# Security Engineering (6)


Email: christian.urban at kcl.ac.uk
Office: N7.07 (North Wing, Bush House)

Slides: KEATS (also homework is there)

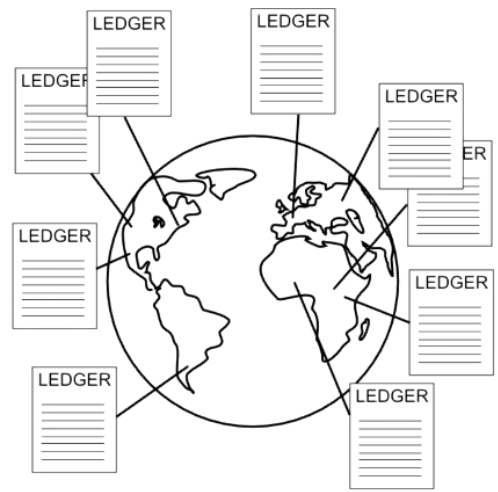
# Over to you...

**What do you know about Bitcoins:**

- blockchain, public (ledger)
- no banks (or trusted party)
- mining, worth a try?
- anonymous
- no gov meddling
- 2009
- there will only a finite amount of bitcoins

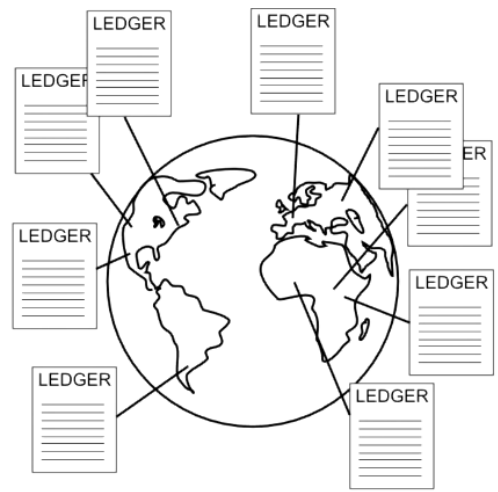
# Bitcoins from 10,000m

- 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 your private key(s)
- 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
- cryptographic hashing (SHA-256)
- digital signatures

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 crypto 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:

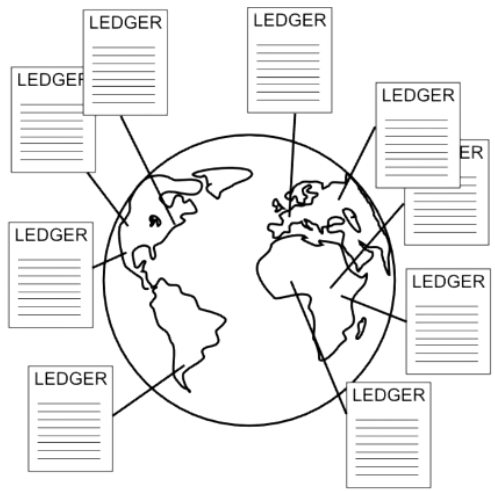
- Bob asks the bank whether the infocoin with that serial number belongs to Alice and
- Alice hasn't already spent this infocoin.

- 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.

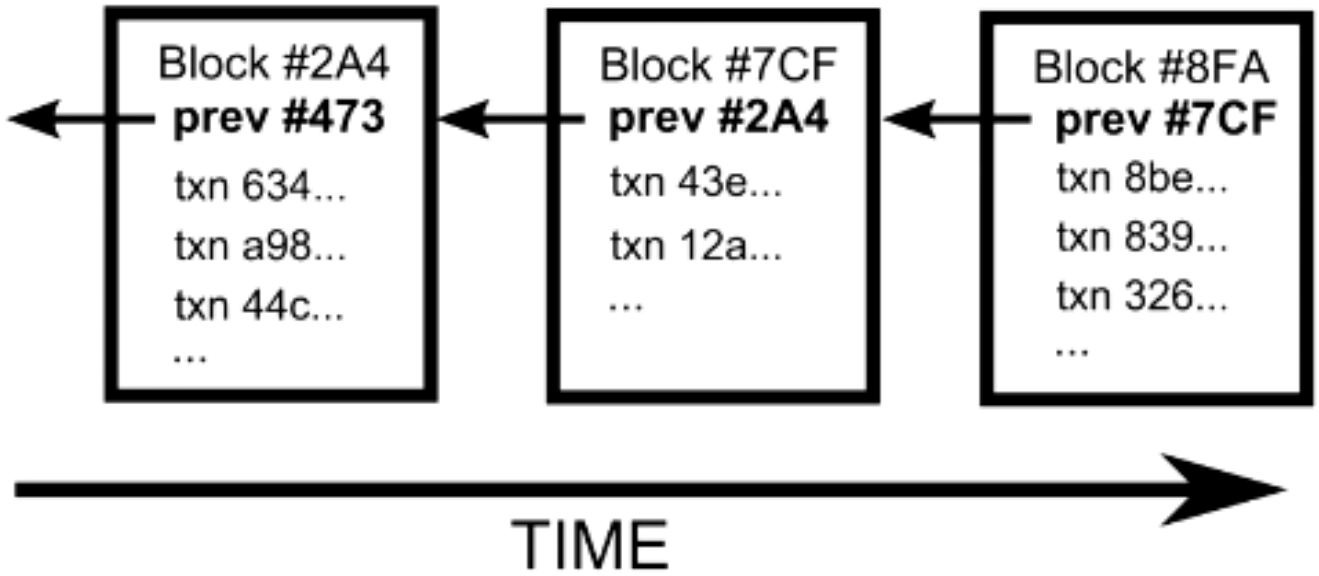
# Blockchain (Public Ledger)

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



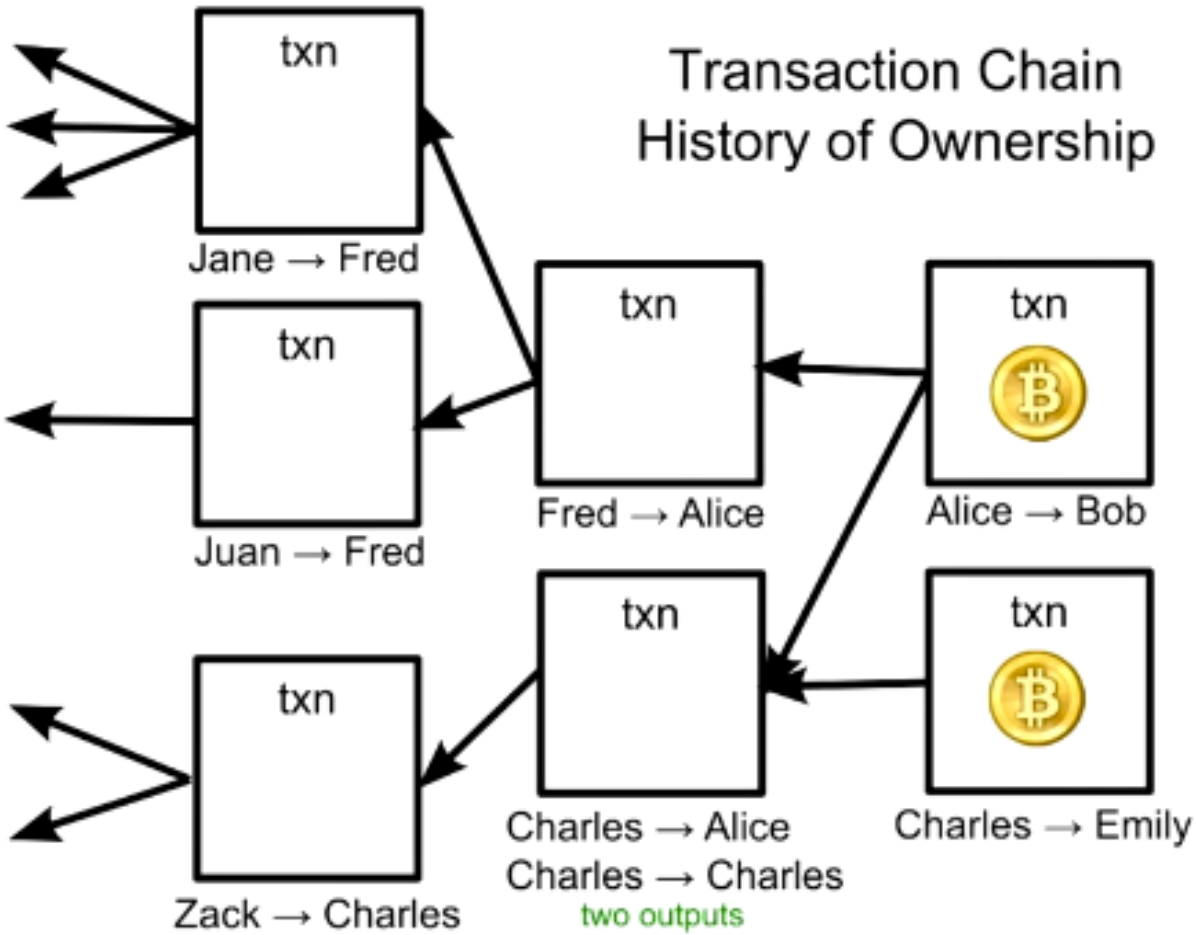
# Blockchain (Public Ledger)




- 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

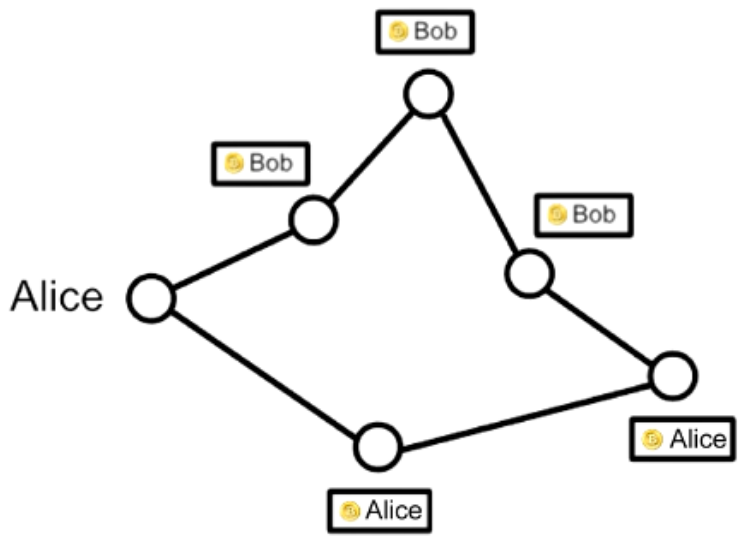
- I , Alice, am giving Bob one infocoin, with serial number 1234567.
- I, Alice, am giving Charlie one infocoin with number 1234567.

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 that 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 that the hash starts with a long run of zeros?

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

# Solving Puzzles

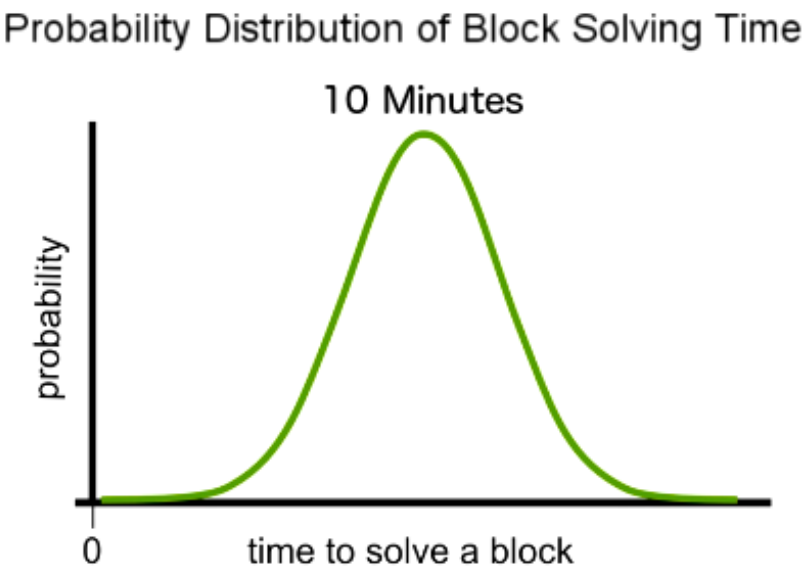
Given a string, say ”Hello, world!”, what is the salt so that the hash starts with a long run of zeros?

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

# Hardness

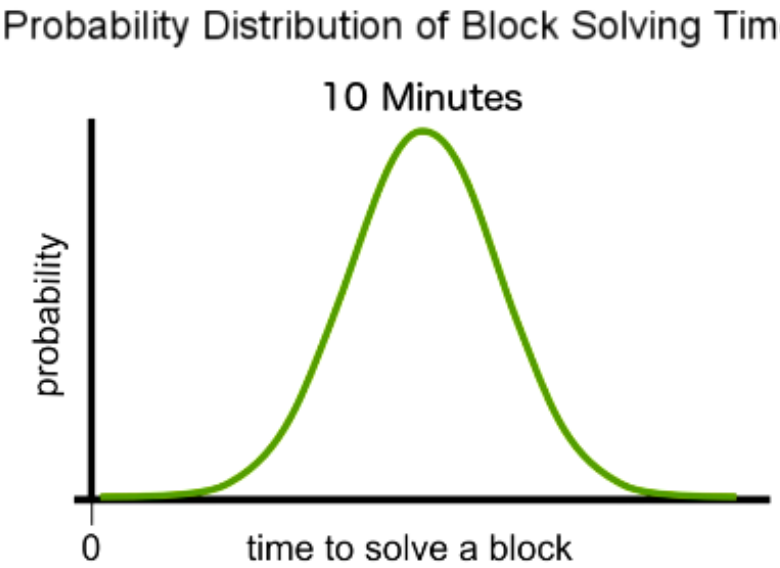
If we want the output hash value to begin with 10 zeroes, say, then we will need, on average, to try $16^{10} \approx 10^{12}$ different salts before we find a suitable salt.

Hardness can be controlled by setting a target (maximum number).



# How to Adjust the Target?

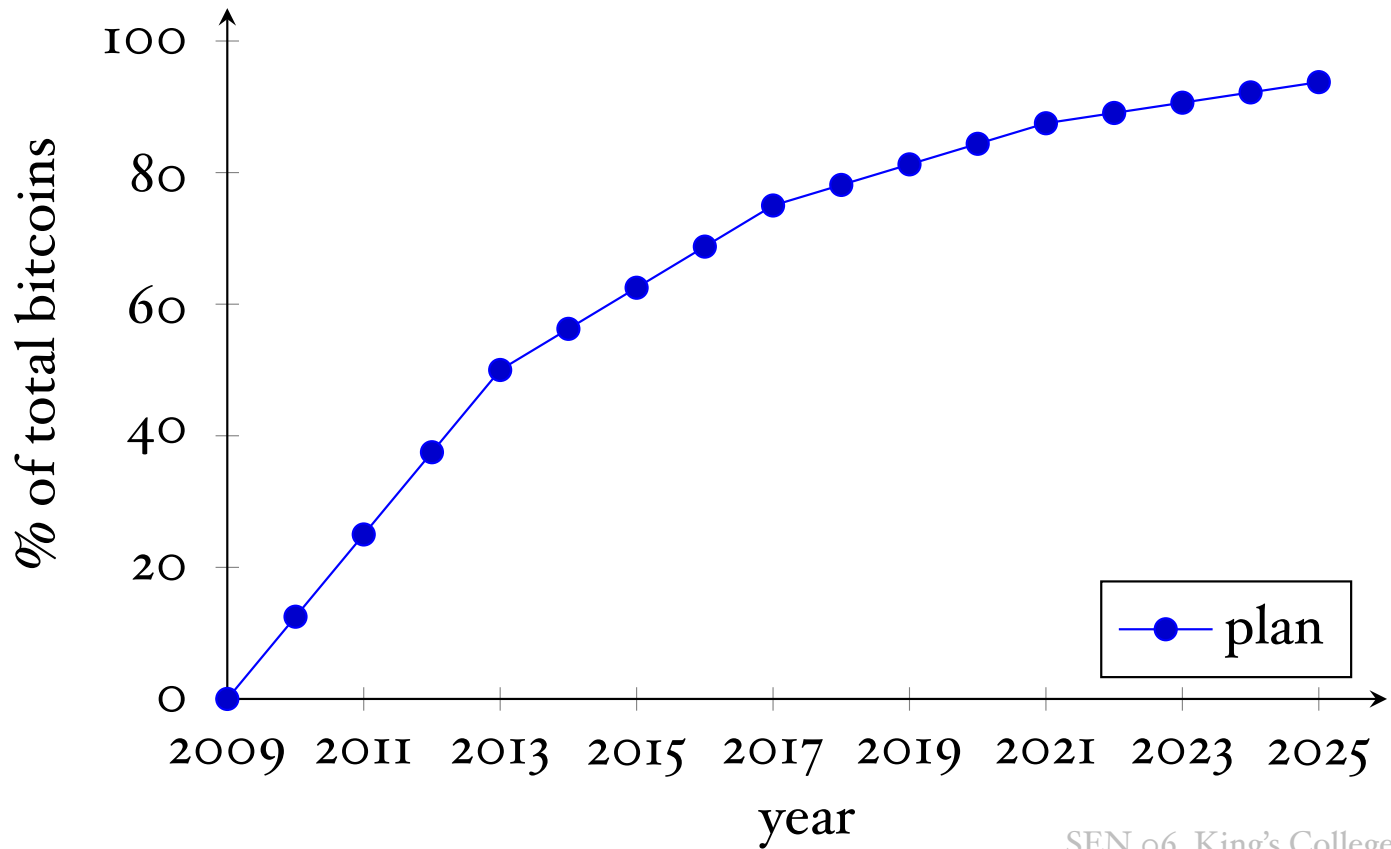
- every 2016 blocks the hardness is adjusted (app 2 weeks)




$$\text{New Hardness} \stackrel{\text{def}}{=} \text{Old Hardness} * \frac{\text{Actual time for the last 2016 blocks}}{2016}$$

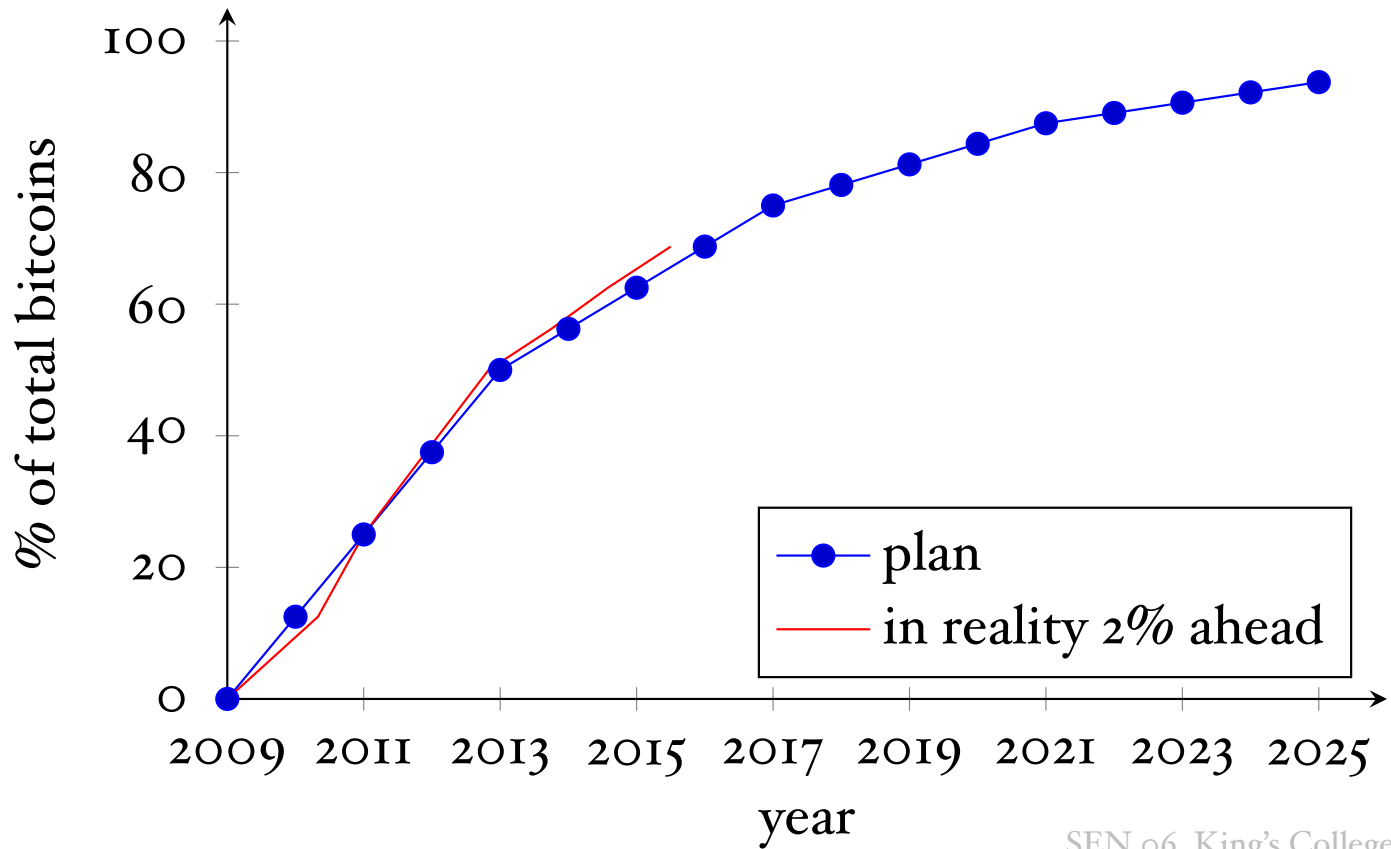
# Bitcoin Schedule

- every 210000 blocks the amount of bitcoins to be mined halves ("reward era")
- every 2016 blocks the hardness is adjusted (app 2 weeks)



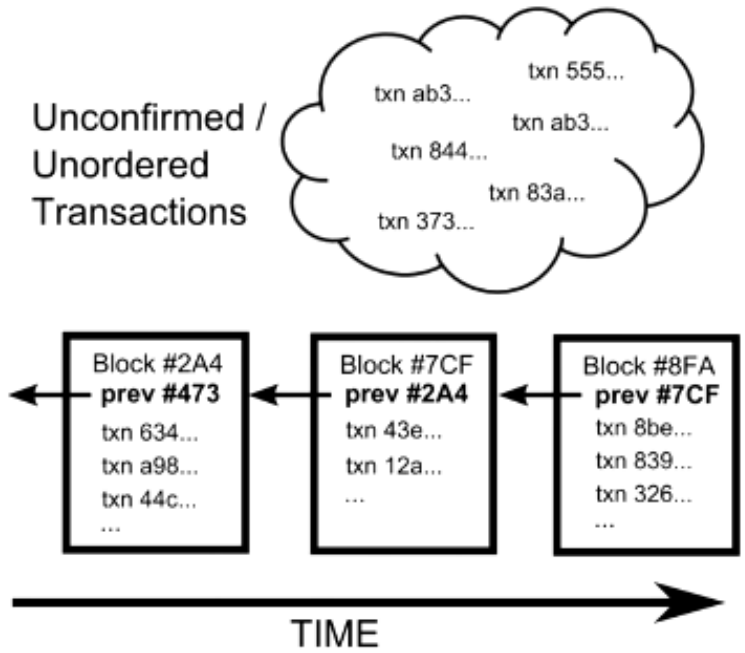
# Bitcoin Schedule

- every 210000 blocks the amount of bitcoins to be mined halves ("reward era")
- every 2016 blocks the hardness is adjusted (app 2 weeks)



# Order of Transactions

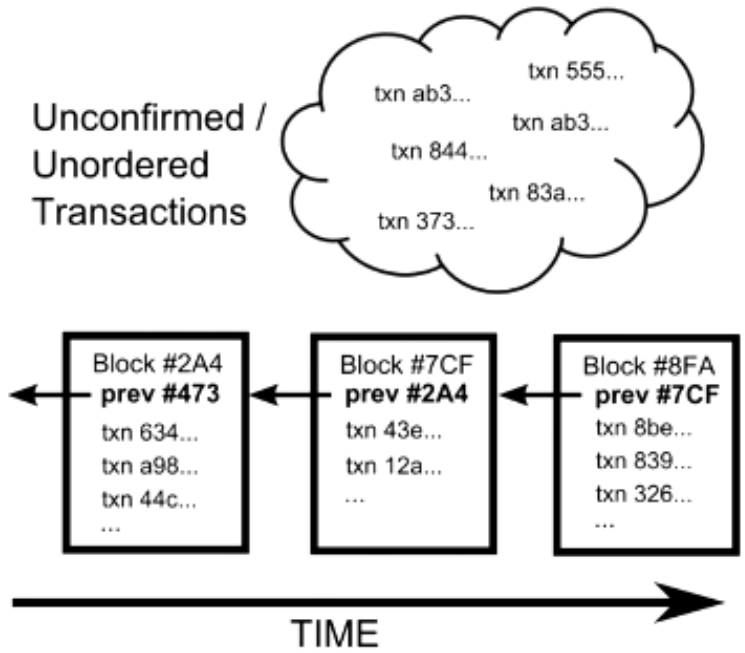
If we don't have such an ordering at any given moment then it may not be clear who owns which Bitcoins.




Say, miner David is lucky and finds a suitable salt to confirm the transactions. Celebration!

# Order of Transactions

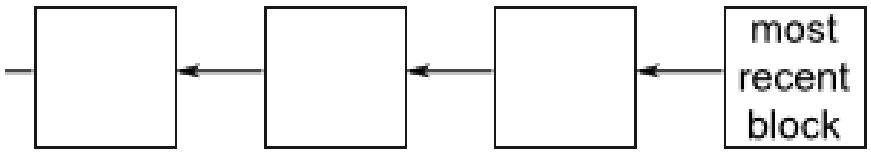
If we don't have such an ordering at any given moment then it may not be clear who owns which Bitcoins.




Say, miner David is lucky and finds a suitable salt to confirm the transactions. Celebration! ??

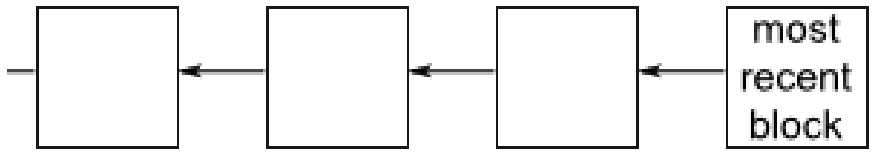
# Forks

Typically the blockchain will look as follows

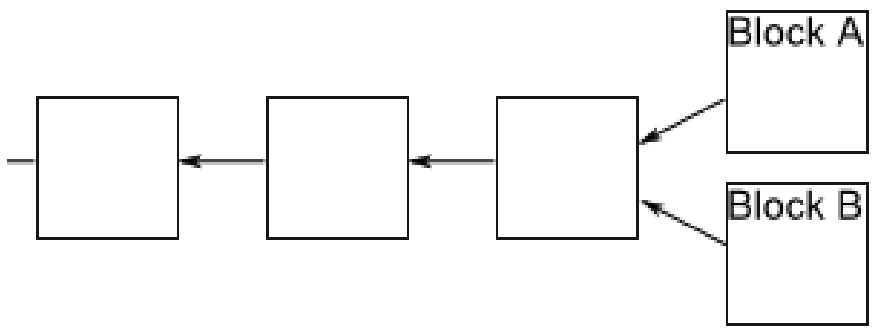


# Forks

Typically the blockchain will look as follows

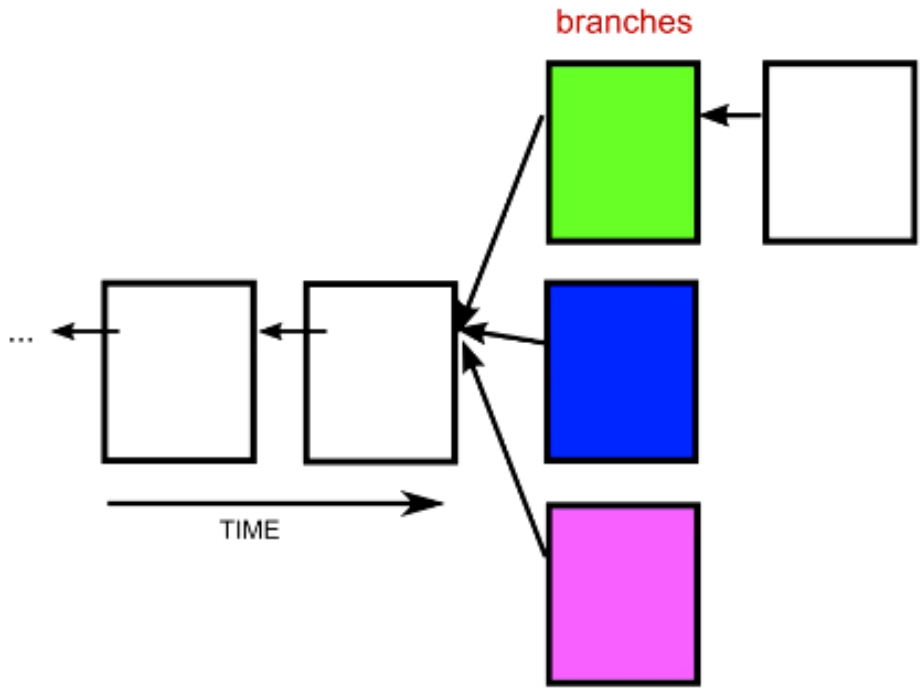



But every so often there is a fork



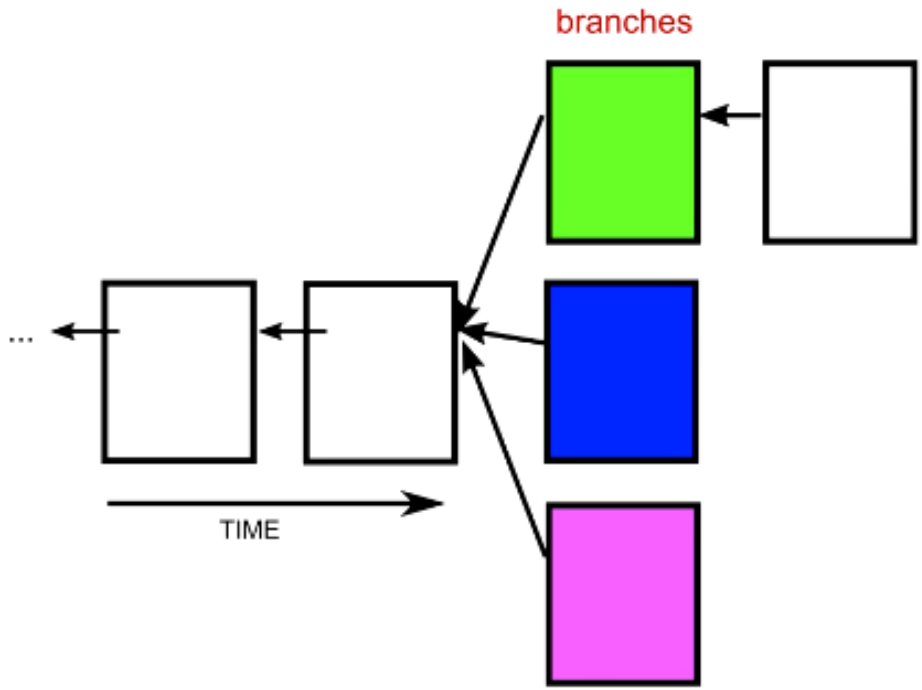

...bugger this is exactly what we are trying to avoid

# The tie is broken if another block is solved




The rule is: if a fork occurs, people on the network keep track of all forks. But at any given time, miners only work to extend whichever fork is longest in their copy of the block chain.

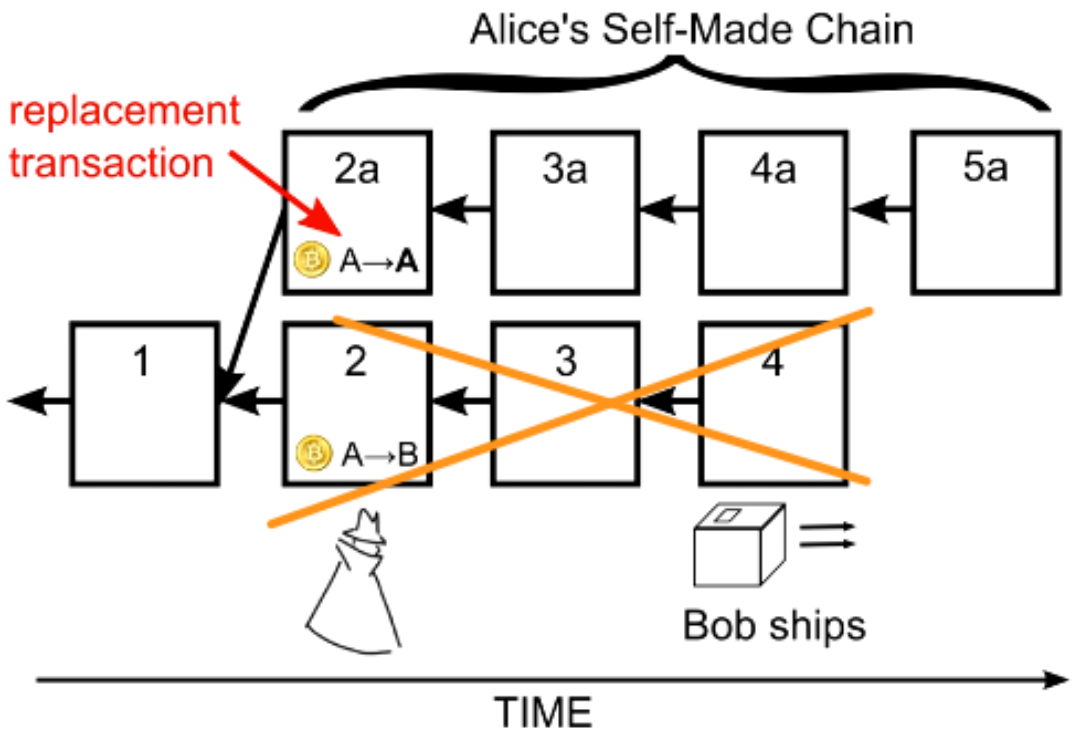
# The tie is broken if another block is solved



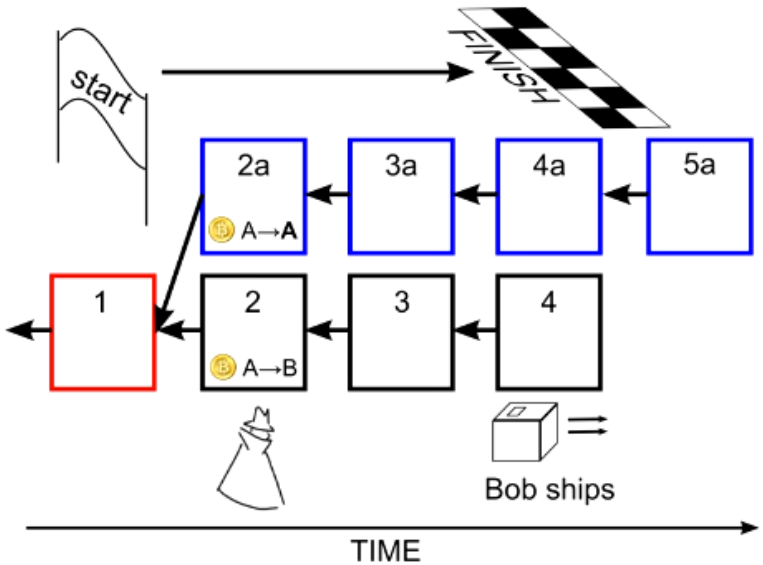

The rule is: if a fork occurs, people on the network keep track of all forks. But at any given time, miners only work to extend whichever fork is longest in their copy of the block chain. (It is actually not a rule, but an incentive!)

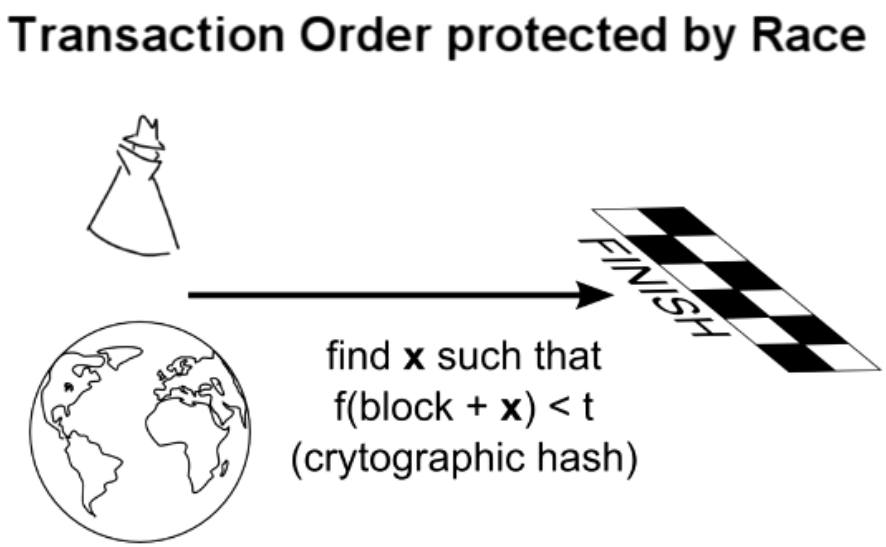
# Double Spending Again

So if Alice wants to fake it, she needs to produce a longer chain:

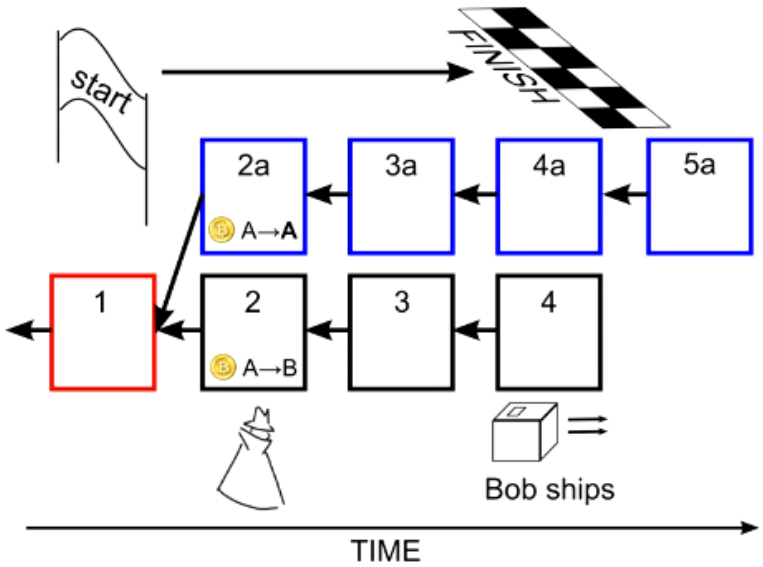


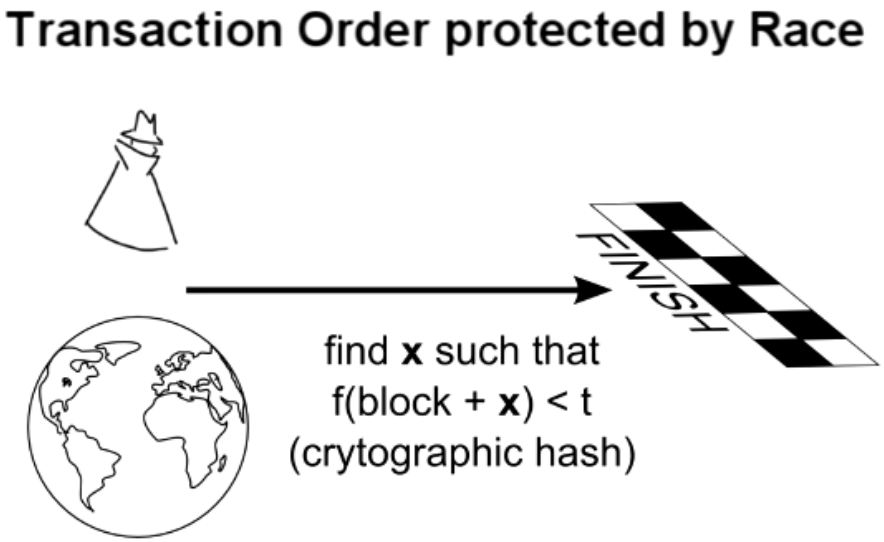
# Racing Against the World



# Racing Against the World




By convention, a transaction is "confirmed" if:

(1) it is part of a block in the longest fork, and (2) at least 5 blocks follow it in the longest fork. In this case we say that the transaction has "6 confirmations".

(might take 1h+...but for creditcards you have 6 months chargeback)

# Mining Pools

On average, it would take several years for a typical computer to solve a block, so an individual's chance of ever solving one before the rest of the network, which typically takes 10 minutes, is negligibly low.

# Mining Pools

On average, it would take several years for a typical computer to solve a block, so an individual's chance of ever solving one before the rest of the network, which typically takes 10 minutes, is negligibly low.

Many people join groups called mining pools that collectively work to solve blocks, and distribute rewards based on the work contributed. These act somewhat like lottery pools among co-workers, except that some of these pools are quite large, and comprise more than 50% of all the computers in the network.

BTCC, the largest mining pool, has limited its members to not solve more than 6 blocks in a row. https://blockchain.info/pools

# Bitcoins for Real

- you need a public-private key (the hash of the public key to determines your bitcoin address)
- if you want to receive bitcoins, you publicise this address
- there are $2^{160}$ possibilities
  (no check for duplicates)

# Bitcoins for Real

- you need a public-private key (the hash of the public key to determines your bitcoin address)
- if you want to receive bitcoins, you publicise this address
- there are $2^{160}$ possibilities
  (no check for duplicates)

- transactions contain "payment scripts"
  (non-Turing-complete scripting language)

  simplest script: pay-to-public-key

# Multi-Signature Addresses

- ...Bitcoin Improvement Proposal

# Multi-Signature Addresses

- ...Bitcoin Improvement Proposal

- pay-to-public-key (explained so far)
- pay-to-script-hash (since 2012)

# Multi-Signature Addresses

- ...Bitcoin Improvement Proposal

- pay-to-public-key (explained so far)
- pay-to-script-hash (since 2012)

  can specify: requires M out of N signatures

  for example
  1-of-2: me and my wife, or

  2-of-2 in banking/companies

# Dispute Mediation

- say, a client and an (online) merchant do not trust each other

# Dispute Mediation

- say, a client and an (online) merchant do not trust each other

- 2-of-3: mutually trusted escrow service
  1. client sends money to 2-of-3 transaction
  2. merchant sends out goods
  3. if goods are OK, client sends signed transaction to merchant, merchant can sign and receive the money (publish in blockchain)
  4. if goods are defective, and merchant is nice, merchant sends signed transaction to client, client can sign and receive the money back
  5. if client and merchant disagree, then they ask escrow service who signs a transaction and sends it to "winning" party

# A Transaction Msg

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```

# A Transaction Msg

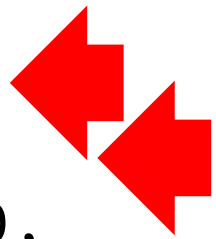

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```


the hash of the msg that follows; kind of serial number

# A Transaction Msg

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```

# A Transaction Msg



the transaction has one inout and one output (could be more)

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```

# A Transaction Msg

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```

# A Transaction Msg

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```

# A Transaction Msg

```
{"hash":"7c4025...",
 "ver":1,
 "vin_sz":1,
 "vout_sz":1,
 "lock_time":0,
 "size":224,
 "in":[
    {"prev_out":
      {"hash":"2007ae...",
       "n":0},
       "scriptSig":"304502... 042b2d..."}],
 "out":[
   {"value":"0.31900000",
    "scriptPubKey":"OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG"}]}
```

the hash of the incoming transaction (incoming serial number)

# A Transaction Msg


```
{"hash":"7c4025...",
 "ver":1,
 "vin_sz":1,
 "vout_sz":1,
 "lock_time":0,
 "size":224,
 "in":[
    {"prev_out":
      {"hash":"2007ae...",
       "n":0},
       "scriptSig":"304502... 042b2d..."}],
 "out":[
   {"value":"0.31900000",
    "scriptPubKey":"OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG"}]}
```


use the 0th output of the incoming transaction

# A Transaction Msg

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```

the public key and signature of the sender

# A Transaction Msg

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```

use $x$ amount of the incoming money

# A Transaction Msg

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```

public key of the receiver

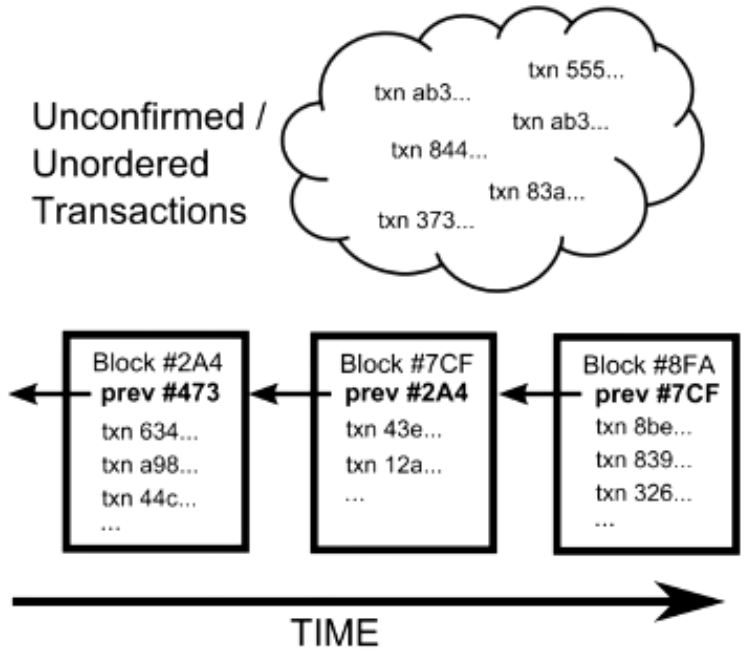
# A Transaction Msg

```
{”hash”:”7c4025...”,
 ”ver”:1,
 ”vin_sz”:1,
 ”vout_sz”:1,
 ”lock_time”:0,
 ”size”:224,
 ”in”:[
    {”prev_out”:
      {”hash”:”2007ae...”,
       ”n”:0},
       ”scriptSig”:”304502... 042b2d...”}],
 ”out”:[
   {”value”:”0.31900000”,
    ”scriptPubKey”:”OP_DUP OP_HASH160 a7db6f...
                   OP_EQUALVERIFY OP_CHECKSIG”}]}
```

you do not need a central authority to issue serial numbers

there are no “coins”, just a long series of transactions

# A Block in the Blockchain




- each block is hashed and contains a reference to the earlier block
- contains the "salt" and address of whoever solved the puzzle

# Transaction History

you can follow back the transaction history until you reach either

- the genesis block (a transaction without input of 50 bitcoins), or
- a coinbase transaction (this is the reward of the miner who validated a block of transactions in the blockchain)

# Lost Bitcoins?

- somebody needs to be able to generate a key-pair for the signature (for this you need the private key)

- somebody spends your bitcoins fraudulently (you cannot charge them back)... bad luck
- you can send bitcoins to a "non-existing" address (Mt. Gox)

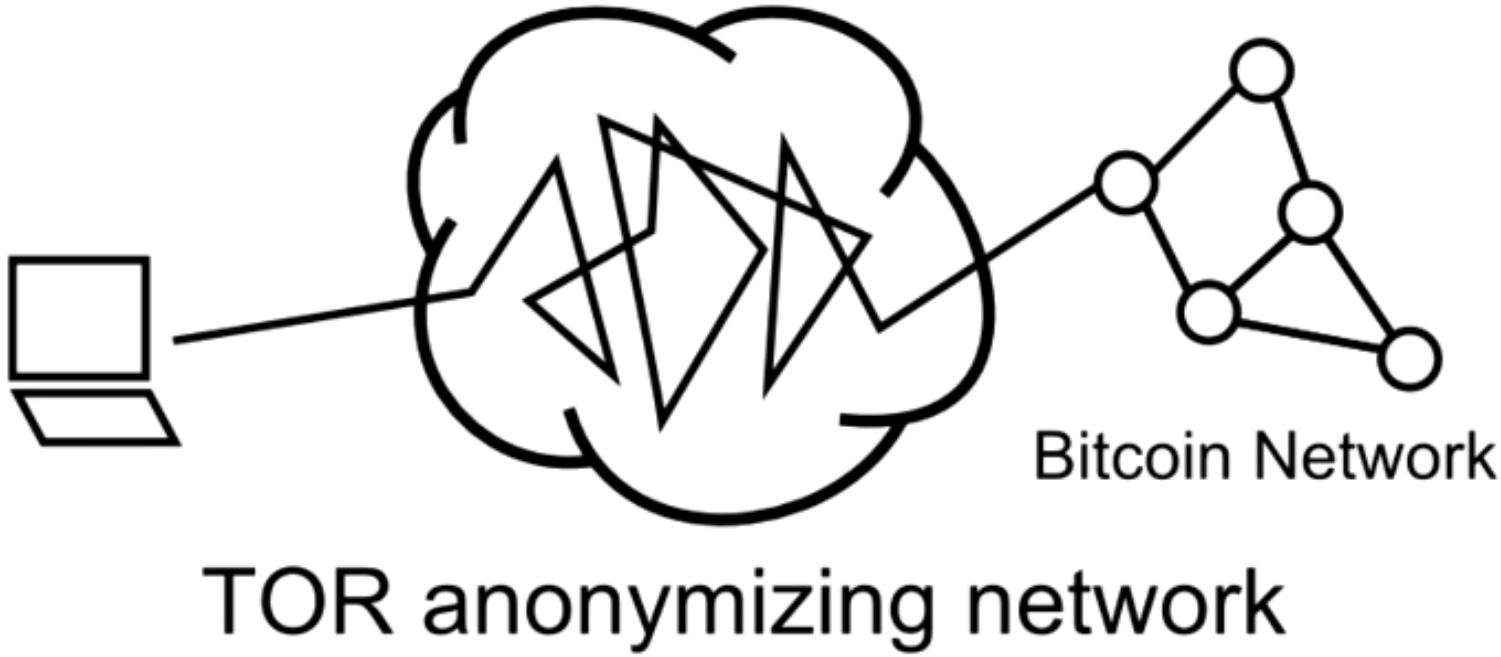
# Good Points

An attacker can't:

- reverse other people's transactions
- change the number of coins generated per block
- create coins out of thin air
- send coins that never belonged to an attacker
- you cannot meddle with the "history"

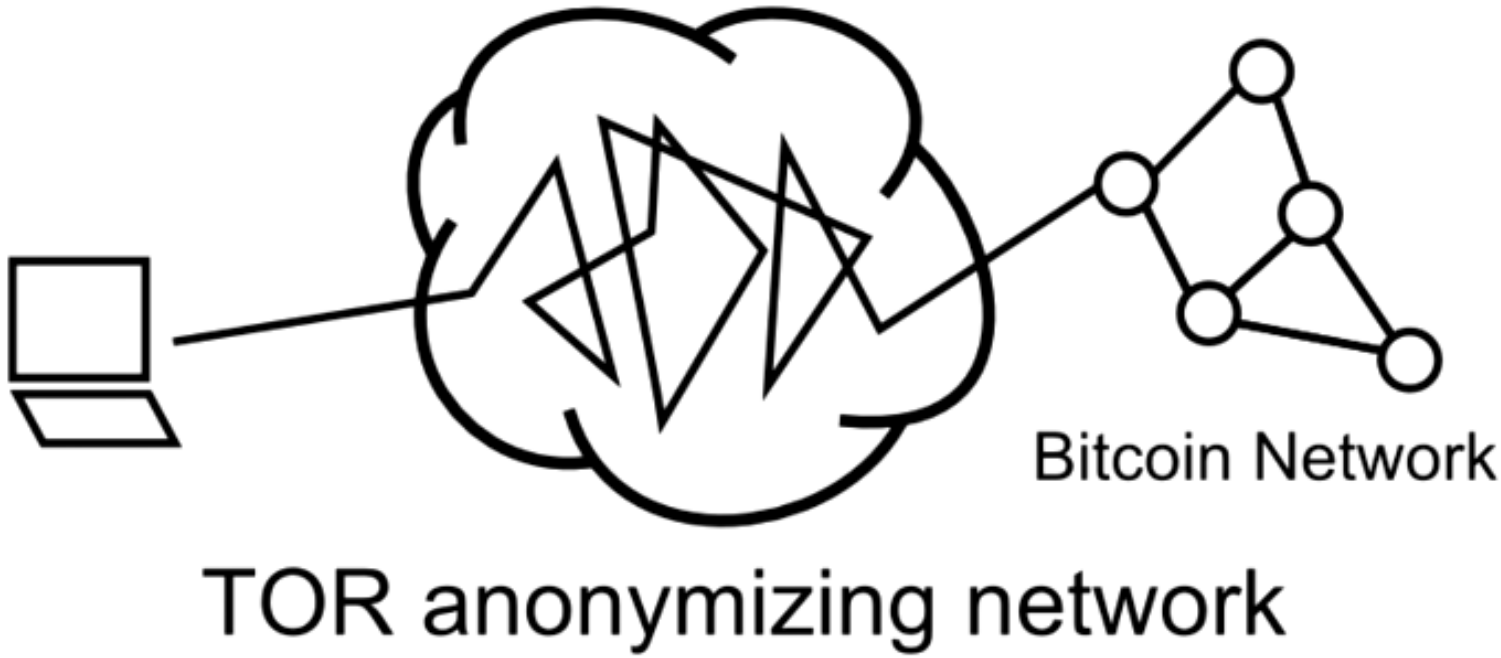
The system can be scaled to all world transactions??

# But I did not Inhale...




the ledger is public “forever”; everybody can inspect how money was transferred from which address to which address; maybe not ideal for money laundering

# But I did not Inhale...




You should use a new PKI-pair for every transaction; but few do (merchants). A design flaw(?): combining transactions.

# Anonymity

"How anonymous is Bitcoin? Many people claim that Bitcoin can be used anonymously. This claim has led to the formation of marketplaces such as Silk Road (and various successors), which specialize in illegal goods. However, the claim that Bitcoin is anonymous is a myth. The block chain is public, meaning that it's possible for anyone to see every Bitcoin transaction ever. Although Bitcoin addresses aren't immediately associated to real-world identities, computer scientists have done a great deal of work figuring out how to de-anonymize 'anonymous' social networks. The block chain is a marvellous target for these techniques."

# Other Cryto-Currencies

Coloured Coins:

- record a 1 bitcoin transfer, say, and give it a "colour"
- the transaction stands for a share in "Foo Inc, UK".
- only makes sense if the share is much more valuable than just 1 Bc.

# Bitcoin vs Gov

Purported absence of potential government interference?

# Bitcoin vs Gov

Purported absence of potential government interference? Far from it:

- government could compel "mayor players" to blacklist bitcoins (exchanges)
- coerce developer community (e.g. Lavabit)
- put pressure on mining pools, or be big a miner itself

# Take Home Points

- Don't gamble! I am not a first mover in such things.
- Cool idea, but I am sure there will be a Bitcoin 2.0.
- It still depends on a lot of old-fashioned security (e.g. keeping private-keys secret)

- Having now the knowledge how it works, go back and listen to what people/media make of it.

# Next 4 weeks by Jose Such

You can still send me homework for weeks 1 - 6.