Bitcoin, a decentralized and trustless protocol

Thomas Sibut-Pinote

Inria Saclay

February 12, 2015

1. Motivation
   - Introduction
   - Questions
2. Some Required Basics
3. Transactions
4. The Bitcoin Protocol
5. Even cooler stuff

Motivation

In 2008, an anonymous programmer going by the pseudonym of Satoshi Nakamoto invented Bitcoin, a decentralized digital payment protocol without third party which is also a currency.

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

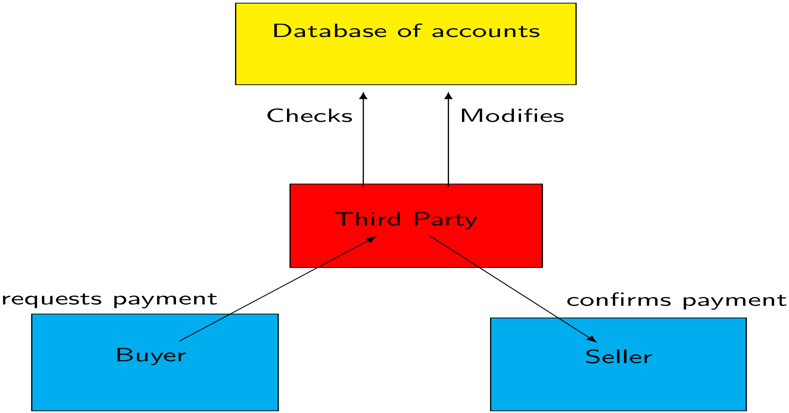
Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot

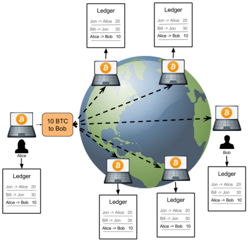
Why decentralized?

How do other digital payment networks work?



What does Bitcoin do differently?

- Bitcoin uses peer-to-peer technology to operate with no central authority or banks;
- Managing transactions and the issuing of bitcoins is carried out collectively by the network;
- It is open-source; its design is public, nobody owns or controls Bitcoin and everyone can take part;



Potential advantages

- An open protocol allows for innovation
- No censorship (example: Wikileaks)
- Instantaneous transactions across the globe
- Transactions are cheap
- Robust against attacks
- With unique properties, Bitcoin allows exciting uses that could not be covered by any previous payment system.

Some of the questions we have to address

- How do we get consensus between all the peers?
- How does the protocol prevent someone from spending someone else's bitcoins?
- How does the protocol prevent 'double spend'?
- How are bitcoins created?

1 Motivation

2 Some Required Basics
- Cryptographic Hashing Functions
- Some uses of hash functions
- Dual Key Cryptography

3 Transactions

4 The Bitcoin Protocol

5 Even cooler stuff

Hashing functions

Let U be a universe of keys and n be an integer. A hashing function is a function

$$h : U \to \{0, \cdots, n-1\}$$

Another way to say it:
It maps digital data of possibly arbitrary size to digital data of fixed size.

Hashing functions

Let U be a universe of keys and n be an integer. A hashing function is a function

$$h : U \to \{0, \cdots, n-1\}$$

Another way to say it:
It maps digital data of possibly arbitrary size to digital data of fixed size.

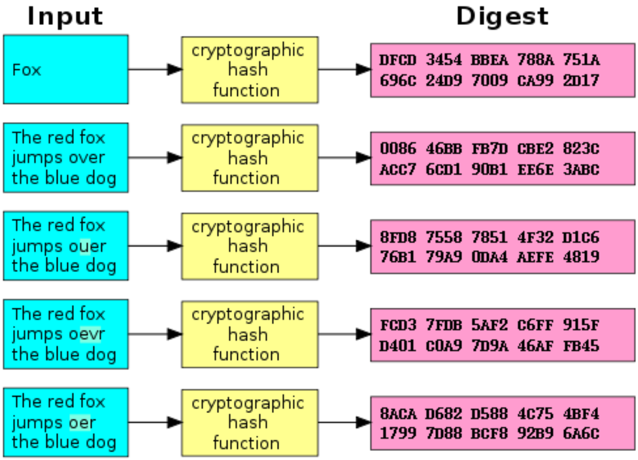
Example: $SHA256 : \{0,1\}^* \to \{0, 1, \cdots, 2^{256} - 1\}$ maps any sequence of bits to a binary integer of size 256.
SHA256("bitcoin")=
6b88c087247aa2f07ee1c5956b8e1a9f4c7f892a70e324f1bb3d161e05ca107b (hex)

Cryptographic Hashing Functions

Cryptographic Hashing Functions are hashing functions with the following informal properties:

- h is "easy" to compute.
- h^{-1} is "hard" to compute in the sense that there is no better way to find an antecedent to $y \in \{0,1\}^n$ than to try all inputs;
- It is infeasible to modify an x without dramatically changing $h(x)$;
- It is infeasible to find $x \neq y$ such that $h(x) = h(y)$.

Illustration (SHA1)



Unique identifier

For all intents and purposes, a hash of a file can be thought of as a unique digital identifier of this file, i.e morally 'if two files have the same hash, they are identical'.

Some uses

Here we think of h as

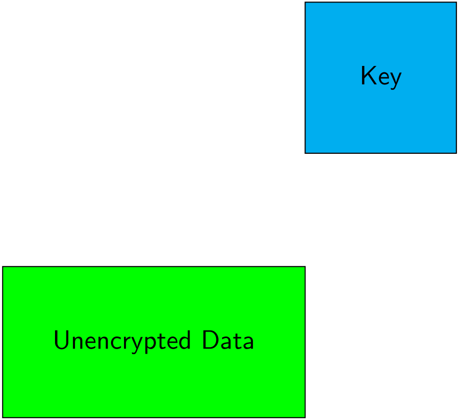
$$h : \{0,1\}^* \to \{0,1\}^n$$

Some uses of cryptographic hash functions are:

- Checking that a file has not been tampered with or corrupted during a transfer. *Hash of torbrowser : a06ad5dbbfe4f53e49edb4064cfbe275727b1e98*
- Keeping someone busy for a certain amount of time on average by asking them for a y such that $h(x, y)$ begins with d zeros. (used in Bitcoin). This is called a *proof of work*.
- Proving knowledge of information without disclosing it (application of Bitcoin). *I know the answer to the math homework but I won't tell you, here is the proof: 3d902580c053c4edb2ccdc3edb7c70806ed03bb4*

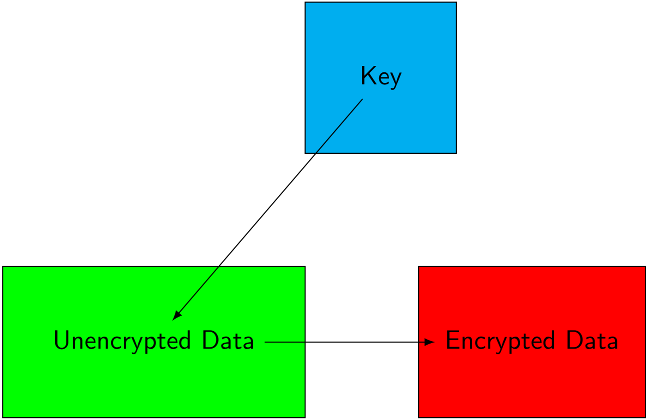
Single Key Encryption

Principle: use a key to encrypt data.



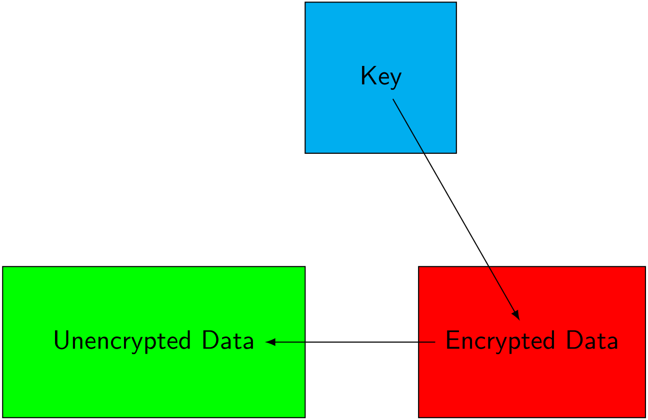
Single Key Encryption

Use the key to encrypt.



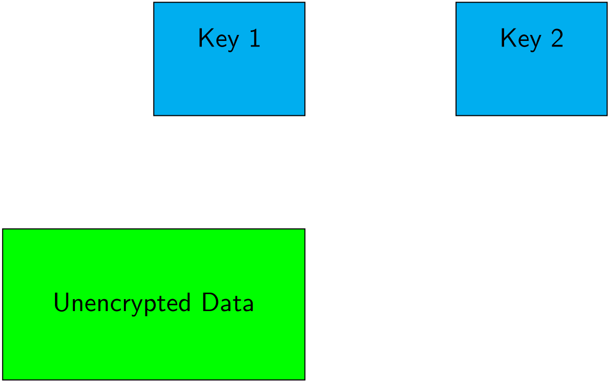
Single Key Encryption

Use the same key to decrypt.



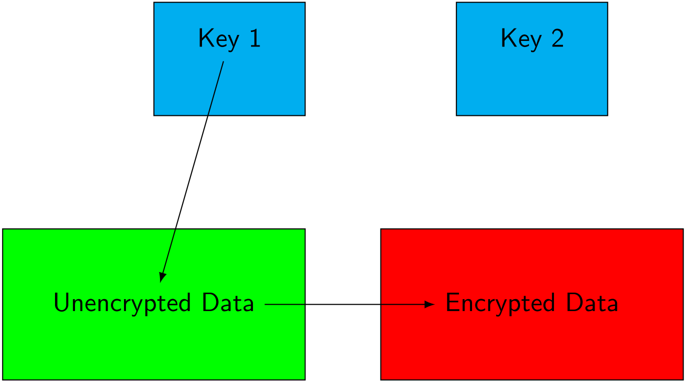
Dual Key Encryption

Now we have two keys, generated together.



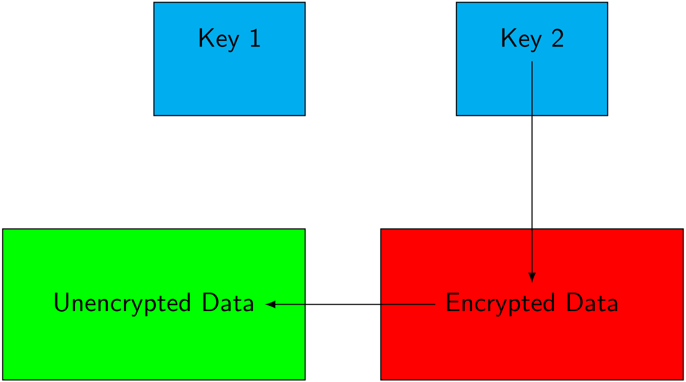
Dual Key Encryption

Encrypt with key 1.



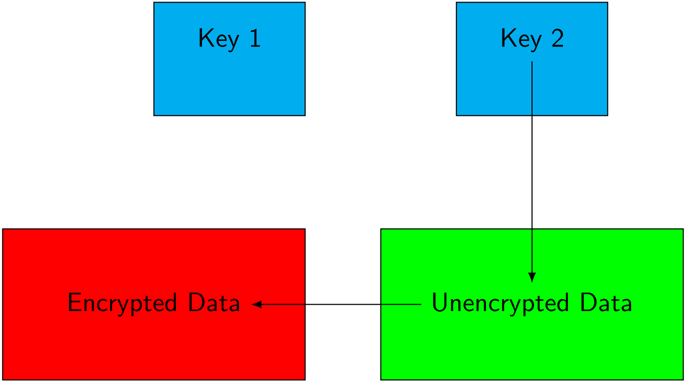
Dual Key Encryption

Decrypt with key 2.



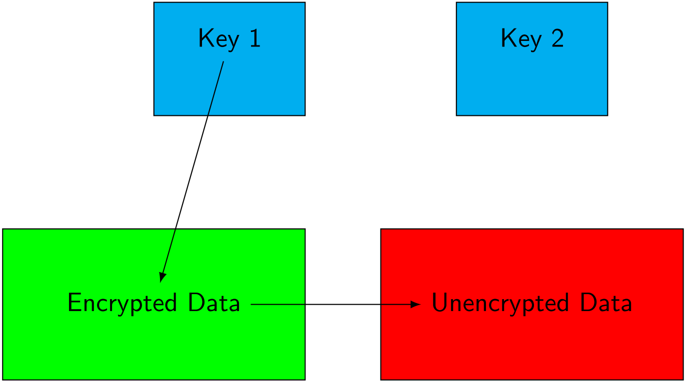
Dual Key Encryption

Or symmetrically, encrypt with key 2.



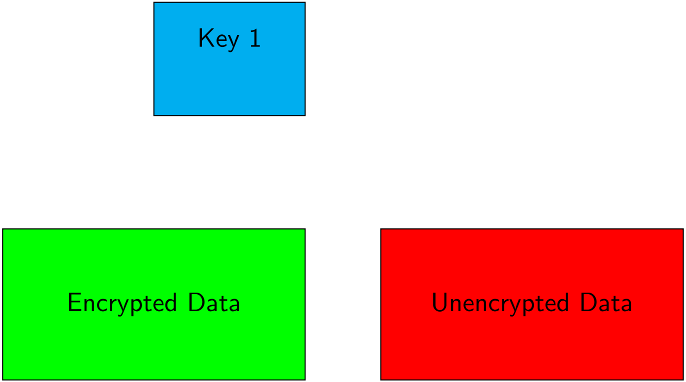
Dual Key Encryption

.. and then decrypt with key 1.

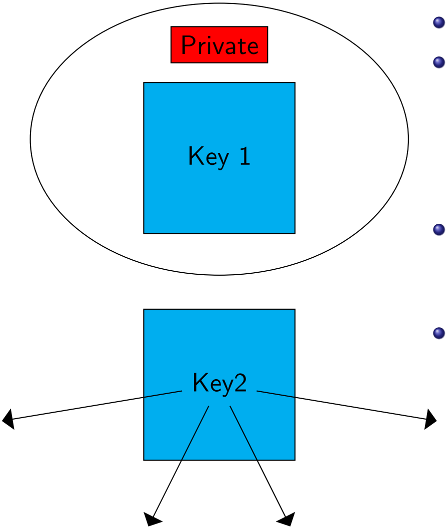


Dual Key Encryption

With only one key, you can't unencrypt your own data.



Public/Private Key



- Keys are still symmetrical;
- A private key is kept secret so that only you can read the messages you receive; even the person who wrote them cannot decrypt them later!
- Your public key is given out publicly so people can write to you.
- Encrypting with Key 1 is called **signing**: it certifies your **identity**.

1 Motivation

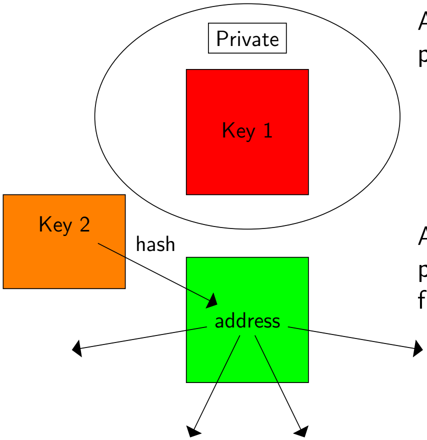
2 Some Required Basics

3 Transactions
- Application to transactions

4 The Bitcoin Protocol

5 Even cooler stuff

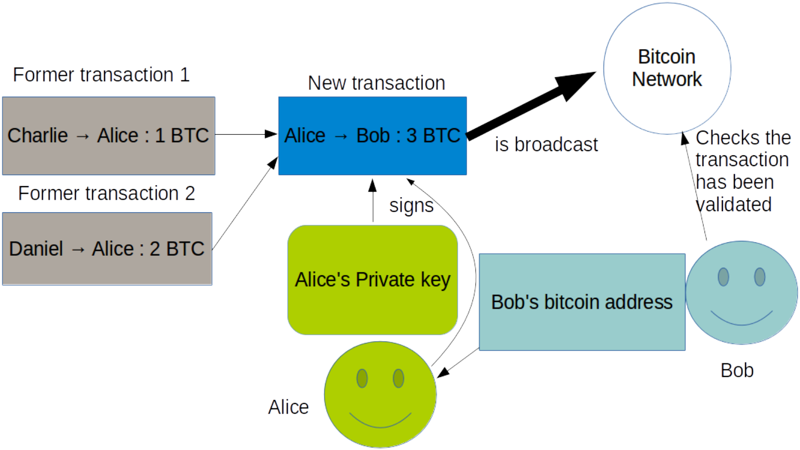
Accounts and addresses



A Bitcoin account is a (private key, public key) pair.

A Bitcoin address is a hash of a public key, and is used to receive funds.

Principle of a transaction



What is the expressiveness of transactions?

Transactions can express complex conditions.
They are not Turing-complete, but for example they can:

- talk about the time: 'this transaction is not redeemable until May 1^{st} 2017'
- use basic cryptographic primitives, basic math primitives 'this transaction is redeemable if you can decrypt this message'
- use hashing functions, among which SHA-256 : 'this transaction is redeemable to anyone producing an x such that $h(x) = y$'

1 Motivation

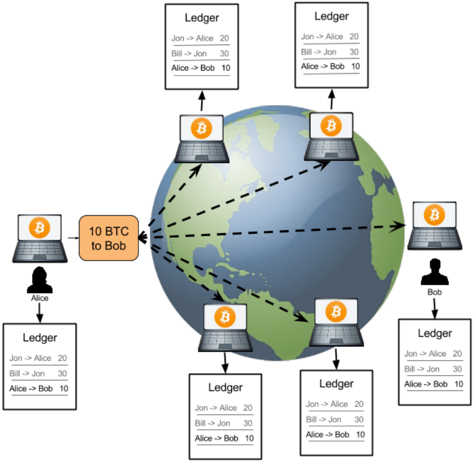
2 Some Required Basics

3 Transactions

4 The Bitcoin Protocol
- The Bitcoin Network
- The blockchain, a shared ledger of transactions
- How new blocks are added
- Why does it work?
- The 51% attack

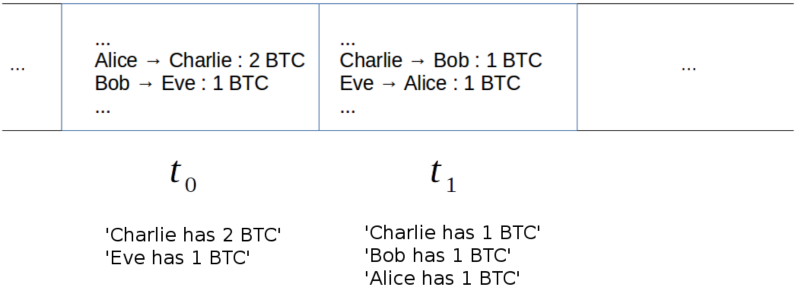
5 Even cooler stuff

Now that we've seen the idea of how Alice can create a transaction to Bob, we need to look at the global picture.



Bitcoins do not exist; transactions do.

The Bitcoin network of peers (or nodes) cooperate to establish a consensus on a set of transactions ordered in time called the blockchain. This set is constantly being built.



Those transactions **track the ownership of bitcoins**.

Consensus

- The blockchain is a chain of blocks of transactions which is shared across all members of the Bitcoin network.
- The blockchain is the result of a *consensus* among the peers and *nothing else*. There is no central authority approving or discarding anything.

How does one add a new block?

- In theory, anyone can try to add a block: those who do are called miners
- They receive transactions which are broadcast across the network and aggregate them into a new block
- Now, they need to play a kind of lottery to earn the right for their block to be on the blockchain: they have to build a **proof of work**, which essentially consists in making some heavy computations that take time. **The more computing power they have, the higher their probability of winning is.**

But why?

Why is proof of work necessary?

- It maintains a constant rate of emission of new bitcoins: difficulty is adjusted every two weeks so that a block is added on average every ten minutes;
- It makes it easy to spot the 'right' blockchain: it is the longest one **because more computation power must have been devoted to building it**.

Proof of work under the hood

Suppose that

- k_n is the hash of the previous block on the current blockchain;
- th is the hash of all the transactions in the block being built;
- d is an integer called the *difficulty*.

Finding a valid proof of work is finding a y such that $h(k_n, th, y)$ begins with d zeros. The properties of CHF make this a kind of *lottery*.

Proof of work under the hood

Suppose that

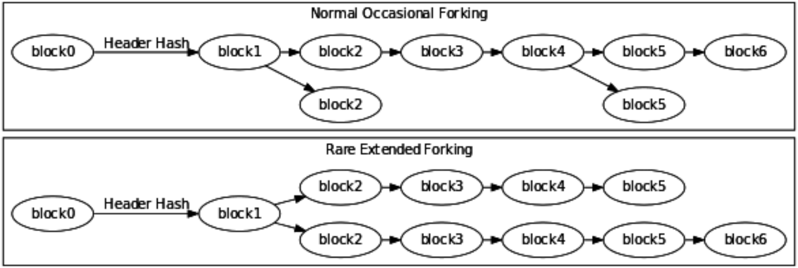
- k_n is the hash of the previous block on the current blockchain;
- th is the hash of all the transactions in the block being built;
- d is an integer called the *difficulty*.

Finding a valid proof of work is finding a y such that $h(k_n, th, y)$ begins with d zeros. The properties of CHF make this a kind of *lottery*.

The first miner to find a block with a valid proof of work gets a reward of 25 bitcoins as the block is added to the blockchain. **This is how bitcoins are created!**

Couldn't there be two longest blockchains?

For a short period of time, there could be *two or more* 'longest blockchains' for example if two miners mine a block at the same time.



but this situation has an exponentially decreasing probability of persisting through time.

The 51 % attack

If someone comes to possess more than 50 % of the total computing power of the network, they can

- reject blocks from other miners and still mine every single block with probability 1 in the long run;
- reject specific transactions or transaction types or senders to receivers permanently;
- rewrite history as far as they want in theory, in the close past in practice.

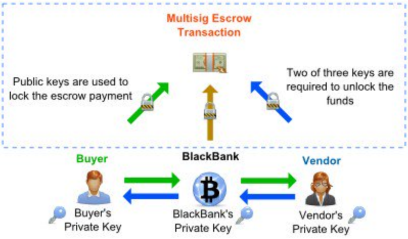
If they have slightly less (30 %, 40 %) they can also do those things with a little help from chance but not with probability 1.
However, no one can:

- Steal bitcoins from someone (unless they rewrite history before those bitcoins were mined);
- Create bitcoins out of thin air or at a higher asymptotic rate than planned by the protocol.

1. Motivation
2. Some Required Basics
3. Transactions
4. The Bitcoin Protocol
5. Even cooler stuff
    - Multisig and escrow
    - Namecoin
    - Beyond

Multisig and escrow

- A transaction can provide *m*-of-*n* checking: 'to spend this transaction, you must have at least *m* signatures from these *n* public keys'
- This provides the possibility of escrow (unfortunate homonym with French)



Namecoin

Namecoin is a decentralized open source information registration and transfer system based on the Bitcoin cryptocurrency.

- Securely record and transfer arbitrary names (keys).
- Attach a value (data) to the names (up to 520 bytes, more in the future).
- Transact namecoins, the digital currency ($\mathbb{N}$, NMC).

What's the point?

- It can be used as a decentralized, independent DNS
- It could be used to store identity information such as email, GPG key, BTC address, TLS fingerprints..
- It can be used (and Bitcoin too) to timestamp events: if the hash of a document was put on the blockchain in April 2014, then surely someone had that document at that time.

And many other things

- Bitshares $\rightarrow$ BITUSD, BITEUR
- Ethereum: Turing-complete transactions
- Alternatives to POW: Proof-of-Stake
- Counterparty: creating tokens on top of the blockchain (example: LTBcoin)

Thanks!