Security Engineering (8)

Email: christian.urban at kcl.ac.uk
Office: S1.27 (1st floor Strand Building)

Slides: KEATS (also homework is there)

Last Week's Survey

About Bitcoins:

bitcoins are anonymous

not regulated by any government

Should one mine for Bitcoins?

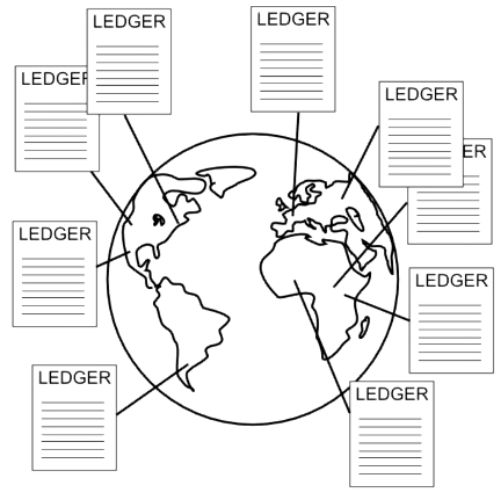
untracable spending of money?

bitcoins cannot get lost, all transactions are recorded

fixed amount of bitcoins in circulation (no inflation)

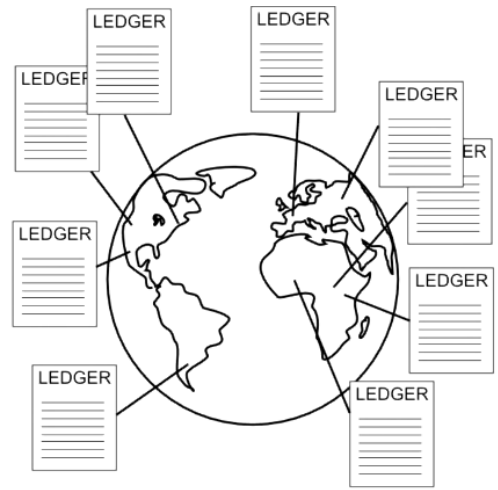
Bitcoins from 10,000m

- a crypto "currency" by Satoshi Nakamoto (likely a pen name)
- a digital resource designed to be scarce (max 21 Mio bitcoins—deflationary currency)
- mined by solving special puzzles involving hashes
- transaction history (ledger/blockchain) is P2P distributed (12 GB)
- three "mining pools" produce currently more than 50% of bitcoins
- can be stolen and also lost
- anonymous?



Bitcoins from 10,000m

- a crypto "currency" by Satoshi Nakamoto (likely a pen name)
- a digital resource designed to be scarce (max 21 Mio bitcoins—deflationary currency)
- mined by solving special puzzles involving hashes
- transaction history (ledger/blockchain) is P2P distributed (12 GB)
- three "mining pools" produce currently more than 50% of bitcoins
- can be stolen and also lost
- anonymous?
- surely a scam/ponzi scheme!



Bitcoins

- you create a public-private key pair
- you have a 'wallet' which can be
  - electronic (on your computer, passwords)
  - cloud-based (passwords)
  - paper-based

  and contains only the public-private key
- Bitcoins can be stolen or lost
- Mt. Gox: hacked $\Rightarrow$ insolvent
- no form of dispute resolution
  (against current consumer laws)

Underlying Ideas

It establishing trust in a completely untrusted environment

- public-private key encryption
- digital signatures
- cryptographic hashing (SHA-256)

If Alice sends you: $msg, \{msg\}_{K^{priv}_{Alice}}$ ...?

Lets Start with "Infocoins"

$$\{\text{I, Alice, am giving Bob one infocoin.}\}_{K^{priv}_{Alice}}$$

- no-one else could have created that message
- Alice cannot deny the "intend" of sending Bob money

Lets Start with "Infocoins"

$$\{\text{I, Alice, am giving Bob one infocoin.}\}_{K^{priv}_{Alice}}$$

- no-one else could have created that message
- Alice cannot deny the "intend" of sending Bob money

- Q: What is money?
  A: Well a string like above (or later messages like that)

Double Spend

$$\{\text{I, Alice, am giving Bob one infocoin.}\}_{K^{priv}_{Alice}}$$

- Alice could keep sending Bob this message over and over again (did she mean to send 10 ICs?)

Double Spend

$$\{\text{I, Alice, am giving Bob one infocoin.}\}_{K^{priv}_{Alice}}$$

- Alice could keep sending Bob this message over and over again (did she mean to send 10 ICs?)
- we need to have a serial number

$$\{\text{I, Alice, am giving Bob infocoin \#1234567.}\}_{K^{priv}_{Alice}}$$

Double Spend

$$\{\text{I, Alice, am giving Bob one infocoin.}\}_{K^{priv}_{Alice}}$$

- Alice could keep sending Bob this message over and over again (did she mean to send 10 ICs?)
- we need to have a serial number

  $$\{\text{I, Alice, am giving Bob infocoin \#1234567.}\}_{K^{priv}_{Alice}}$$
- but then we need a trusted source of serial numbers (e.g. a bank)

No Banks Please

With banks we could implement:

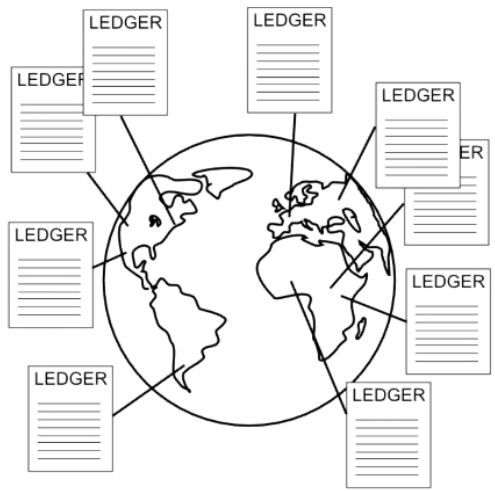
- Bob asks the bank whether the infocoin with that serial number belongs to Alice and
- Alice hasn't already spent this infocoin.

- If yes, then Bob tells the bank he accepts the infocoin.
- The bank updates the records (ledger) to show that the infocoin with that serial number is now in Bob's possession and no longer belongs to Alice.

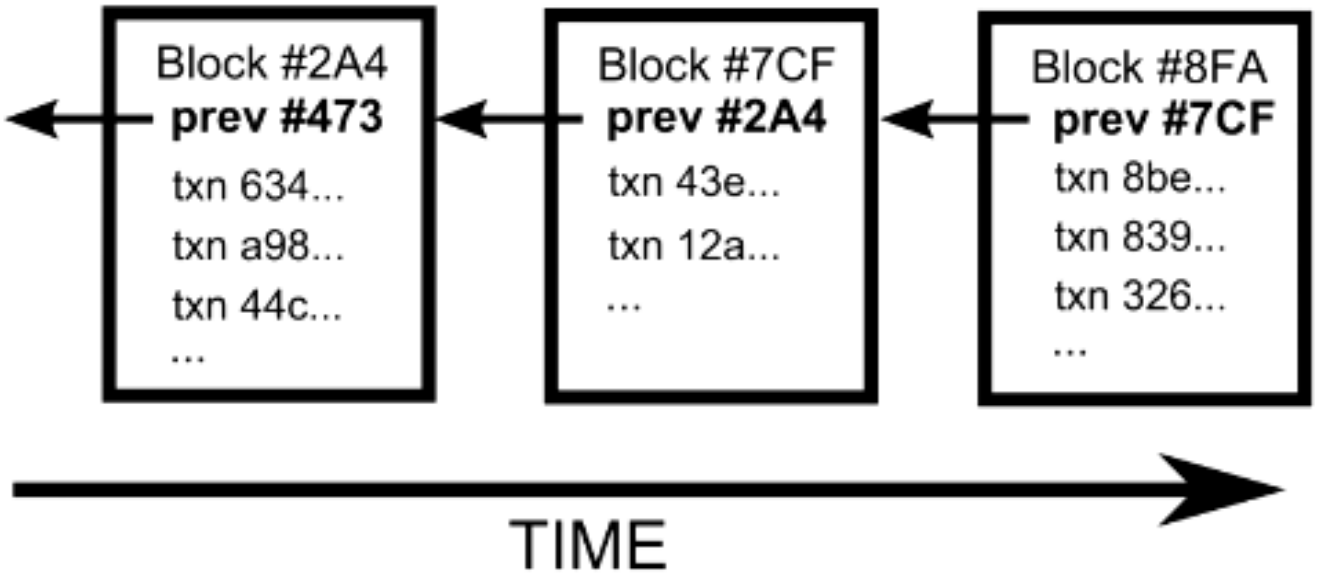
Blockchain (Public Ledger)

The solution for double spend:

- make everybody the bank, everybody has the entire transaction history — will be called **blockchain**
- Bob checks whether the infocoin belongs to Alice and then broadcasts the message to everybody else



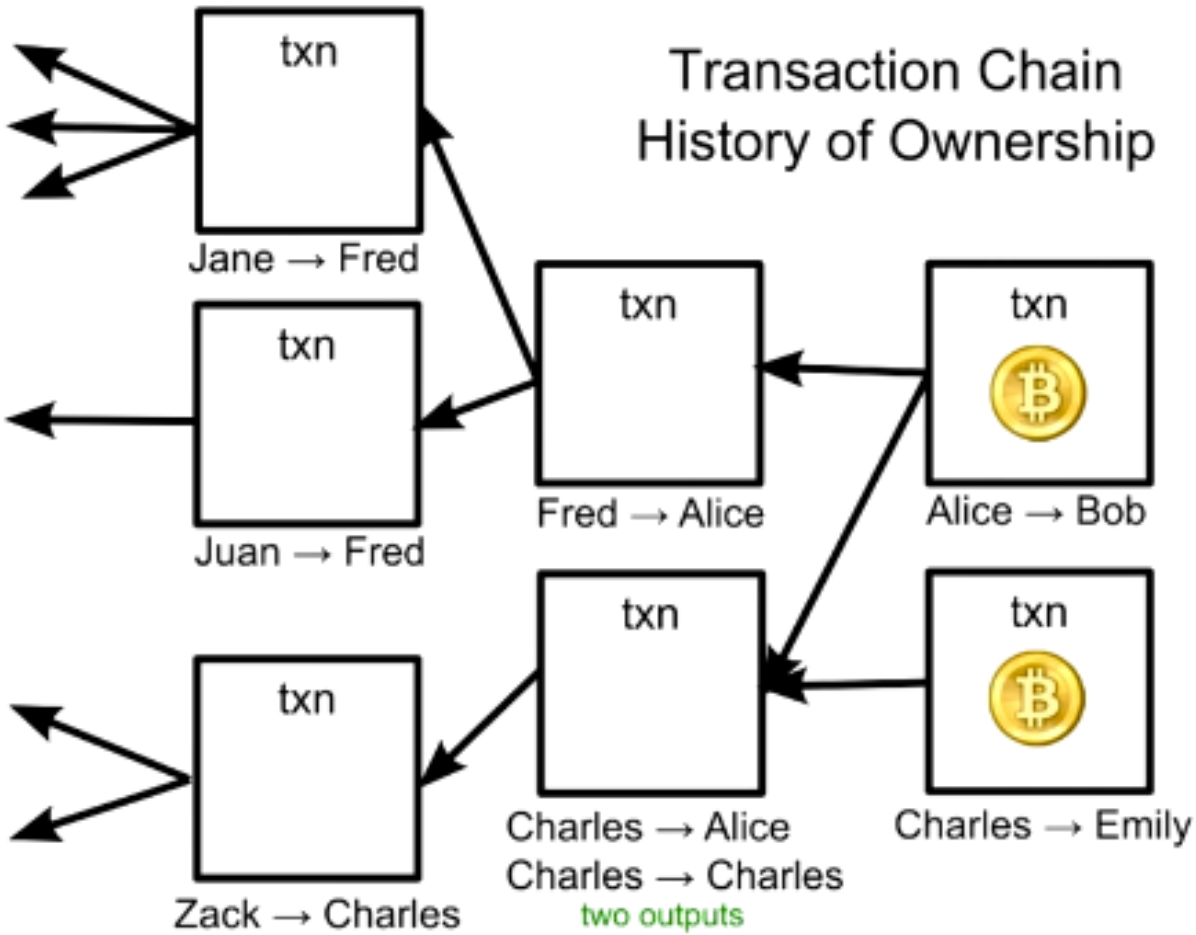
Blockchain (Public Ledger)



- each block is hashed and contains a reference to the earlier block; “validates” potentially more than one transaction

Transaction Graph

older current



Double Spend Again

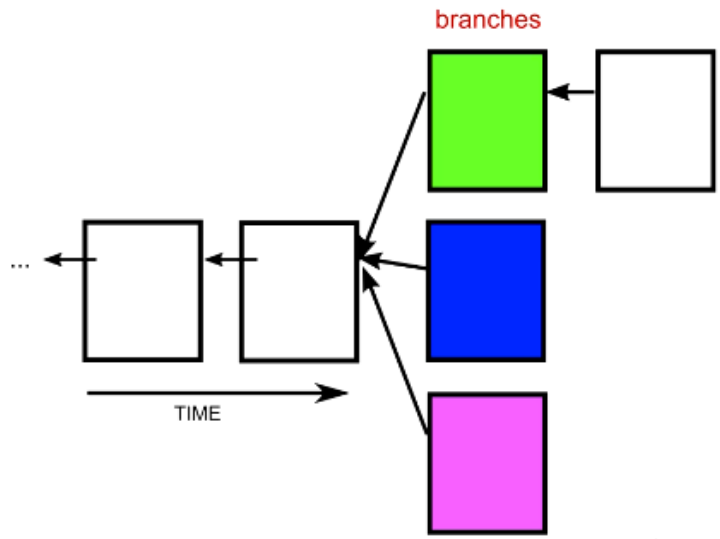
- I , Alice, am giving Bob one infocoin, with serial number 1234567.
- I, Alice, am giving Charlie one infocoin with number 1234567.

How should other people update their blockchain (public register)?

Double Spend Again

- I , Alice, am giving Bob one infocoin, with serial number 1234567.
- I, Alice, am giving Alice one infocoin with number 1234567.

How should other people update their blockchain (public register)?



Creating Agreement

Once enough people have broadcast that message, everyone updates their block chain to show that infocoin 1234567 now belongs to Bob, and the transaction is accepted.

Creating Agreement

Once enough people have broadcast that message, everyone updates their block chain to show that infocoin 1234567 now belongs to Bob, and the transaction is accepted.

But what if Alice sets up a large number of separate identities, let's say a billion, on the Infocoin network. When Bob asks the network to validate the transaction, Alice's puppet identities say "Yes his transaction is validated", while actually the rest network says Alice's transaction is OK?

!! Proof-of-Work !!

The idea is counterintuitive and involves a combination of two ideas:

- to (artificially) make it computationally costly for network users to validate transactions, and
- to reward them for trying to help validate transactions

!! Proof-of-Work !!

The idea is counterintuitive and involves a combination of two ideas:

- to (artificially) make it computationally costly for network users to validate transactions, and
- to reward them for trying to help validate transactions

this is called mining: whoever validates a transaction will be awarded with 50 bitcoins — this halves every 210,000 transactions or roughly every 4 years (currently 25 BC); no new bitcoins after 2140 – then only transaction fees

Solving Puzzles

Given a string, say ”Hello, world!”, what is the salt so the hash starts with a long run of zeros?

```
h(”Hello, world!0”) =

1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
```

Solving Puzzles

Given a string, say ”Hello, world!”, what is the salt so the hash starts with a long run of zeros?

```
h(”Hello, world!0”) =
1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
h(”Hello, world!1”) =
e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
```

Solving Puzzles

Given a string, say "Hello, world!", what is the salt so the hash starts with a long run of zeros?

```
h("Hello, world!0") =
1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
h("Hello, world!1") =
e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
...
h("Hello, world!4250") =
0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9
```