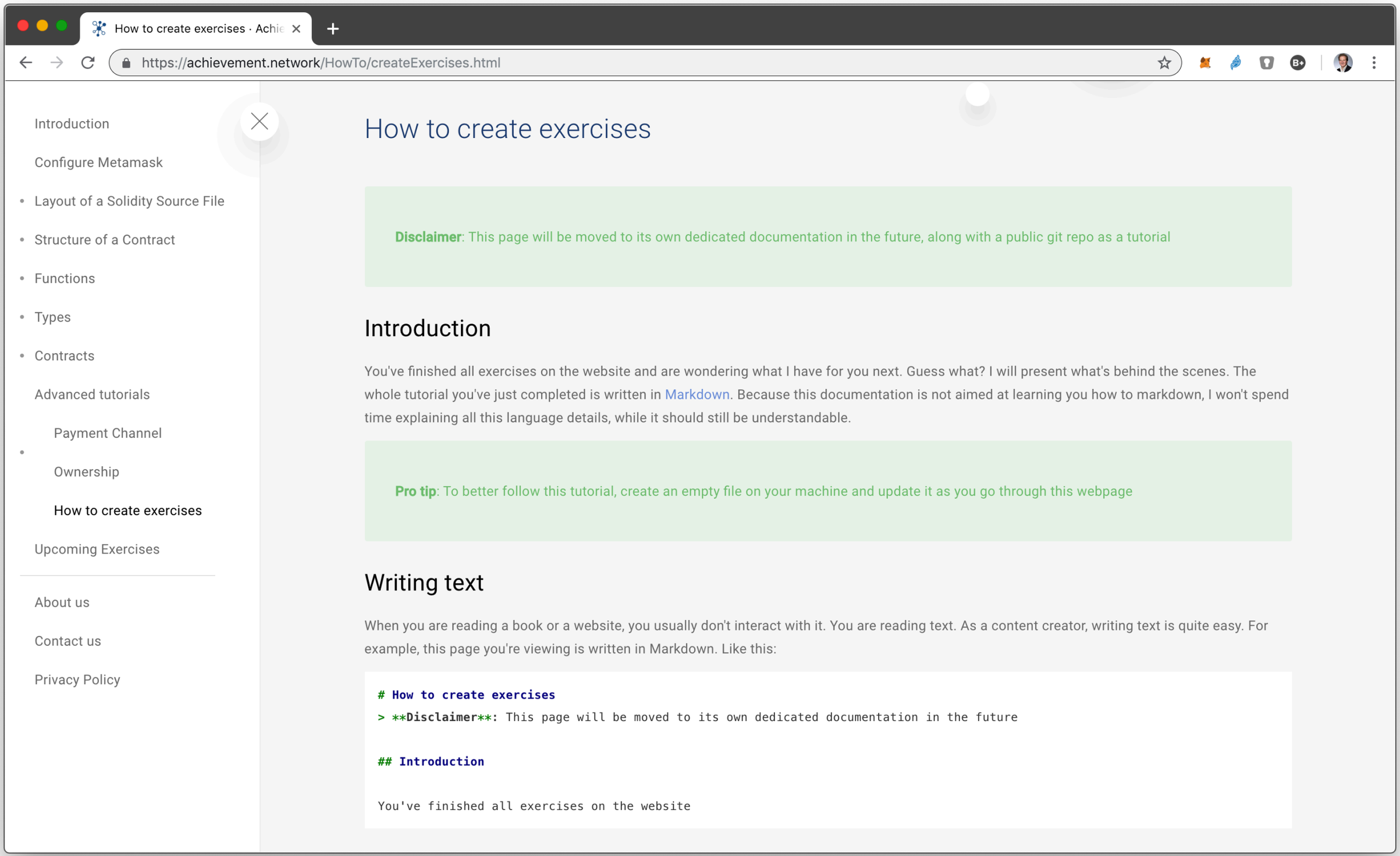


- https://bitcoin.org/bitcoin.pdf

CW1 and 3: Create an exercise on Achievement.Network



CW1 and 3: Create an exercise on Achievement.Network



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 or/and the pyethereum simulator
- Write a report about the project

CW4: Write a scalable Blockchain Application

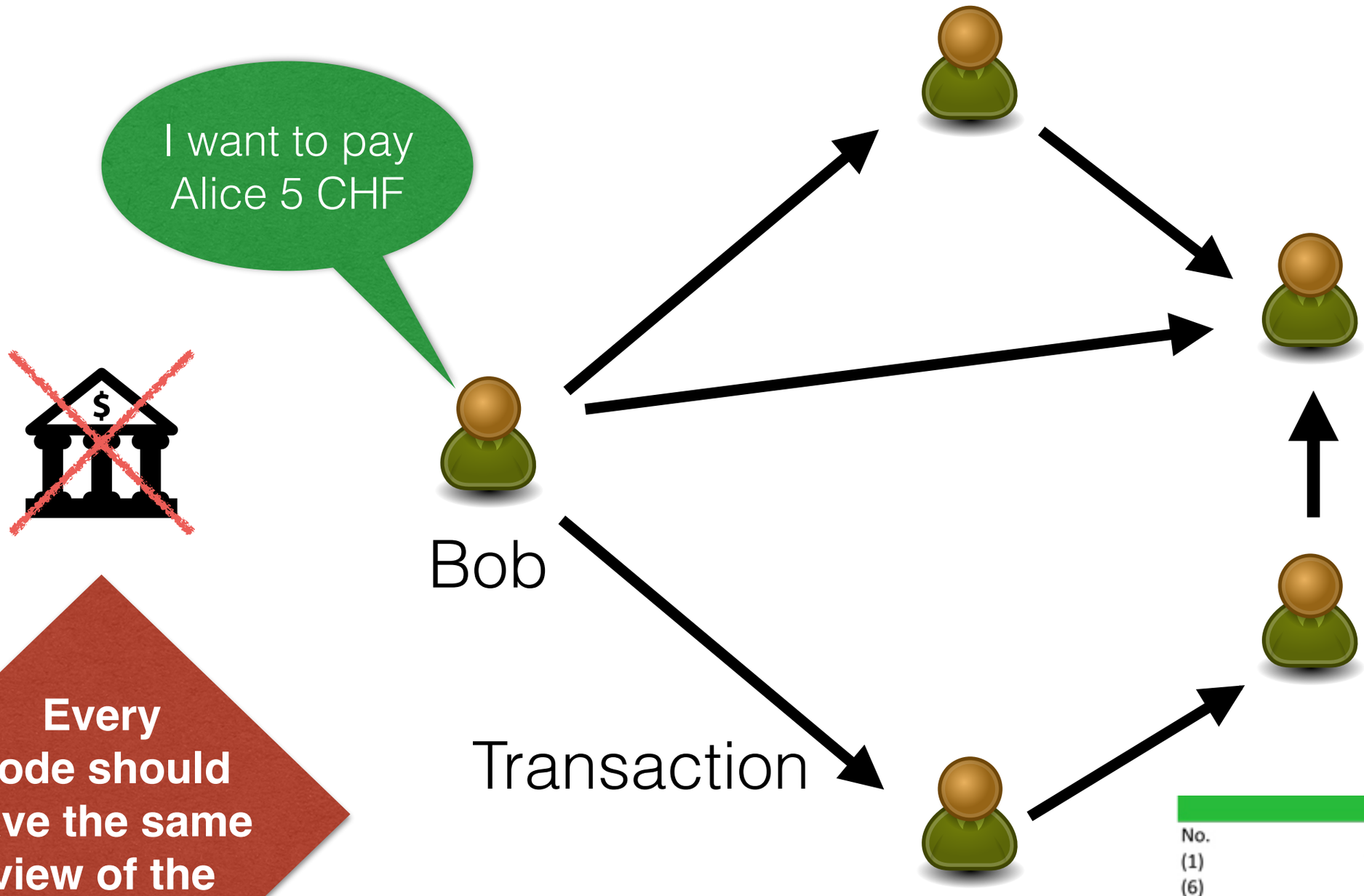
- Choose a use-case
- Choose a technology to scale
  - 2-party channels
  - Commit chains
- Write a report

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



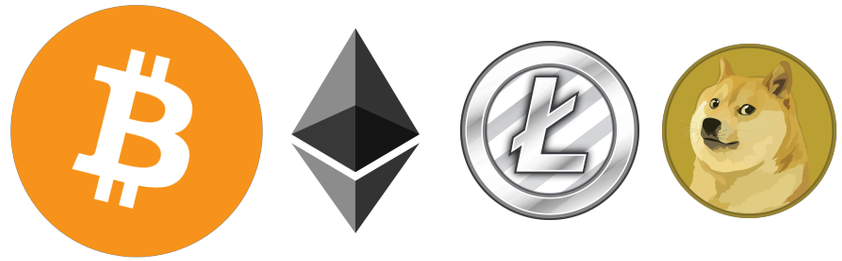
Cash			
No.		No.	
(1)	10,000	(5)	500
(6)	500	(8)	200
(12)	2,000	(9)	500
(14)	25,000	(10)	500
		(11)	1,000
		(13)	1,000
		(15)	1,000
Totals	37,500	Totals	(4,700)

Every node should have the same view of the ledger

Blockchain != Blockchain

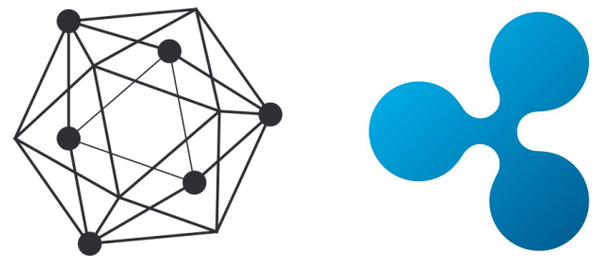
Open and Decentralised Blockchains

- Ethereum
- Bitcoin



Permission-based Blockchains

- Hyperledger
- Ripple
- Stellar



Open and Decentralised Blockchains

Bitcoin

- First introc
  - Pseudor
- Peer-to-pe
- No trusted
- Blockchair
  - Transac
  - Blocks

Is Bitcoin a Decentralized Currency?

Arthur Gervais* Ghassan O. Karame** 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

- E

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 || 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 || x) = h in time significantly less than 2^n.

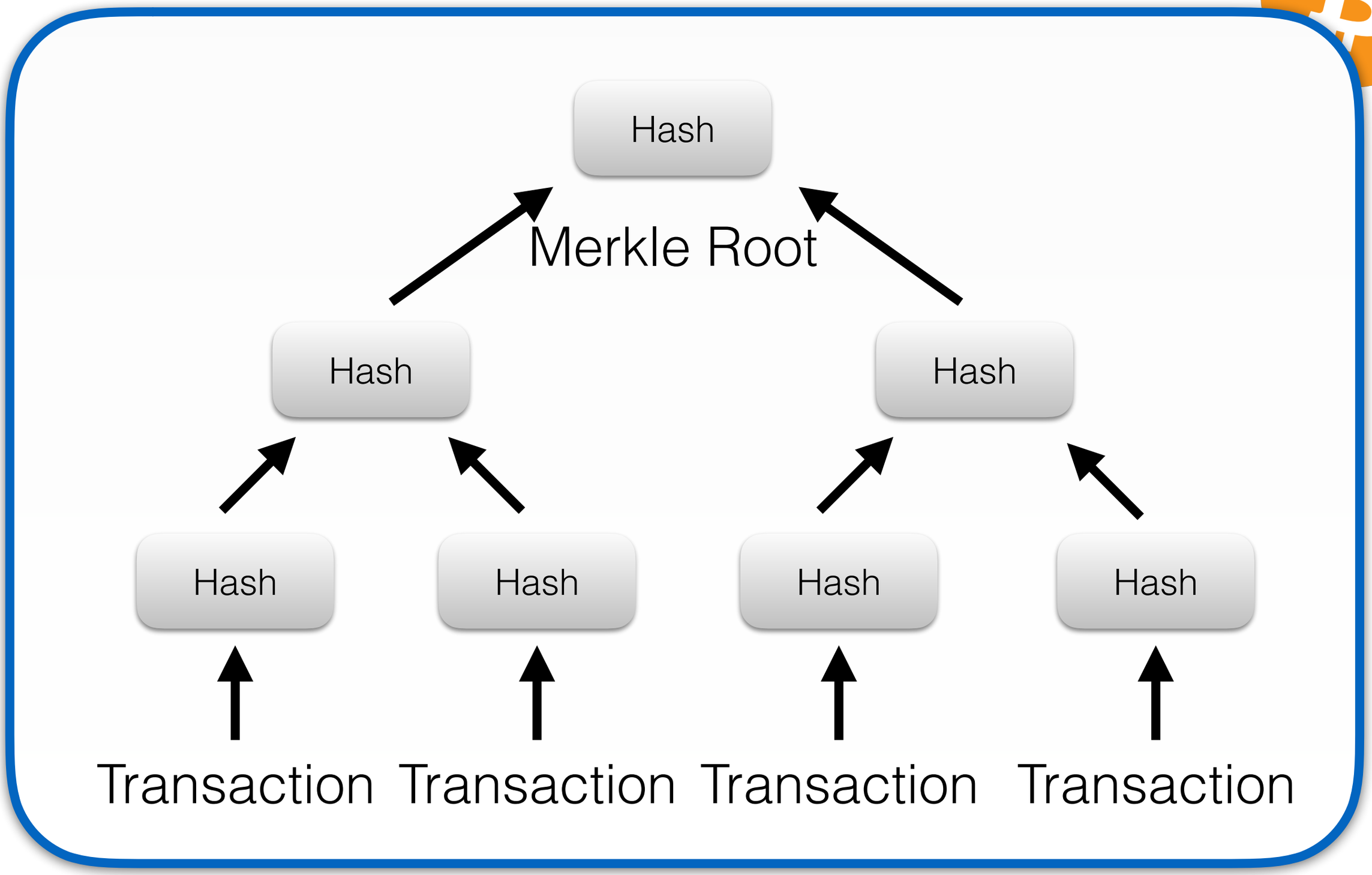
Search puzzle

- 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


Hash
Merkle Root
Hash
Hash
Hash
Hash
Hash
Hash
Hash
Transaction
Transaction
Transaction
Transaction

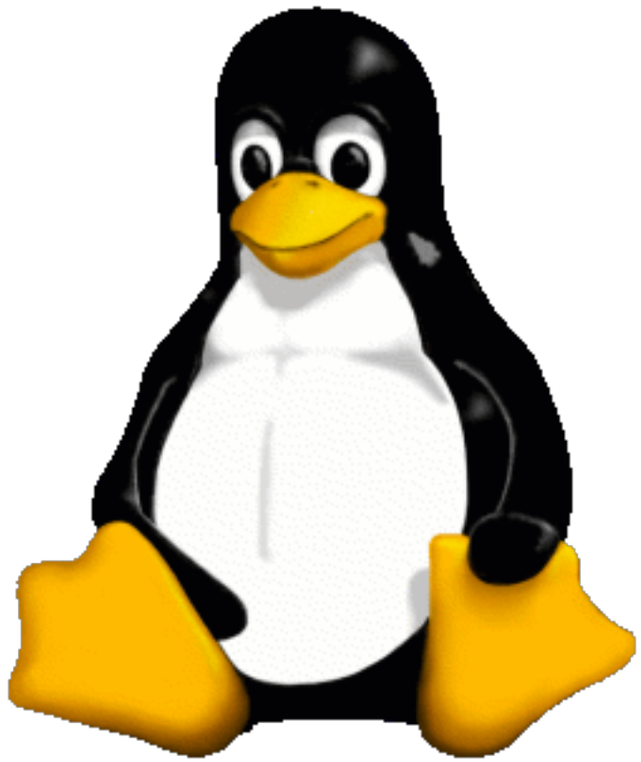
Data Types

- 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 ic
- Hash of a
- 0 < Balar
- 1 Satoshi

1EGa

3J98t

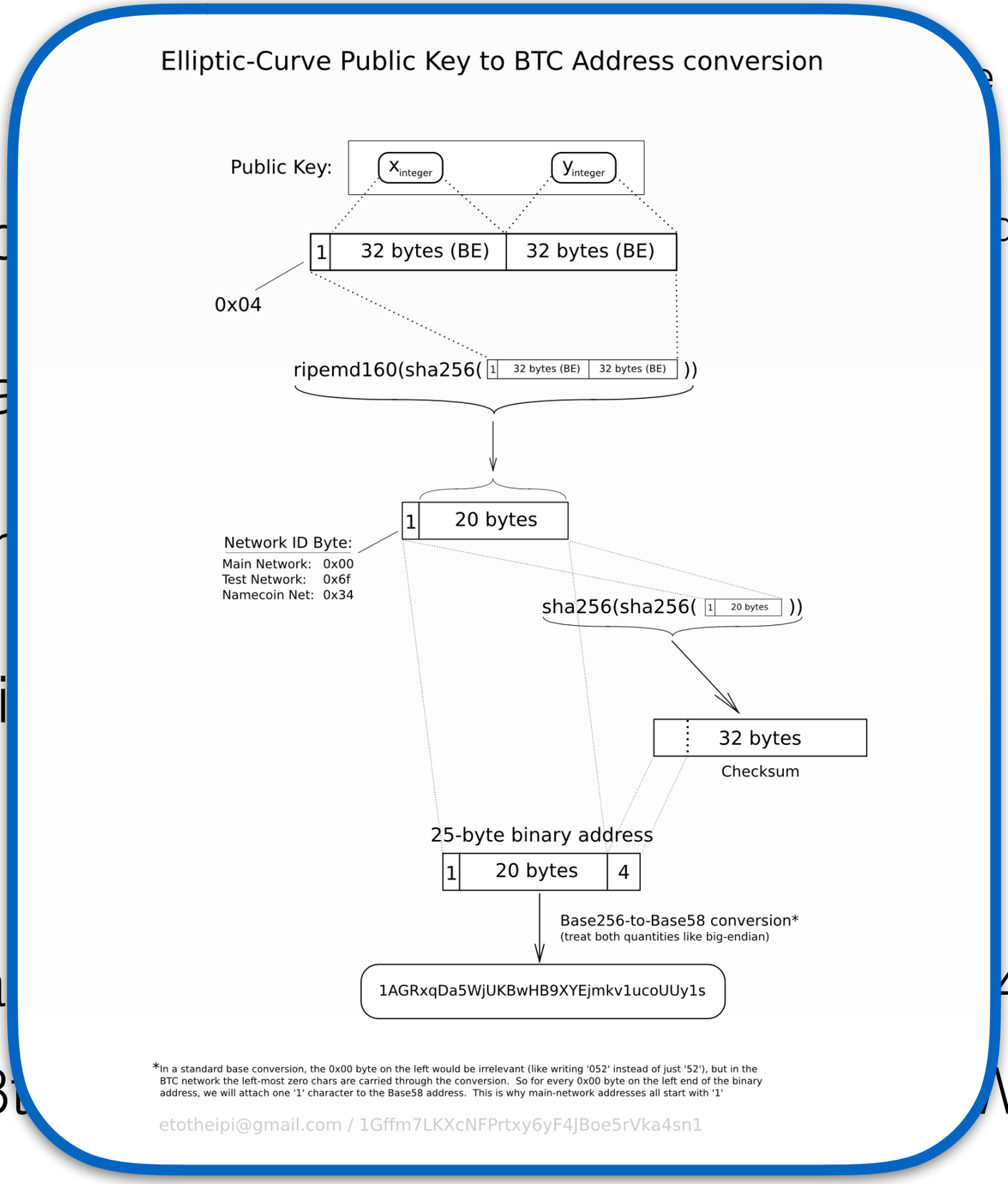
ob

@Eve

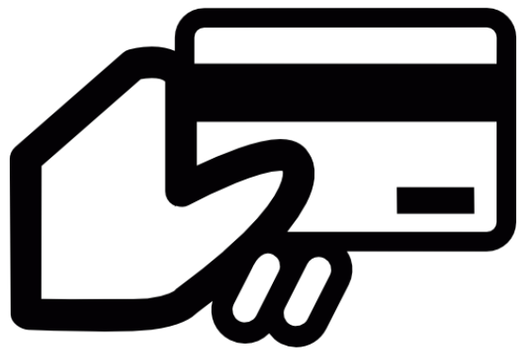
@ETH

4YTEr

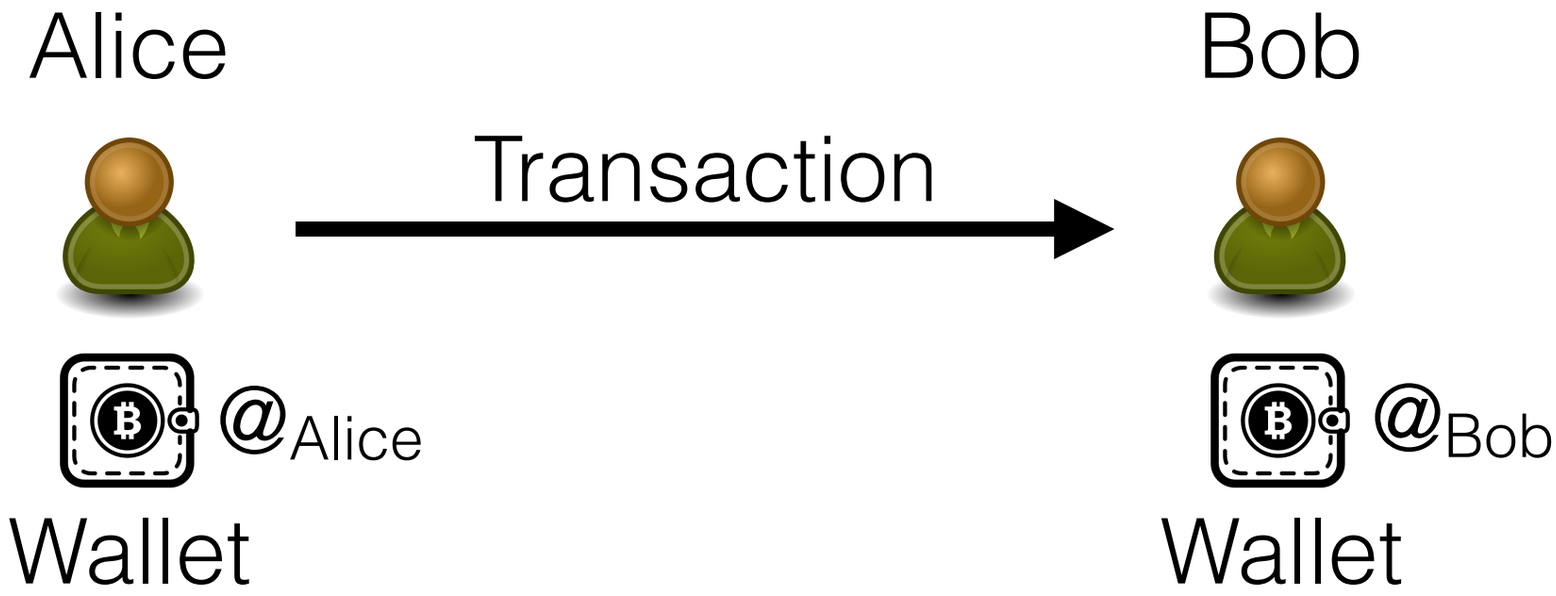
WNLy

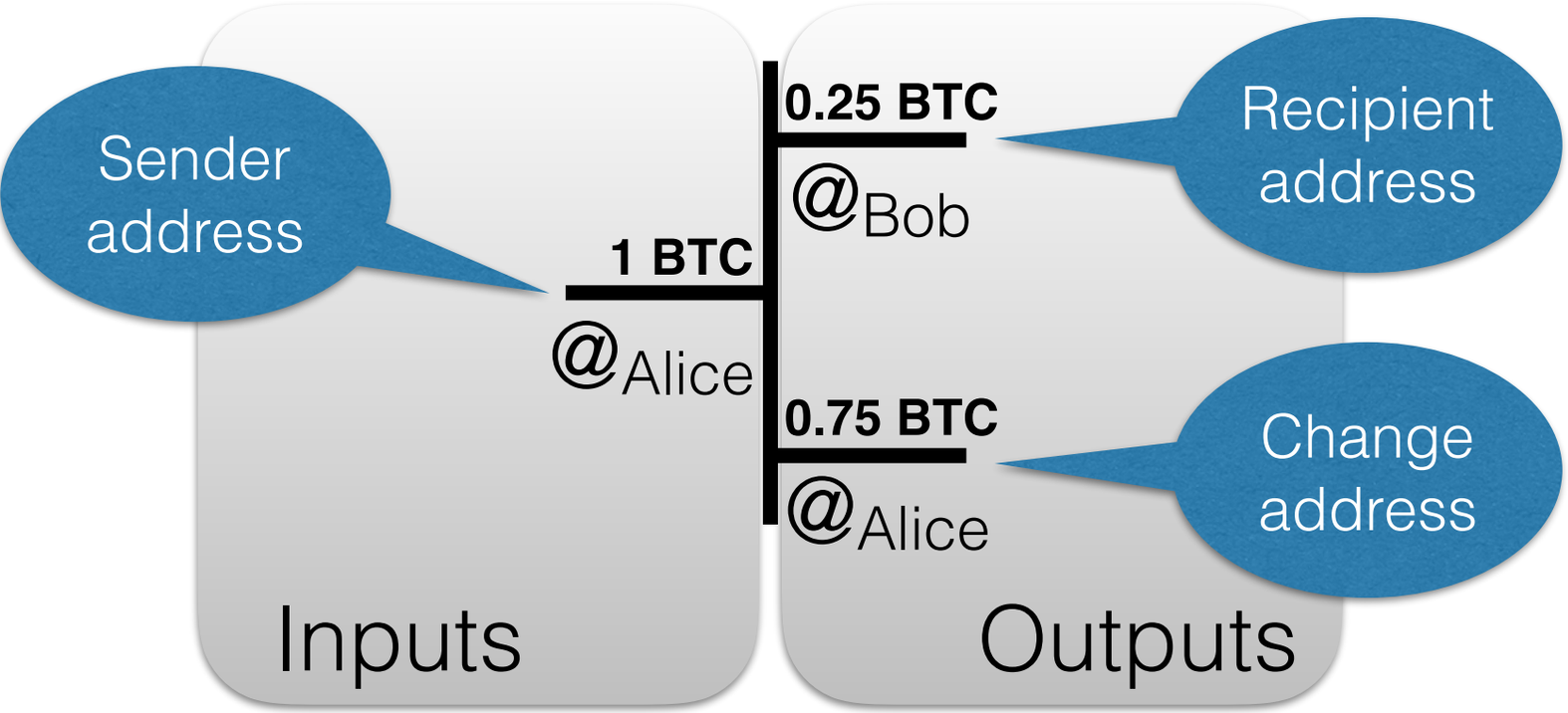


Transactions



Transactions in Bitcoin



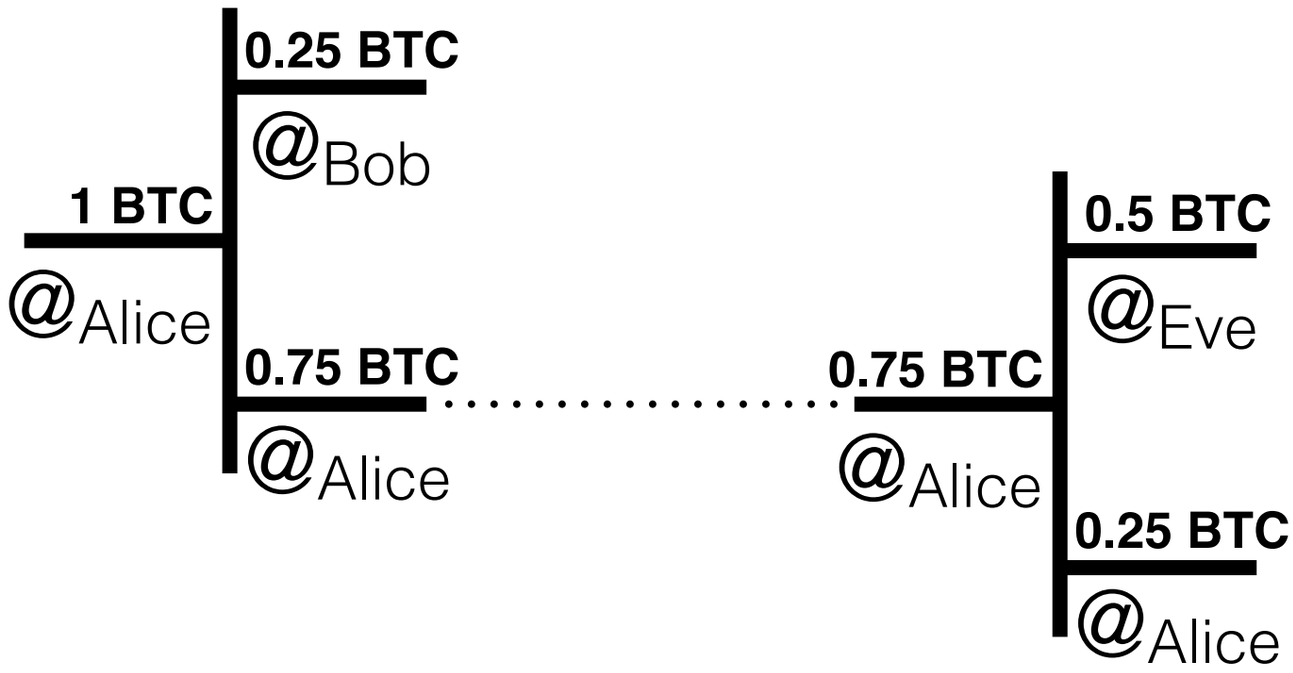


Transaction Fees

$$\sum inputs \geq \sum outputs$$

Difference are fees

Transactions in Bitcoin

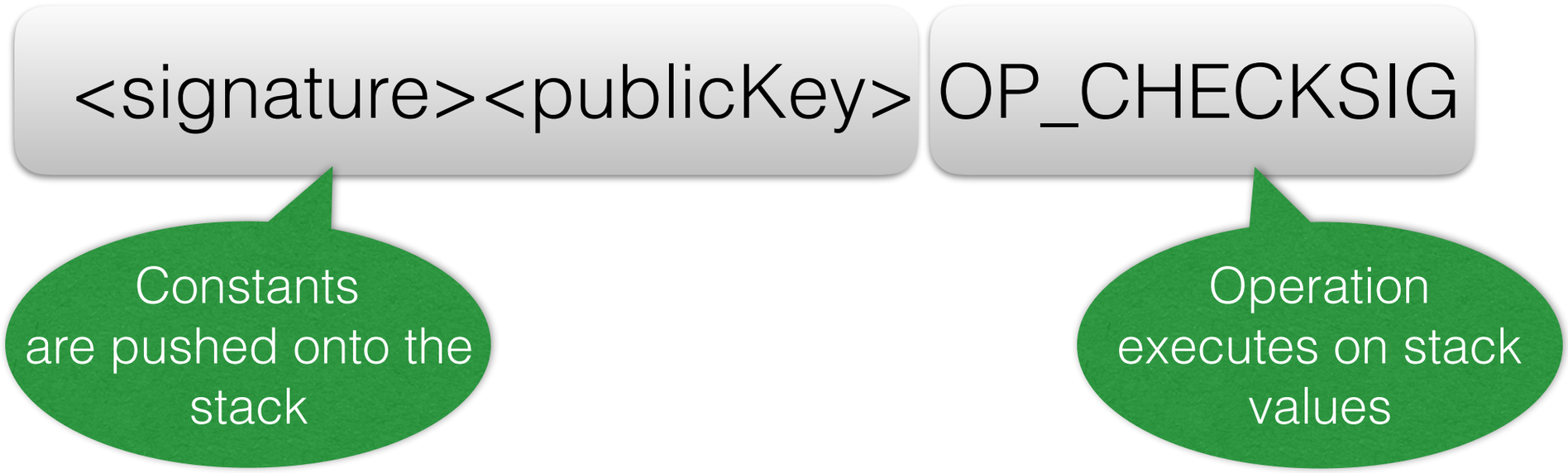


Transaction 1 Transaction 2

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



<PubKey> <Sig>	OP_DUP
Execution Stack	Execution Code

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

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

<PubKey> <Sig>	OP_CHECKSIG
Execution Stack	Execution Code

True

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

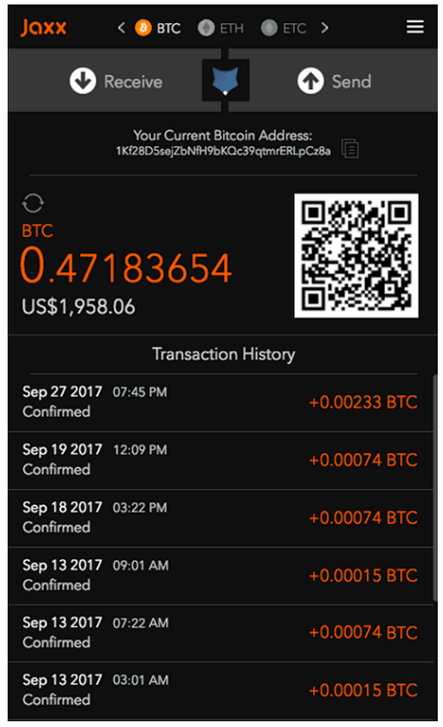
Blockchain Wallets

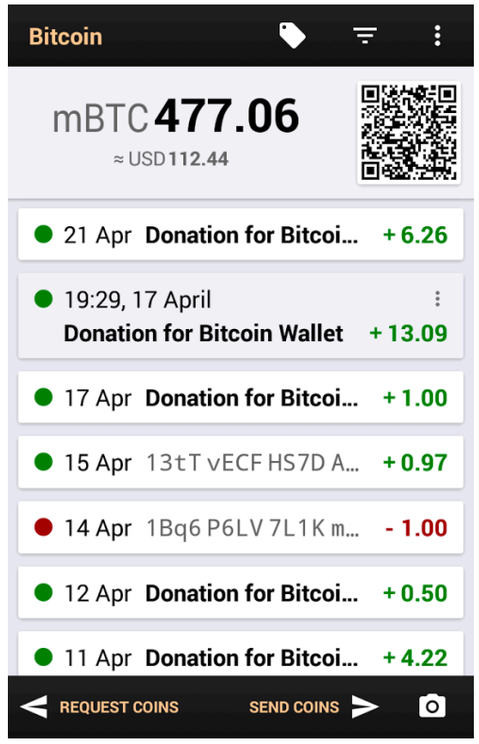
- Bitcoin Core
  Full node, downloads all transactions

- Bitcoin Wallet
  Android

- Jaxx
  Multi-chain Wallet

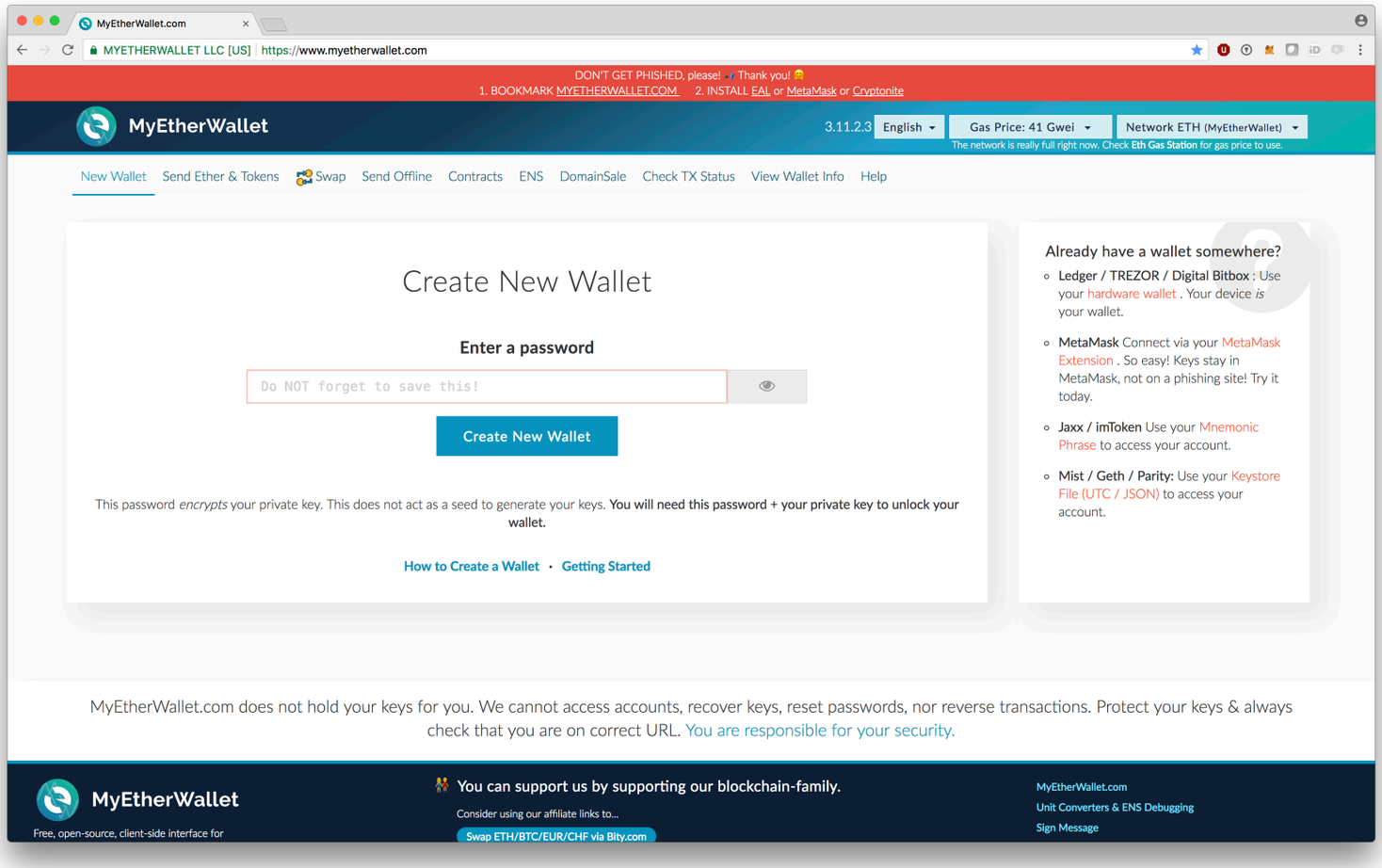
- Ledger Nano S

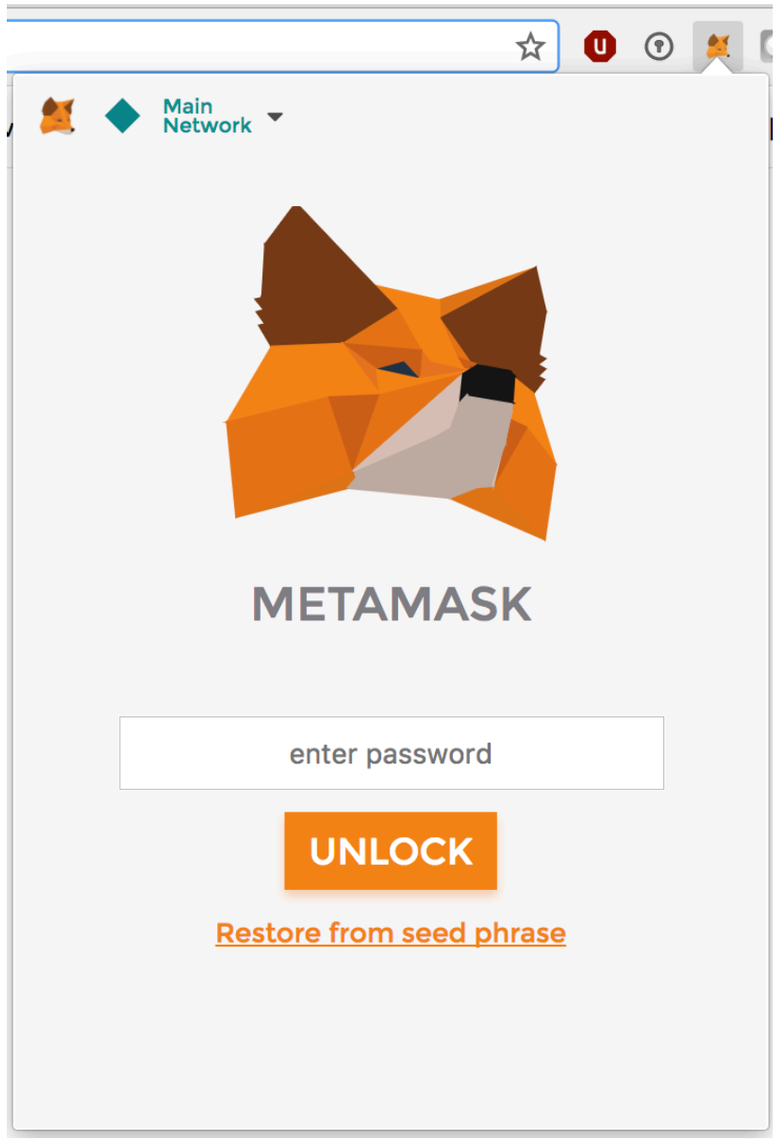




Blockchain Wallets

- MetaMask
- MyEtherWallet

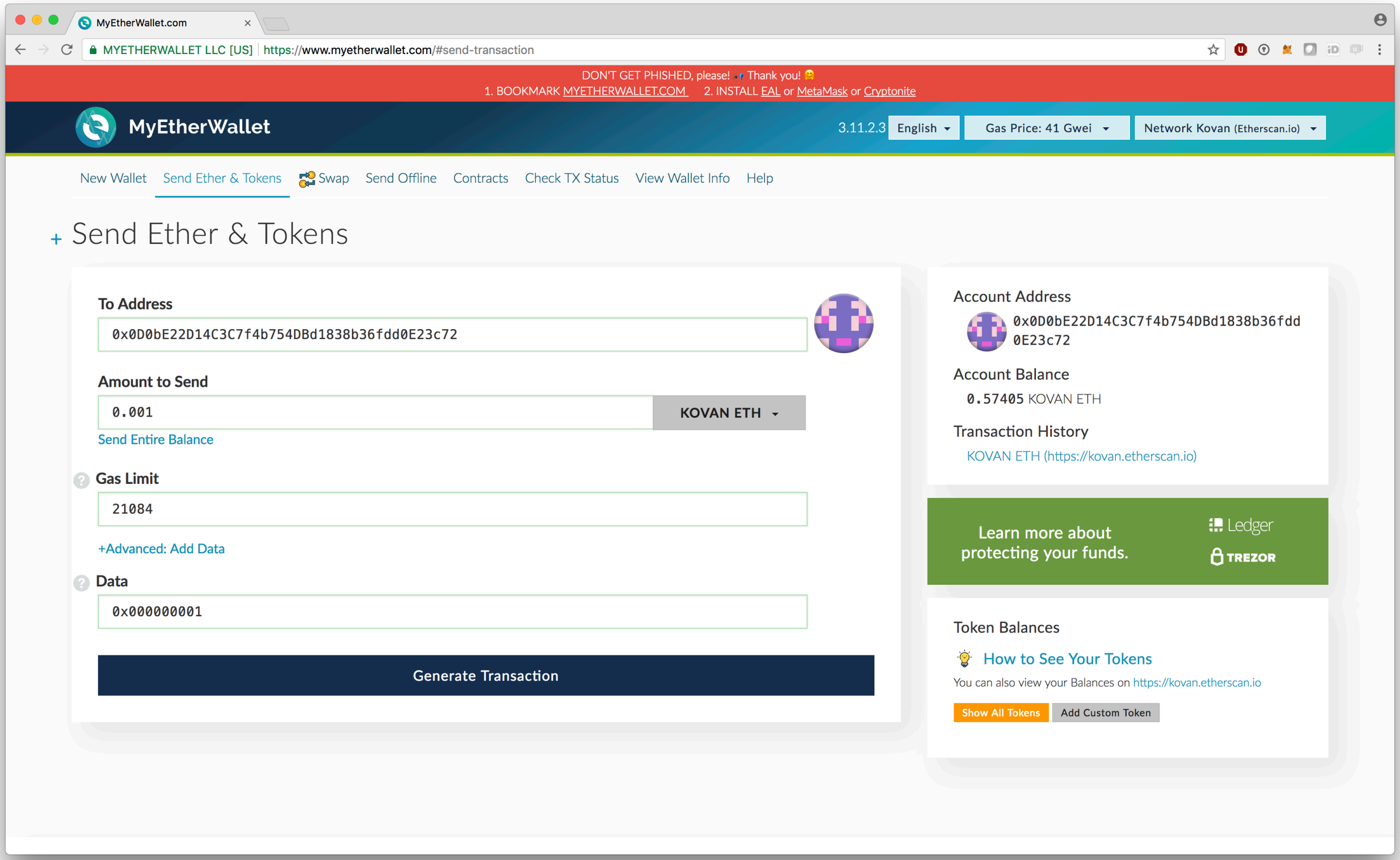




For next class..

- Create a wallet on MyEtherWallet
- Get test Ether for the Rinkeby network
- Send to address “0x0D0bE22D14C3C7f4b754DBd1838b36fdd0E23c72”, 1 Rinkeby Test Ether and your student ID as data attached.

Ethereum Testnet Transaction



Checklist

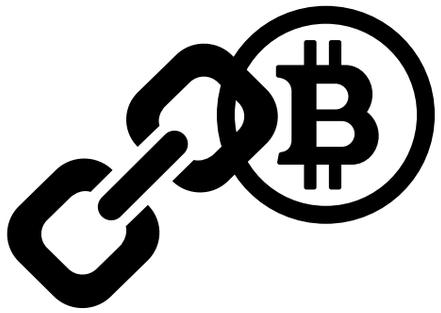
- Generate a keypair
- Give your “address” to someone
- Get someone else’s “address”
- Lookup the history of an “address”
- Use the transaction id/hash to look it up

What can you do with Crypto?

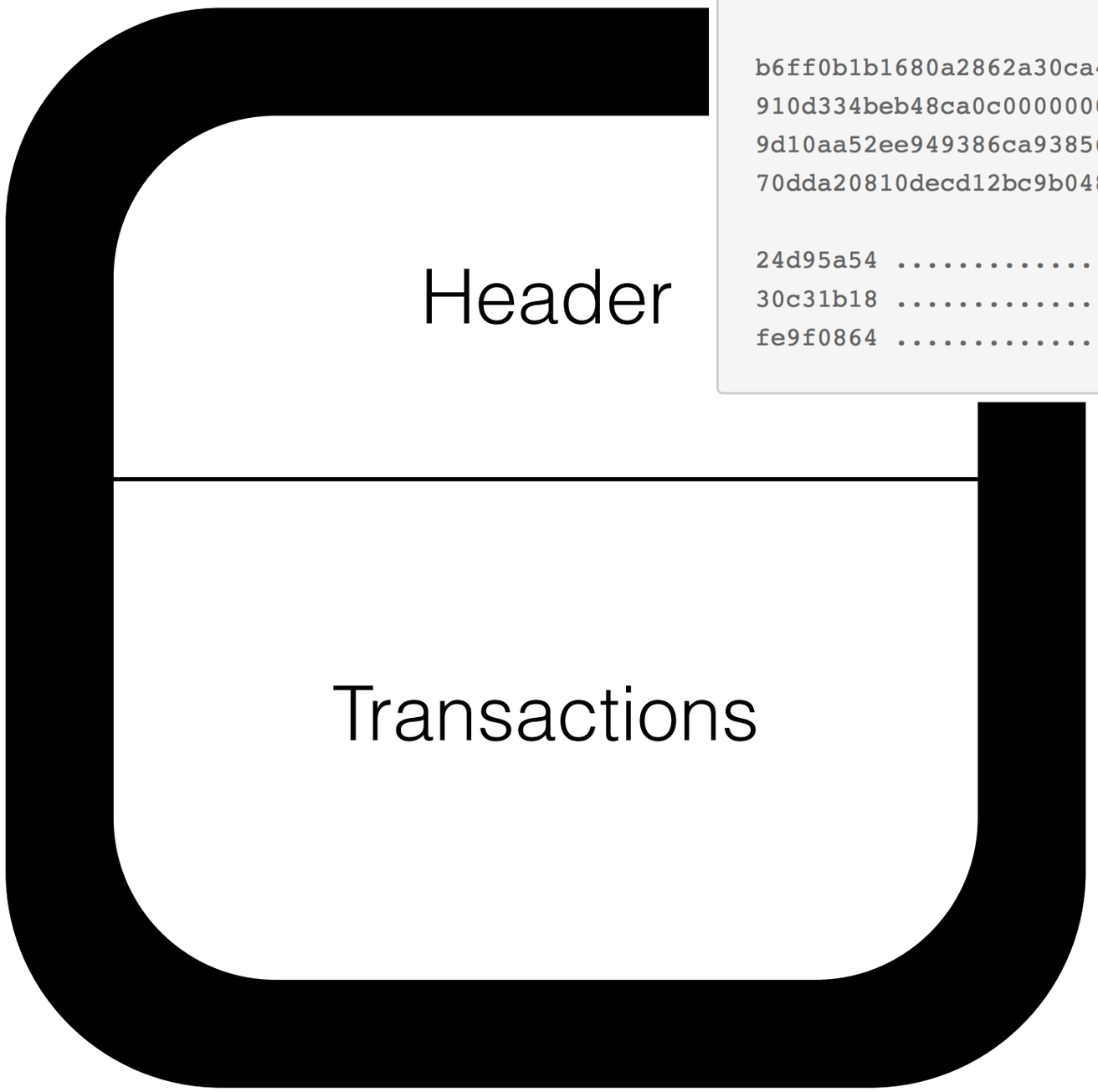
- Trade with other people
- Exchange to other crypto/FIAT
- Buy something online
- Donate
- Build applications/games

… and many more things!

Blockchain



Block



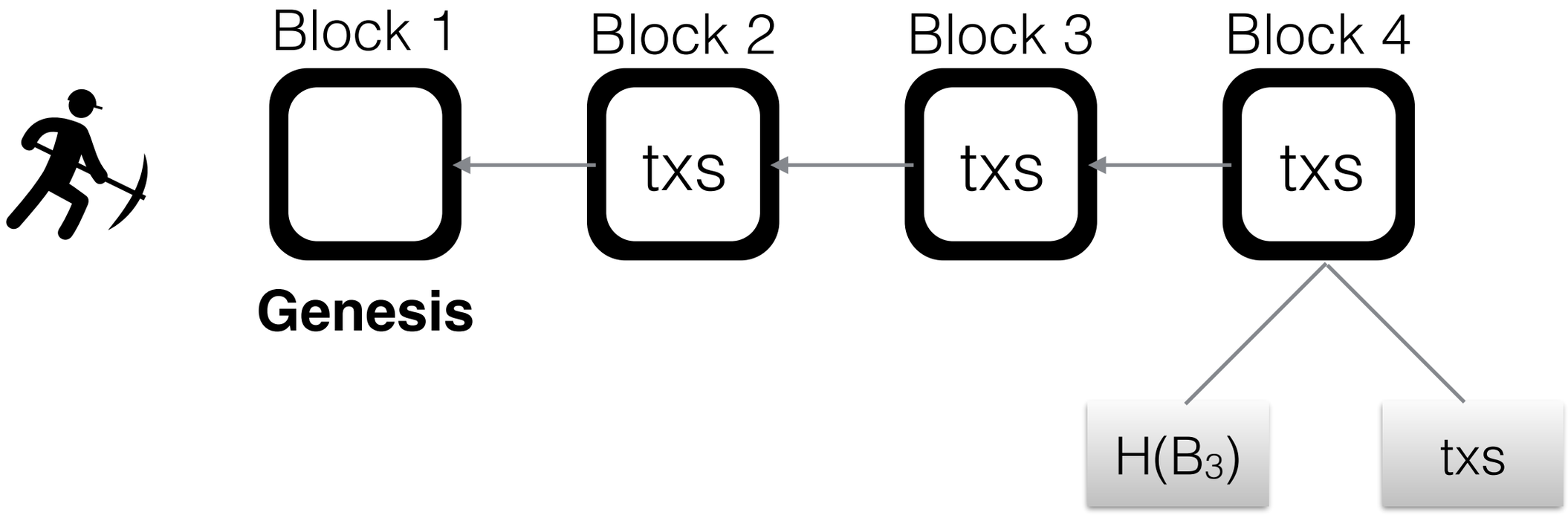

```
02000000 ........................... Block version: 2

b6ff0b1b1680a2862a30ca44d346d9e8
910d334beb48ca0c0000000000000000 ... Hash of previous block's header
9d10aa52ee949386ca9385695f04ede2
70dda20810decd12bc9b048aaab31471 ... Merkle root

24d95a54 ........................... Unix time: 1415239972
30c31b18 ........................... Target: 0x1bc330 * 256**(0x18-3)
fe9f0864 ........................... Nonce
```

https://bitcoin.org/en/developer-reference#block-headers

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

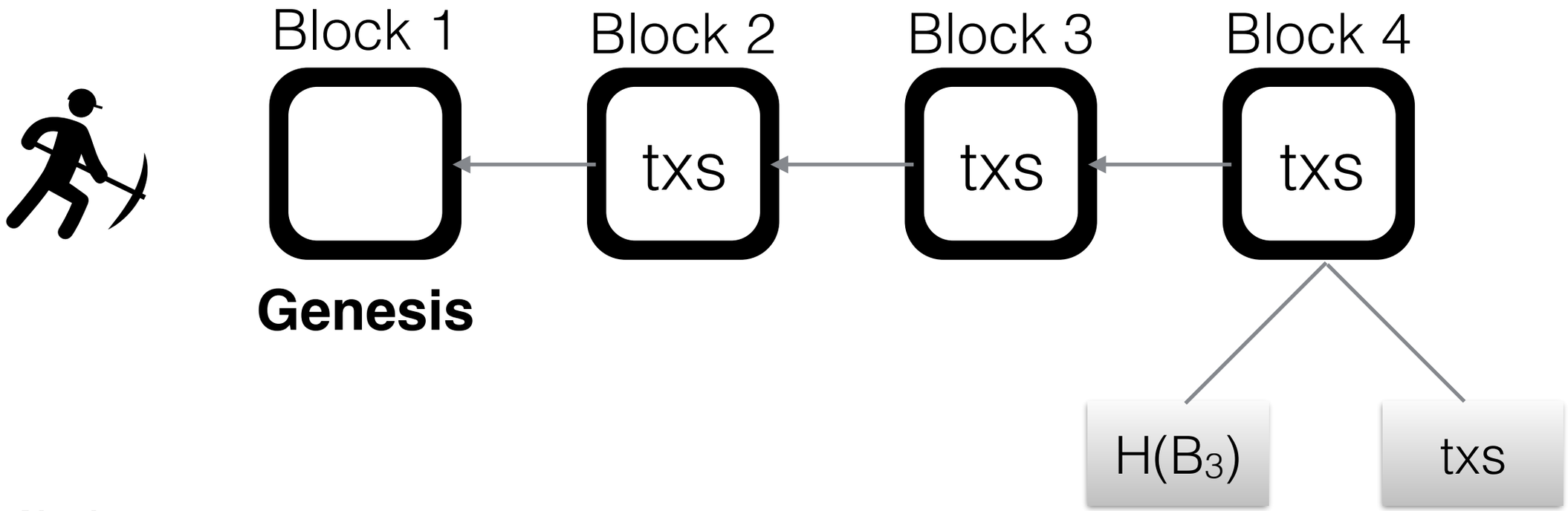
3. Construct a block template

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.

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

Best known approach: **Brute Force**

Controls the **difficult**

Mining Difficulty

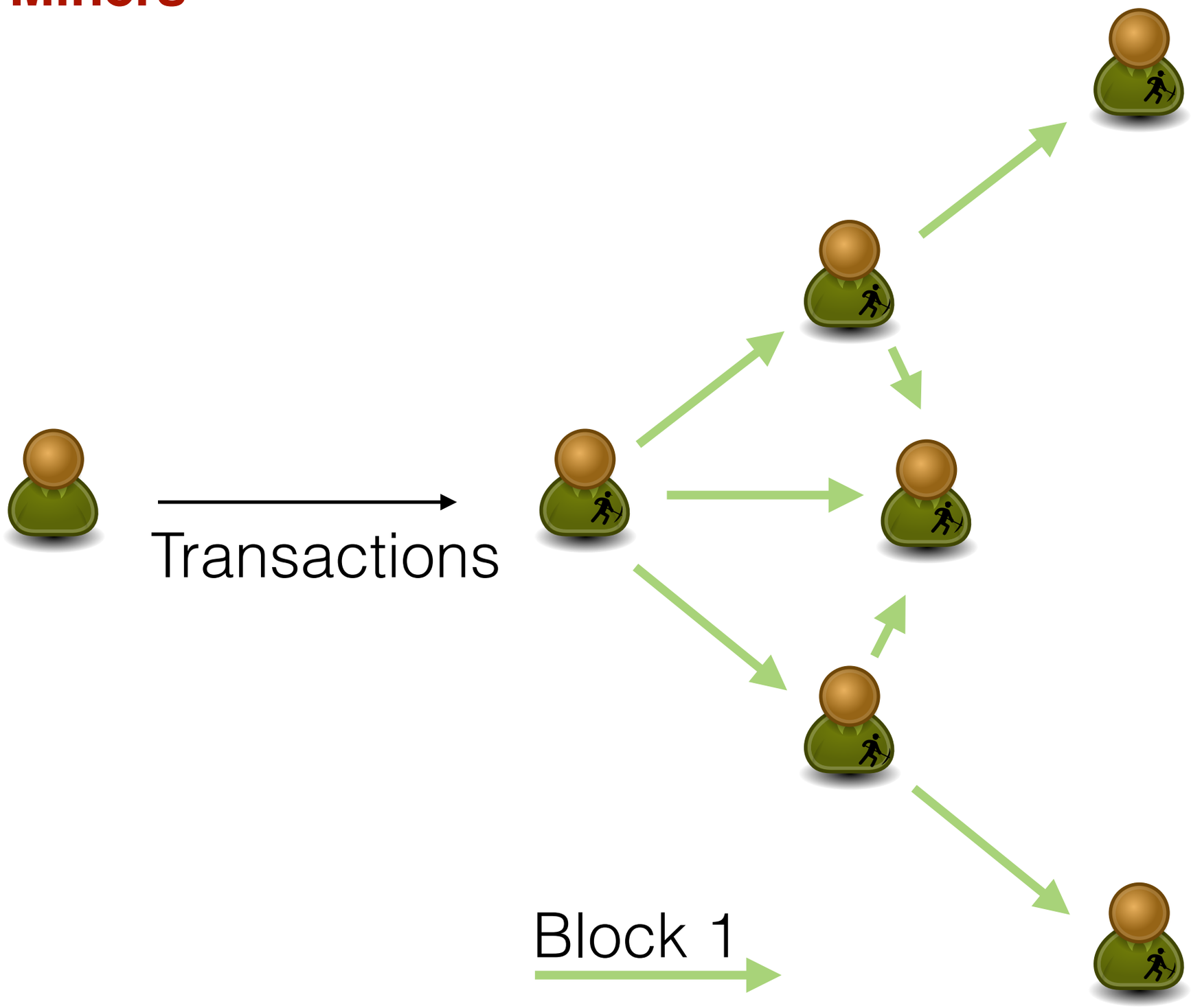
$$\text{Hash}(\text{Hash}(B_3)|\text{txs}|\text{N}) < \text{target} = 0\text{x}000^{**}$$

~~Hash(Block_3 | merkle_root | 0xbeed) = 0x03ef..~~

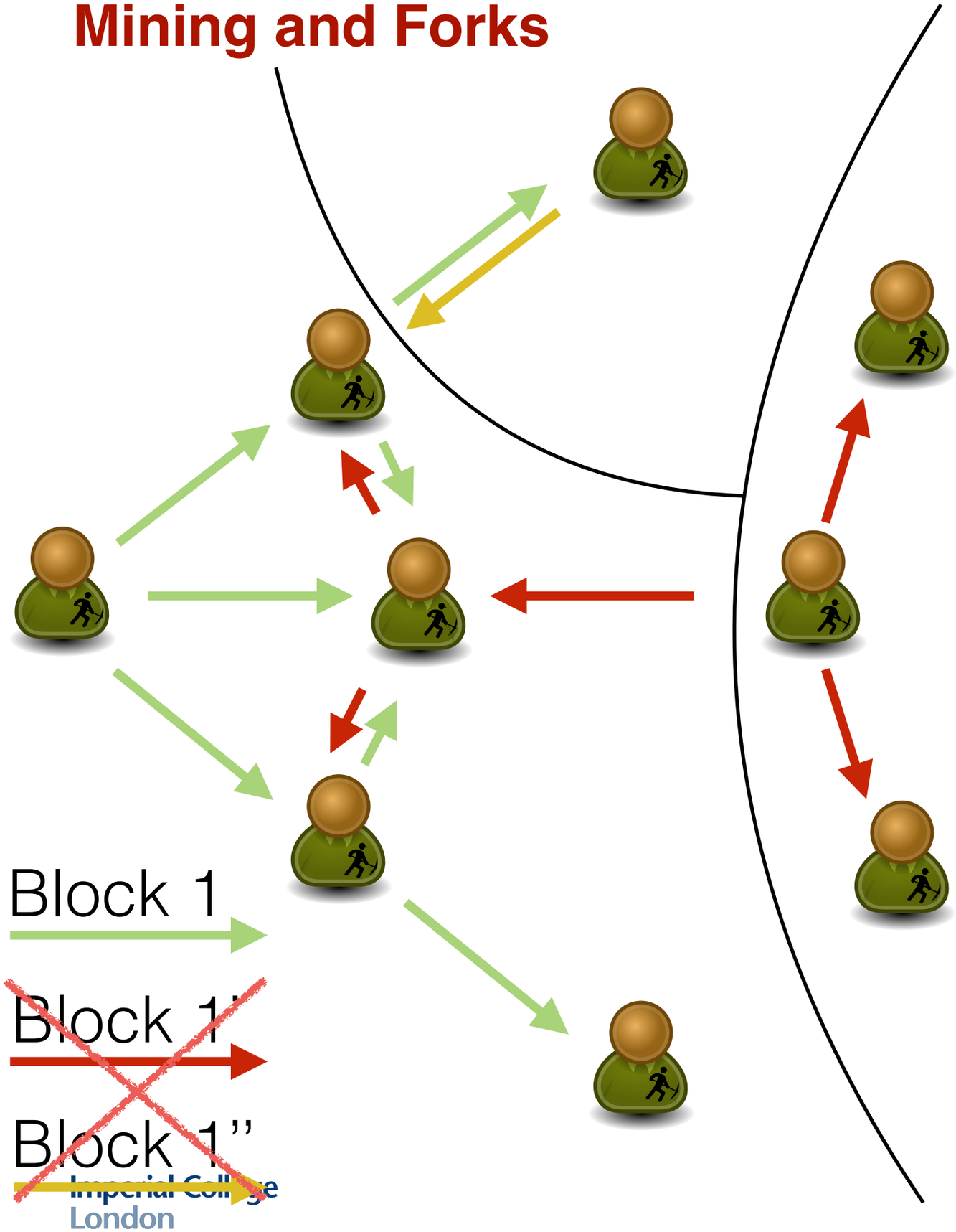
~~Hash(Block_3 | merkle_root | 0xbeee) = 0x12ef..~~

Hash(Block_3 | merkle_root | 0xbeef) = 0x000f..

Miners



Mining and Forks



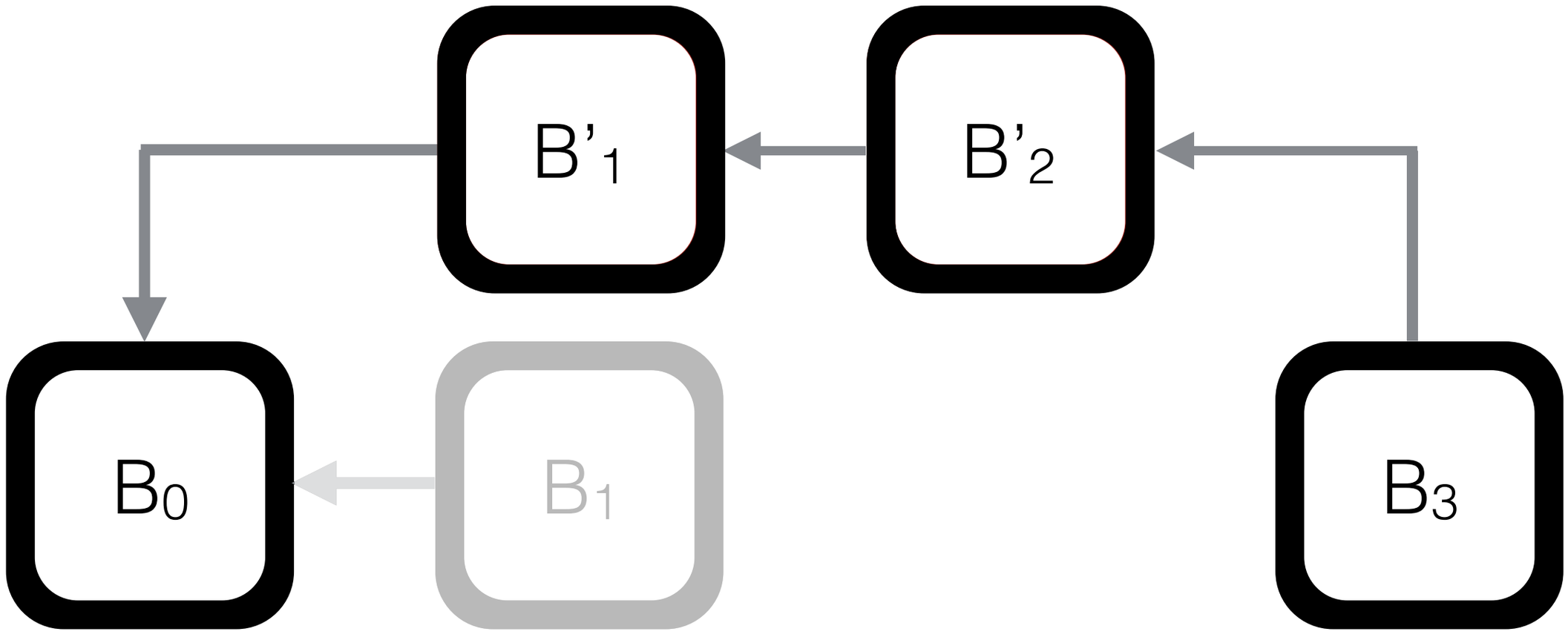
- Network partition
- **Stale blocks = lost efforts**

Selfish Mining

Denial of Service

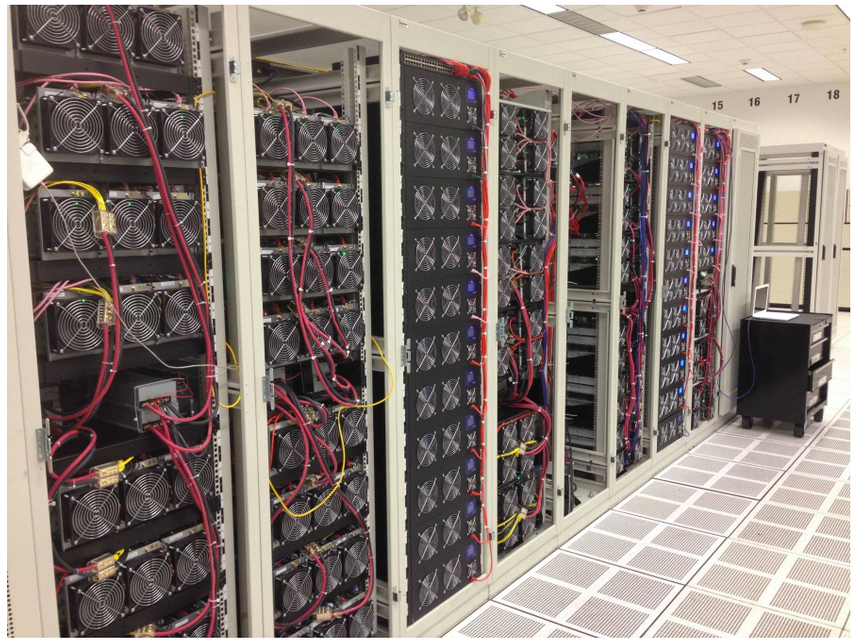
Double Spending

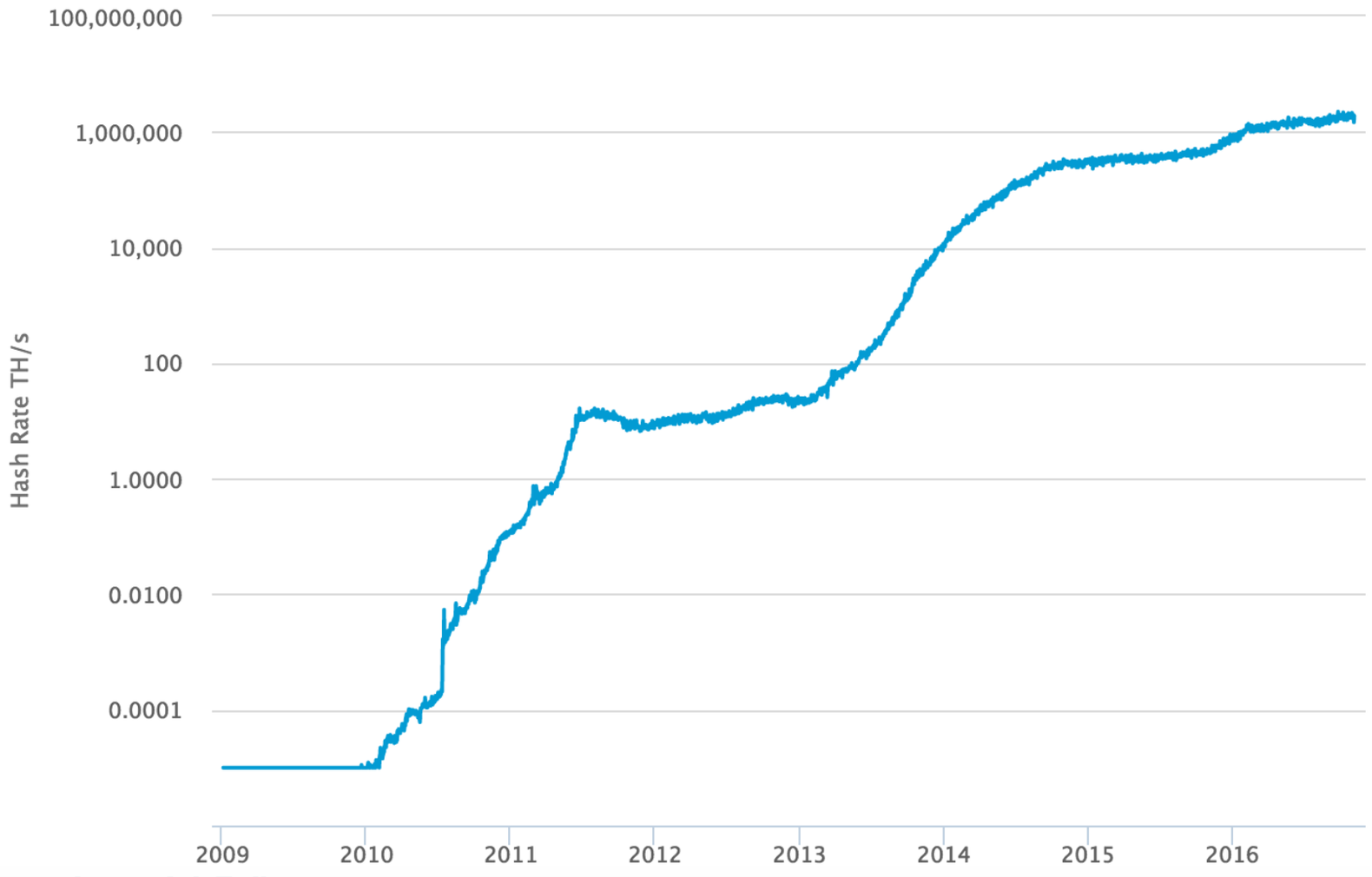
Blockchain Forks

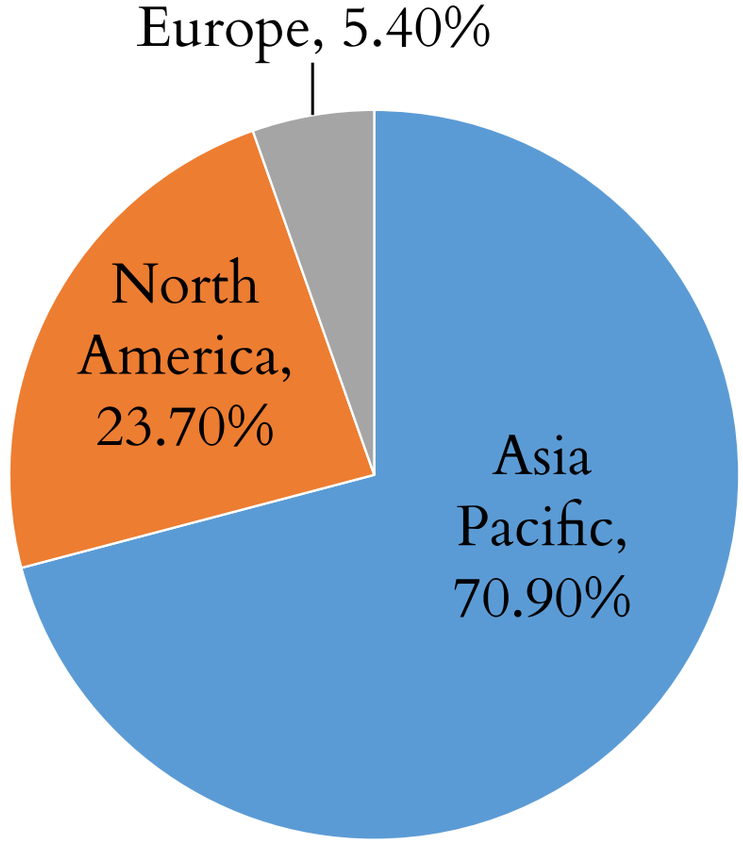


Mining

- From CPU to GPU to ASICs







Imperial College London

Mining Pools

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