

Introduction to Blockchains

33c3

Niklaus 'vimja' Hofer
niklaus@mykolab.ch

December 28, 2016

```
4946 454c 743d 6865 6c70 7461 0a65 4458
5646 4549 4557 2832 3a3a 2428 7328 6568
6c6c 7728 6968 6863 6a20 7975 6864 0a29
618a 6c6c 283a 2824 4946 454c 2a23 6470
8a66 2a8a 4858 4aef 3a55 6320 656e 6e61
638a 656c 6e61 6a3a 5089 6972 2a29 612e
7975 2a29 6c2e 676f 2a28 6a2e 7861 2a29
677e 7475 2a29 732e 686a 2a28 742a 636f
8a6a 582e 4f48 594e 283a 6964 7473 6c63
6165 8a6e 6964 7473 6c63 6165 3a6e 096a
75c 286d 2824 4946 4558 2a29 6470 6165
283a 6976 7765 768a 6569 3a77 2828 2a28
4828 4c49 2945 782e 6864 096a 245c 5828
```

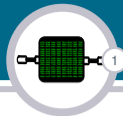
```
2f74 6f66 746e 2f73 7974 6570 2f31 7275
2f77 6568 786c 7465 6369 752f 7668 3872
2e61 6670 3a62 4f8a 7475 7570 2074 7277
7469 6574 296e 6e6f 7420 6065 6c70 7461
2e65 6470 2966 3128 2832 6170 6567 2c73
3328 3235 3838 2932 7962 6574 2973 8a2e
4450 2046 7473 7461 7369 6974 7363 8a2e
3228 3131 5820 4644 6f20 6a62 6365 7374
6f20 7475 6f20 2966 3831 3838 2820 616d
2e78 3820 3833 3638 3738 0a29 3128 3837
6328 6a6f 7270 7365 6573 2064 626f 656e
736e 6f20 7475 6f20 2966 3831 3838 2820
616d 2e78 3520 3838 3838 2938 206e 3431
```

```
7265 7348 6365 6974 6e6f 6170 6567 2073
357b 7b7d 7d38 7d7d 5c0a 7740 6972 6574
6966 656c 6e7b 7661 7b70 685c 6165 6364
6a6f 616d 646e 7b20 625c 6165 656d 4072
7573 7362 6365 6974 6e6f 6170 6567 2073
387b 7b7d 7d38 7d7d 5c0a 7740 6972 6574
6966 656c 747b 636f 7b70 625c 6165 656d
4072 7573 7362 6365 6974 6e6f 6e69 6f74
2963 337b 7b7d 7d31 4c7b 616f 6964 676e
7420 6568 5420 6568 656d 6120 646e 5420
6568 656d 4f20 7470 6f69 736e 7b7d 7d39
387b 7b7d 7d33 0a7d 485c 7277 7469 6665
6c69 7865 616e 7d76 5c7b 6568 6a61 6f63
6a6d 6e61 2964 5c7b 6562 6a61 7265 7348
6275 6573 7463 6f69 656e 746e 7972 7628
7d38 337b 7b7d 7d31 397b 7b7d 6f4c 6461
```

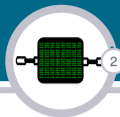
```
0a2e 5c0a 6674 7448 636f 5c3d 7277 7469
3765 5c0a 786f 6e65 756f 3774 3a20 6820
6574 786d 616c 6574 742e 636f 2a27 8a6a
745c 4866 6a73 3a6d 775c 6972 6574 8a38
6f5c 6570 6f6e 7475 2038 283d 746d 6865
6c70 7461 2a65 6a73 276d 8a2e 580a 6361
618a 656f 6120 7b74 7265 6579 646e 4920
666e 3a6f 4520 706d 7974 6820 6f67 286b
4288 6865 726f 4385 656c 7261 6f44 7563
6584 746e 2a27 6a6f 6920 796e 7475 6c20
6e69 2965 3432 2a32 508a 6361 616b 6567
6120 7b74 7265 6579 646e 4920 8a6e 3a6f
4520 706d 7974 6820 6f67 296b 4160 746e
7265 614c 7473 6053 7089 756f 2774 6f20
206e 6e69 7570 2074 696c 656e 3220 3234
2764 6f20 296e 6e69 7570 2074 696c 656e
```

```
7372 6f69 286e 337b 332e 7833 7d74 0a7d
485c 7277 7469 6665 6c69 7865 616e 7d76
5c7b 6562 6a61 7265 6540 646e 8e69 7570
6974 6f66 8874 7265 6576 7372 8f69 206e
337b 332e 7833 7d74 0a7d 735c 6c65 6365
4074 616c 676e 6175 6567 657b 8f6e 896c
6873 0a70 485c 7277 7469 6665 6c69 7865
6f74 7863 5c7b 6573 656c 7463 6c40 8e69
756f 6761 7865 6a65 6c67 7369 7a68 0a70
405c 7277 7469 6665 6c69 7865 6f6c 7a66
5c7b 6573 656c 7463 6c40 6e61 7567 6761
7865 6a65 6c67 7369 7a68 0a70 485c 7277
7489 6665 6c69 7865 6f6c 7d74 5c7b 6573
656c 7483 6c40 6e61 7567 6761 7865 6a65
6c67 7369 7a68 0a70 485c 7277 7469 6665
6c69 7865 616e 7d76 5c7b 6568 6a61 6f63
```

```
8665 6c69 7865 616e 7d76 5c7b 6568 6461
6f63 686d 6e61 2964 5c7b 6562 6a61 7265
7340 6275 6573 7463 6f69 706e 6761 7365
7b20 7d33 347b 7d70 0a7d 485c 7277 7469
6665 6c69 7865 6f74 7a63 5c7b 6562 6a61
7265 7340 6275 6573 7463 6f69 696e 746e
6170 7863 7d32 317b 7b7d 6f53 7275 6563
6970 6979 7365 7b7d 7d35 387b 7b7d 7d32
0a70 485c 7277 7469 6665 6c69 7865 616e
7d76 5c7b 6568 6a61 6f63 6a6d 6e61 2064
5c7b 6562 6a61 7265 7340 6275 6573 7463
6f69 656e 746e 7972 7b20 7a38 327b 7b7d
7d31 357b 7b7d 6f53 7275 6563 6a20 6c69
7365 7d7d 685c 6165 6364 6a6f 616d 646e
7b20 625c 6165 656d 4072 7573 7362 6385
6974 6e6f 6170 6567 2073 357b 7b7d 7d34
```



- ▶ IT student at BFH
 - ▶ Specialisation in Infosec
 - ▶ Been working with Blockchain Technology for 1.5 years
 - ▶ Bachelor thesis on the analysis of the Bitcoin Blockchain
- ▶ Founding member of the Chaostreff Bern



Systems for representing ownership

- State-transition systems
- Double-spend

Blockchain (without PoW)

- The underlying network
- Establishing a Consensus
- Double-spend on Blockchains

Blockchain

- PoW and mining
- Solving double-spend
- Network attack

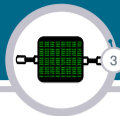
Bitcoin

- Blocks
- Light clients

Bonus Slides

- Branches in the chain

Content II



Double-spend
Mining
Pruning client

Content

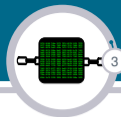
Double-spend
Mining
Pruning client

With this presentation, I want to tell you not only how Blockchains work, but also why they work the way they do and why the ideas and concepts are robust enough.



Section 1

Systems for representing ownership



Systems for representing ownership

State-transition systems

Double-spend

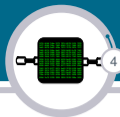
Blockchain (without PoW)

Blockchain

Bitcoin

Bonus Slides

Systems of ownership as state-transition-systems



Systems representing ownership can be modelled as state-transition system.

- ▶ Finance
- ▶ Estate
- ▶ ...

Mapping:

State Collection of who owns what

Transition Transferring ownership to someone else

Example for financial system:

State Collection of all accounts

Account Owner and associated amount

Transition Transaction (Moving value from one account to another)

└ Systems for representing ownership

└ State-transition systems

└ Systems of ownership as state-transition-systems

Systems representing ownership can be modelled as state-transition system.

- Finance
- Estate
- ...

Mapping:

State Collection of who owns what
Transition Transferring ownership to someone else

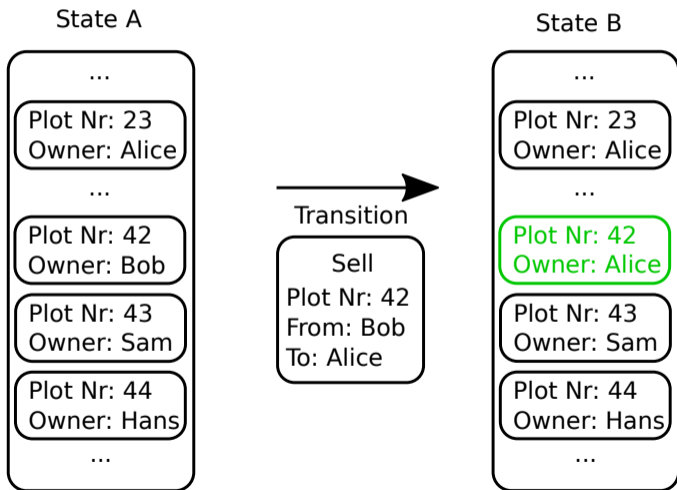
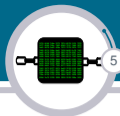
Example for financial system:

State Collection of all accounts
Account Owner and associated amount
Transition Transaction (Moving value from one account to another)

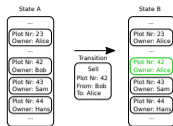
One of the core ideas of this talk is that ownership can be represented as a state-transition system.

If we represent ownership in a state-transition system, then a state is the information about who owns what at a given point in time. If someone transfers ownership to someone else (for example by selling an object), then that marks a transition which leads to a new state.

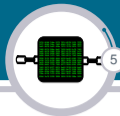
We can easily map financial systems to a state-transition-system. All accounts together represent the state. Every time someone makes a transaction, this marks a transition in the system which leads to a new state.



- Systems for representing ownership
 - State-transition systems
 - Showcase



Here we have an example of an estate system expressed as a state-transition system. A state is the collection of all Plots. Each plot has an owner assigned. The state describes by whom each plot is owned. If a plot is sold, such as in the example where Bob sells Plot number 42 to Alice, the system transitions into a new State.



Systems for representing ownership

State-transition systems

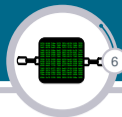
Double-spend

Blockchain (without PoW)

Blockchain

Bitcoin

Bonus Slides

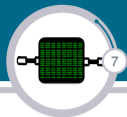


- ▶ Consensus is very important
- ▶ Before each transaction, all parties need to agree on the current state!
- ▶ The reason for this: double-spend

- └ Systems for representing ownership
 - └ Double-spend
 - └ Consensus

- Consensus is very important
- Before each transaction, all parties need to agree on the current state!
- The reason for this: double-spend

Consensus means that all parties of the system agree on the state of the system. I will now show how the system can be exploited if the parties can't agree on a state.



- ▶ To spend something twice
 - ▶ Spending the same money twice
 - ▶ Selling the same plot twice
 - ▶ ...
- ▶ Obviously malicious
- ▶ Well known attack in the Blockchain / Bitcoin world

└ Systems for representing ownership

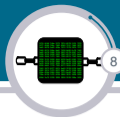
└ Double-spend

└ Double-spend

- To spend something twice
 - Spending the same money twice
 - Selling the same plot twice
 - ...
- Obviously malicious
- Well known attack in the Blockchain / Bitcoin world

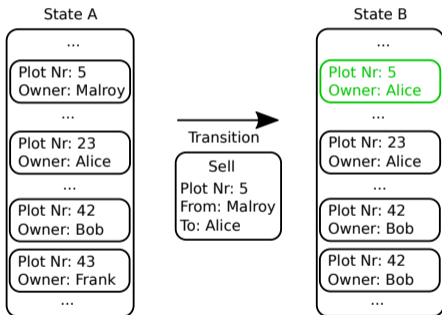
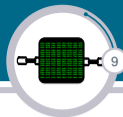
Double-spend is the action of spending the same thing twice. This is rather abstract for digital currencies such as the money we have on bank accounts. It's hard to imagine anyone spending the same money twice that way. It's rather simpler to imagine double-spend with estate: Imagine someone selling the same plot twice.

Double-spend estate example



- ▶ Malroy has a nice property (Plot number 5)
- ▶ Alice is looking to buy a new property
- ▶ Bob also wants to buy a new property
- ▶ Malroy will attempt to sell the same plot to both of them!

Example: Alice's view

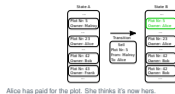


Alice has paid for the plot. She thinks it's now hers.

└ Systems for representing ownership

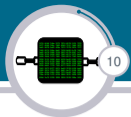
└ Double-spend

└ Example: Alice's view

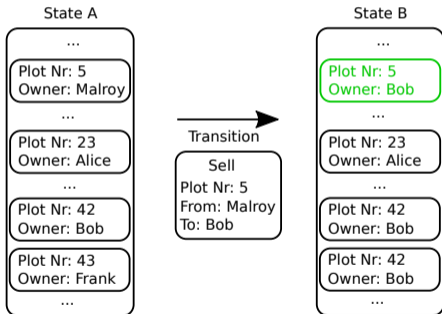


Alice pays Malroy for the plot and Malroy transfers ownership of the plot to Alice. This results in a new state where Alice is now the owner of plot 42. Both Alice and Bob know about this new state the system is now in.

Example: Bob's view



- ▶ Bob does not know about the transfer of ownership from Malroy to Alice



Bob has paid for the plot. He thinks it's now his.

└ Systems for representing ownership

└ Double-spend

└ Example: Bob's view

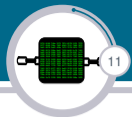
- Bob does not know about the transfer of ownership from Malroy to Alice



Bob has paid for the plot. He thinks it's now his.

Let's imagine Bob has never learned about the previous transfer of ownership which has led to "State B". Bob is still of the opinion that Malroy is the rightful owner of Plot number 5. So Bob will happily "buy" the plot from Malroy who will pretend to transfer ownership over the plot to Bob.

Example: problem



- ▶ Problem: Malroy sold the same plot twice!
- ▶ Alice and Bob do not agree on the current state of the system
- ▶ Their views on the state are incompatible
- ▶ This breaks the system:
 - ▶ Possibility to sell plots many times
 - ▶ People can't trade with each other

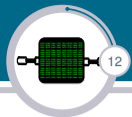
└ Systems for representing ownership

└ Double-spend

└ Example: problem

- Problem: Malroy sold the same plot twice!
- Alice and Bob do not agree on the current state of the system
- Their views on the state are incompatible
- This breaks the system:
 - Possibility to sell plots many times
 - People can't trade with each other

Bob having sold the plot twice is malicious of course. For him it means profit, but for Alice and Bob it's a big problem. Who owns the plot now? Who can use it and who can sell it to someone else. This is also a problem for other people who might want to buy the plot. Who can they buy it from? Having this situation clearly messes up the system. Some kind of conflict resolution would be needed. It would be even better though, if situations like this didn't occur in the first place.



Simple solution in the estate world:

- ▶ Registry of deeds (Grundbuchamt)
- ▶ Central authority
- ▶ Controls the state of the system
- ▶ Every time an estate is sold (a transition is made), it has to be done via the registry of deeds
- ▶ For each transition, the central authority performs certain checks to make sure the transition is compatible with the current state and either accepts or rejects it

That way, everybody can agree on a certain state and that state is always valid.

Systems for representing ownership

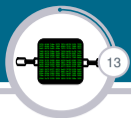
- Double-spend
- Solution

Simple solution in the estate world:

- Registry of deeds (Grundbuchamt)
 - Central authority
 - Controls the state of the system
 - Every time an estate is sold (a transition is made), it has to be done via the registry of deeds
 - For each transition, the central authority performs certain checks to make sure the transition is compatible with the current state and either accepts or rejects it
- That way, everybody can agree on a certain state and that state is always valid.

The double-spend problem has been solved in the estate world. The local registry of deeds watches over the state of the system. Each time a plot is sold, the registry of deeds has to be informed. If they notice an inconsistency, they can intervene.

Take the above example: Malroy wants to sell the plot to Alice. So they go to the registry of deeds together. The registry of deeds checks that the plot really belongs to Malroy. When the Malroy transfers ownership to Alice, the registry of deeds updates their records accordingly. Later on, when Malroy comes back with Bob to sell the plot to him, the registry of deeds will see in their records that Malroy does not own the plot any longer and will prevent him from selling it from Bob. The attack has been prevented.



Requirements:

- ▶ All parties need to agree on the current state
- ▶ All parties need to agree on whether a transition is valid

Banks as central authority:

- ▶ Banks serve as central authorities
- ▶ They control the state and check all transitions
- ▶ Always the case for modern day money transfers

This works surprisingly well.

- └ Systems for representing ownership
 - └ Double-spend
 - └ Solution for monetary systems

Requirements:

- All parties need to agree on the current state
- All parties need to agree on whether a transition is valid

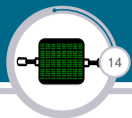
Banks as central authority:

- Banks serve as central authorities
- They control the state and check all transitions
- Always the case for modern day money transfers

This works surprisingly well.

The same solution that worked for the estate system, a central authority controlling the state, can also work for monetary systems.

Typically we see this when making monetary transactions via our banks. The bank acts as a central authority that manages the state and ensures the integrity of the system.



Requirements:

- ▶ Decentralized
- ▶ Needs to work on the internet

Central authority:

- ▶ Possibility of censorship
- ▶ Can be attacked
- ▶ Collects all the data

- └ Systems for representing ownership
 - └ Double-spend
 - └ A Solution for the internet

Requirements:

- Decentralized
- Needs to work on the internet

Central authority:

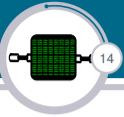
- Possibility of censorship
- Can be attacked
- Collects all the data

If we are looking for a system that allows the participants to agree on a state over the internet, it gets a bit more complicated.

Having a central authority manage the state would of course work. Think of Paypal: They are doing just that - allow anyone to make transactions via the internet in a secure manner. Paypal acts as the central authority managing the state and ensuring the integrity of the system.

However, a central authority can censor the system, can be attacked and hacked and it can of course collect all the data of its users. This then is not an adequate solution for a decentralized internet. Paypal should be prove enough that this is not the solution we are looking for.

A new system is required. A system appropriate for the Internet age.

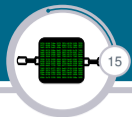


Section 2

Blockchain (without PoW)

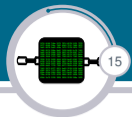
In this chapter I will introduce the basic ideas of Blockchain systems but without Proof of Work (PoW). I will show how they allow the participants to agree on a state. Without PoW though, the concepts explained here were insufficient to secure the Blockchain. This is demonstrated by showing a double-spend attack.

The next chapter will introduce PoW and show how it makes the Blockchain secure.



Blockchain is an internet-age solution to the same problem. It provides these (amazing) properties:

- ▶ No central authority
- ▶ Parties do not need to trust each other
- ▶ Parties need no information on who is participating
- ▶ ... not even any information on how many others are participating
- ▶ And still they can all agree on a state



Systems for representing ownership

Blockchain (without PoW)

The underlying network

Establishing a Consensus

Double-spend on Blockchains

Blockchain

Bitcoin

Bonus Slides

2016-12-28

- └ Blockchain (without PoW)
 - └ The underlying network
 - └ Subsection

Systems for representing ownership

Blockchain (without PoW)

The underlying network

Enabling it on Bitcoin

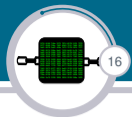
Double-spend on Blockchains

Blockchain

Bitcoin

Bonus Slides

Let's first talk of the network we use to distribute transactions among all participants.



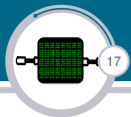
- ▶ We all agree on an initial state
 - ▶ Could be an empty state
 - ▶ Or something else
- ▶ We build a peer to peer network
- ▶ Transactions are published / announced to the p2p network
- ▶ Network relays transactions

- └ Blockchain (without PoW)
 - └ The underlying network
 - └ The general idea

- We all agree on an initial state
 - Could be an empty state
 - Or something else
- We build a peer to peer network
- Transactions are published / announced to the p2p network
- Network relays transactions

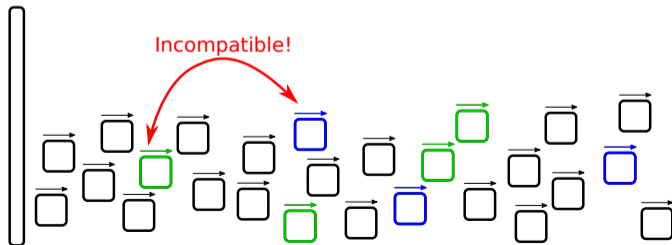
We build a Peer to Peer (p2p) network used to distribute the transactions amongst all participants. Everyone who makes a transaction announced that to the network. The network the relays the transaction until all participants have seen it.

Distributing transactions



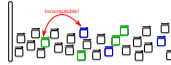
All sorts of problems:

- ▶ Network out of sync
- ▶ Order unclear
- ▶ Double-spend attempts
- ▶ Conflicting transactions
- ▶ Transactions dependant on conflicting transactions

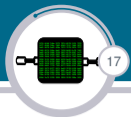


- Blockchain (without PoW)
 - The underlying network
 - Distributing transactions

- All sorts of problems:
- Network out of sync
 - Order unclear
 - Double-spend attempts
 - Conflicting transactions
 - Transactions dependant on conflicting transactions



The transactions on their own are a big mess. Because of delays in the network they don't get consistently spread to all participants, the order in which they arrive may be random, there might even be conflicting transaction. People will attempt double-spends and other attacks.



Systems for representing ownership

Blockchain (without PoW)

The underlying network

Establishing a Consensus

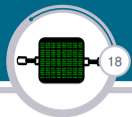
Double-spend on Blockchains

Blockchain

Bitcoin

Bonus Slides

Introducing: the Blockchain



A method is needed to establish a consensus, to agree on a state.
We call this method: Blockchain
(For now: Without Proof of Work, PoW)

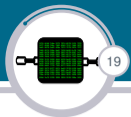
- └ Blockchain (without PoW)
 - └ Establishing a Consensus
 - └ Introducing: the Blockchain

A method is needed to establish a consensus, to agree on a state.
We call this method: Blockchain
(For now: Without Proof of Work, PoW)

We need a way to assemble the transactions which get distributed through our P2P network. We need a method for agreeing on a state which explicitly defines which transactions are valid.

The system to do that is the Blockchain.

(Again: The Blockchain explained in this chapter is a blockchain without Proof of Work (PoW). Because of that it is incomplete and insecure. The next chapter will introduce POW and explain how it secures the Blockchain.)



- ▶ Group transactions into blocks
- ▶ Blocks depend on one-another
- ▶ Anyone can form a new block at any time
- ▶ We define the current state as:
 - ▶ All transactions within the longest branch of blocks applied to the initial state
 - ▶ Transactions only become part of the state once they are inside a block
- ▶ Blocks have to meet certain criteria:
 - ▶ All transactions within the block must be valid
 - ▶ Transactions within the block have to be compatible with one another
 - ▶ Transactions within the block have to be compatible with transactions in earlier blocks

Blockchain (without PoW)

Establishing a Consensus

Blocks

- Group transactions into blocks
- Blocks depend on one-another
- Anyone can form a new block at any time
- We define the current state as:
 - All transactions within the longest branch of blocks applied to the initial state
 - Transactions only become part of the state once they are inside a block
- Blocks have to meet certain criteria:
 - All transactions within the block must be valid
 - Transactions within the block have to be compatible with one another
 - Transactions within the block have to be compatible with transactions in earlier blocks

Here is what we do to establish a consensus: We group the transactions into blocks. Anyone can collect a bunch of transactions and turn them into a new block at any time. Whenever a new block is formed, it is then announced by its creator to the network which will relay the block to all participants.

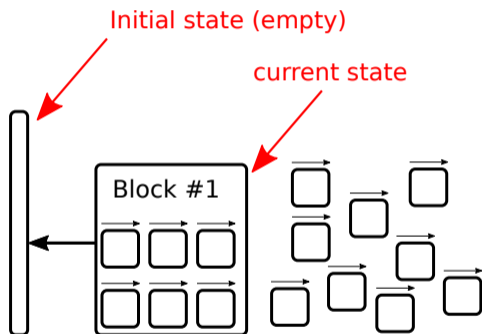
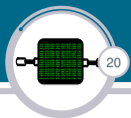
The blocks depend one on another. That is to say, every block has a reference to the block that came before it. That way, the blocks can be connected to a distinct chain of blocks.

All participants agree on a simple rule: The state is the longest chain of blocks. So to form the current state, one takes the initial state and then applies all transactions from block #1, then the transactions from block #2 and so on until all transactions from all blocks have been applied. The result is the current state of the system.

To make sure the resulting state is unambiguous, all blocks have to meet certain criteria.

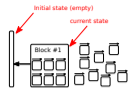
- They may only contain valid transactions.
 - Transactions which transfer a negative amount of money for example are not allowed
- All the transactions have to be compatible with one another. So if someone attempts to spend the same money twice, only one of the two transactions may be part of the same chain

Blocks which do not meet these criteria are not valid blocks and are to be ignored by the participants of the system.



- ▶ Group transactions into blocks
- ▶ Transactions only become part of the state once they are inside a block

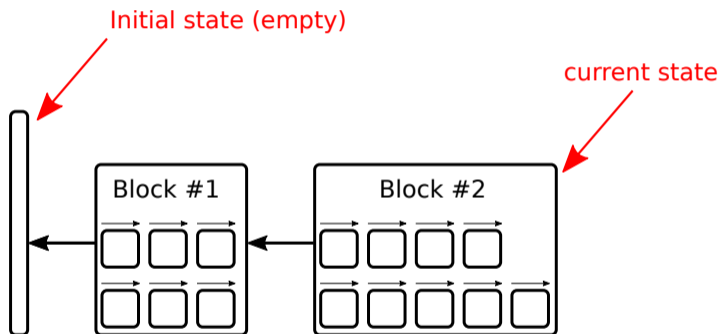
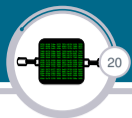
- Blockchain (without PoW)
 - Establishing a Consensus
 - Visualization of Blockchain concepts



- Group transactions into blocks
- Transactions only become part of the state once they are inside a block

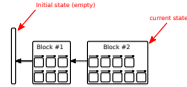
Transactions are grouped into Blocks. Only the transactions which included in a block are part of the state. Transactions that have arrived but not yet included in a block, should be ignored by the client software when calculating the current state.

Visualization of Blockchain concepts



- ▶ Blocks depend on one-another
- ▶ Anyone can form a new block at any time

- Blockchain (without PoW)
 - Establishing a Consensus
 - Visualization of Blockchain concepts

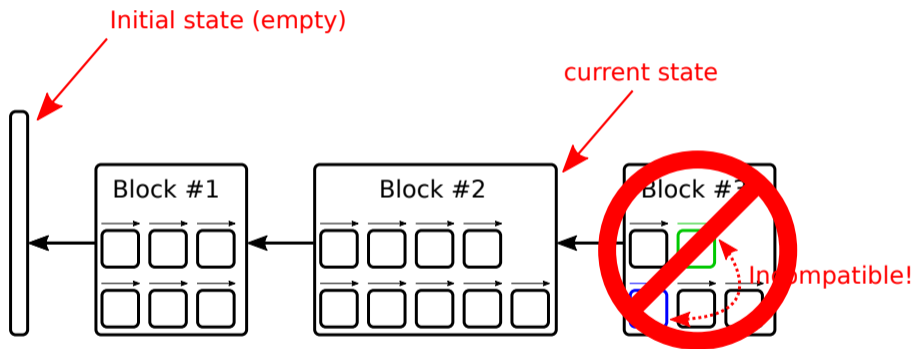
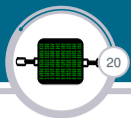


- Blocks depend on one-another
- Anyone can form a new block at any time

Each block has a reference to its previous block. That way the chain from the latest block to the initial state is unambiguous.

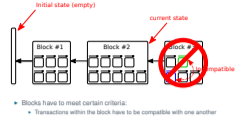
Blocks can be created by anyone at any time. This is important: We want a decentralized network where anyone can participate and no one is able to do censorship. This way a transaction that is ignored by one participant might be included in a new block by another.

Visualization of Blockchain concepts



- ▶ Blocks have to meet certain criteria:
 - ▶ Transactions within the block have to be compatible with one another

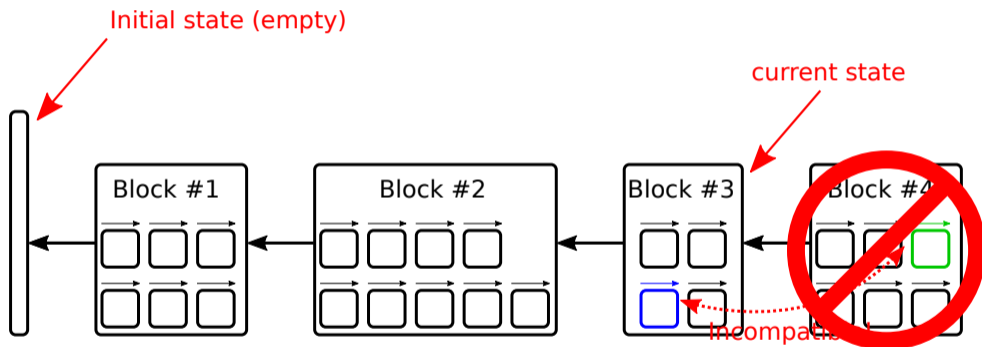
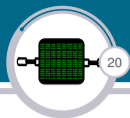
- Blockchain (without PoW)
 - Establishing a Consensus
 - Visualization of Blockchain concepts



Blocks which do not meet the criteria listed above should be ignored by the participants of the network when calculating the current state.

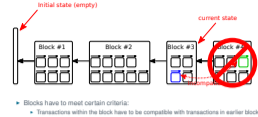
In this example, the "Block #3" contains two transactions which are not compatible with one another. We do not allow for such blocks and thus this block is dubbed invalid and not part of the state.

Visualization of Blockchain concepts



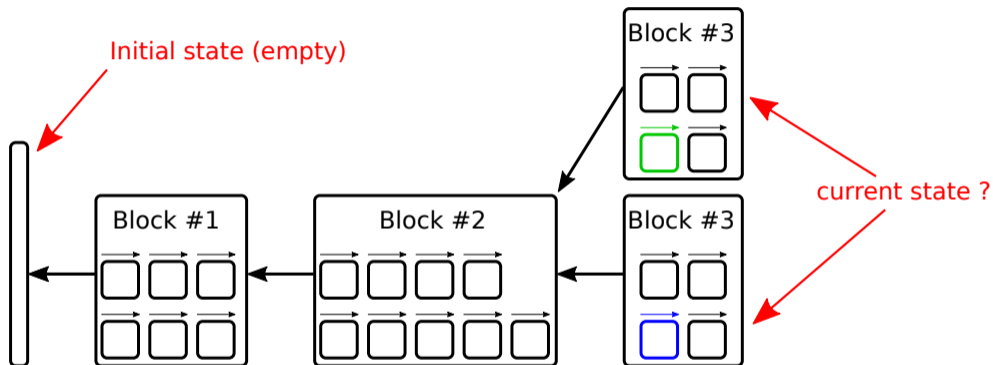
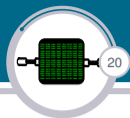
- ▶ Blocks have to meet certain criteria:
 - ▶ Transactions within the block have to be compatible with transactions in earlier blocks

- Blockchain (without PoW)
 - Establishing a Consensus
 - Visualization of Blockchain concepts



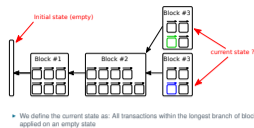
"Block #4" contains a transaction which is not compatible with another transaction which is part of "Block #3". We do not allow this since it would result in the state not being unambiguous. Since the block violates one of our rules, it is not a valid block and does not become part of the state.

Visualization of Blockchain concepts



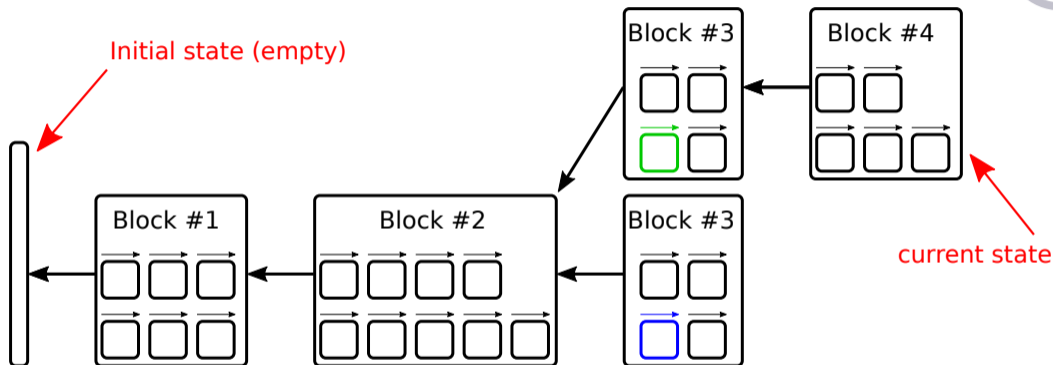
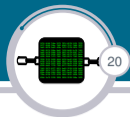
- We define the current state as: All transactions within the longest branch of blocks applied on an empty state

- Blockchain (without PoW)
 - Establishing a Consensus
 - Visualization of Blockchain concepts



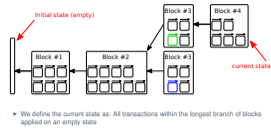
Both of the "Block #3" are valid. They also both form a chain of the same length. This means that the state of the network is no longer unambiguous. Every client might decide for themselves which of the two states they want to accept as the current state.

Visualization of Blockchain concepts



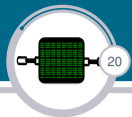
- We define the current state as: All transactions within the longest branch of blocks applied on an empty state

- Blockchain (without PoW)
 - Establishing a Consensus
 - Visualization of Blockchain concepts



Someone has created a new "Block #4". Because the block has a pointer / reference to the block before it, it is obvious which "Block #3" the new block depends on. One of the two chains has thus become longer than the other and is now accepted by the network as the current state.

Which previous block the new one should depend on is left to the person who creates the new block. It is their sole decision.



Systems for representing ownership

Blockchain (without PoW)

The underlying network

Establishing a Consensus

Double-spend on Blockchains

Blockchain

Bitcoin

Bonus Slides

- └ Blockchain (without PoW)
 - └ Double-spend on Blockchains
 - └ Subsection

Systems for representing ownership

Blockchain (without PoW)

The underlying network

Establishing a Consensus

Double-spend on Blockchains

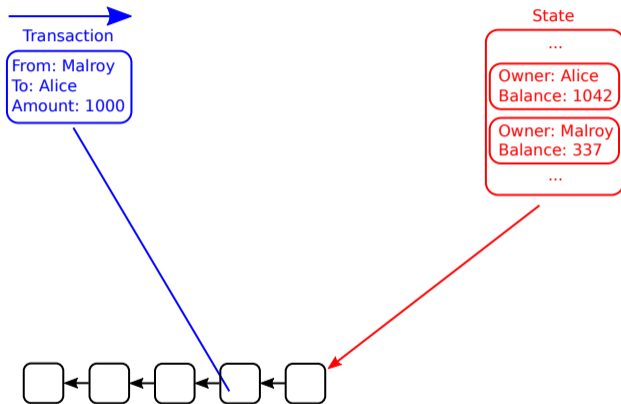
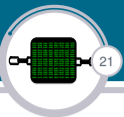
Blockchain

Blockchain

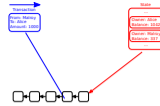
Bonus Slides

The double-spend attack covered in this chapter is the same one potentially used against real Blockchains. Here it can succeed because we don't use a Proof of Work (PoW). In the next chapter we will discuss how actual Blockchains use PoW to protect against that kind of attack.

Double-spend on a Blockchain without POW



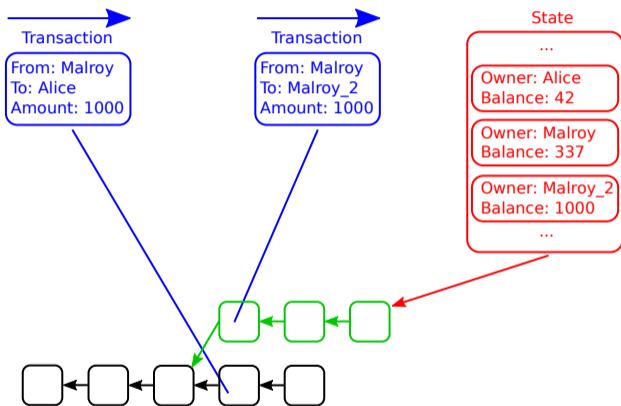
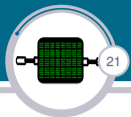
- Blockchain (without PoW)
 - Double-spend on Blockchains
 - Double-spend on a Blockchain without POW



Alice has a shop which sells bicycles. In this example, Malroy will attempt to steal a bicycle. That is to say, he will get Alice to give him the bicycle, but he will keep the money he was supposed to pay for it.

So he goes to Alice and chooses a bicycle. This certain bicycle has a price of 1000 Euro. He makes a transaction transferring the 1000 Euro to Alice. A short while later, someone will create a block including this transaction which results in the transaction becoming part of the state. This means that Alice has received the money. She now hands over the bicycle to Malroy who drives off with it. Further blocks are created over time.

Double-spend on a Blockchain without POW



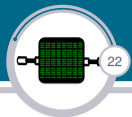
- └ Blockchain (without PoW)
 - └ Double-spend on Blockchains
 - └ Double-spend on a Blockchain without POW



Malroy who has gotten the bicycle and left to go somewhere safe, now starts the attack. He creates a new transaction which transfers the same money he previously sent to Alice to a second account of his instead. This transaction is of course incompatible with his first and won't become part of the chain if he just publishes it.

So Malroy creates a new, alternative block containing this second transaction. Then, he creates a bunch more blocks which all depend on the first one he created. He continues creating more blocks quickly until his chain is longer than the chain currently used on the network. Then he publishes the chain of blocks he has created to the network.

This chain created by Malroy is now the longest chain on the network and thus gets accepted as the state of the system. In this state, Malroy will own all the money, now distributed over two accounts. The transaction transferring the money to Alice is not part of the state anymore.

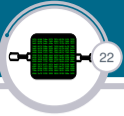


- ▶ Double-spend works
- ▶ It is easy and cheap to create blocks
 - ▶ Anyone can create any number of blocks at no cost

- └ Blockchain (without PoW)
 - └ Double-spend on Blockchains
 - └ Problems

- Double-spend works
- It is easy and cheap to create blocks
 - Anyone can create any number of blocks at no cost

As demonstrated above, the Blockchain described in this chapter is insecure. It is vulnerable to double-spend attacks. This is because the Blockchain described here is incomplete. Let's see how to fix that in the next chapter.



Section 3

Blockchain



Systems for representing ownership

Blockchain (without PoW)

Blockchain

- PoW and mining

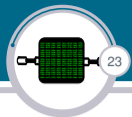
- Solving double-spend

- Network attack

Bitcoin

Bonus Slides

The problem



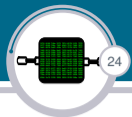
- ▶ Problem: Creating blocks is easy
- ▶ Solution: Make creating blocks hard

- Blockchain
 - PoW and mining
 - The problem

- Problem: Creating blocks is easy
- Solution: Make creating blocks hard

In the previous chapter, Malroy performed a double-spend attack. He did this by creating a large number of blocks quickly to form a branch of the Blockchain that was longer than any other.

The problem is, that it's easy and quick to create new blocks. The solution then is simple: Make creating new blocks hard and time consuming.



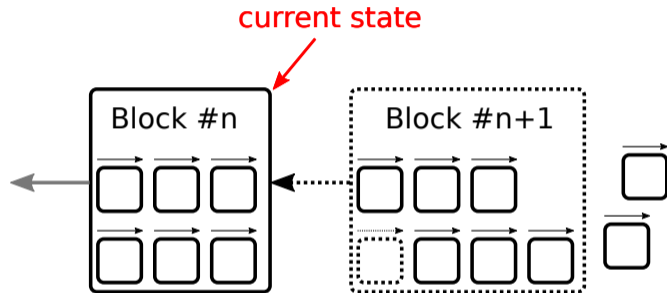
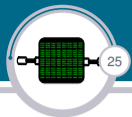
- ▶ Define a challenge taking a block as input
- ▶ For every new block created, the challenge has to be solved
 - ▶ Expensive in time and compute power
- ▶ The solution is called a Proof of Work (PoW)
 - ▶ The process of creating a PoW is called mining
- ▶ The PoW is to be published with the new block
- ▶ Only blocks with a valid PoW are valid blocks

- Blockchain
 - PoW and mining
 - PoW

- Define a challenge taking a block as input
- For every new block created, the challenge has to be solved
 - Expensive in time and compute power
- The solution is called a Proof of Work (PoW)
 - The process of creating a PoW is called mining
- The PoW is to be published with the new block
- Only blocks with a valid PoW are valid blocks

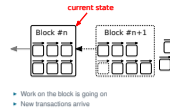
To make creating blocks hard, we define a challenge that has to be solved by anyone who wants to create a new block. The challenge is dependant on the block it is created for and is thus different for each block. The solution to the challenge has to be published together with the block and is called a "Proof of Work" or PoW for short, since it is proof that the person creating the block, called the miner, has performed the work necessary to find this solution. We change the rule for valid blocks so that blocks that do not have a PoW or which's PoW is not valid, are not valid blocks and are to be ignored by the clients on the network.

Creating PoW protected blocks



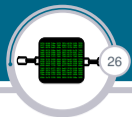
- ▶ Work on the block is going on
- ▶ New transactions arrive

- Blockchain
 - PoW and mining
 - Creating PoW protected blocks



The process of creating blocks changes a bit with the introduction of PoW. They can no longer be created just like that. Miners, which are the people creating new blocks, are constantly working on the next block. In the figure above, the block that is currently being worked on by a miner, is marked by the dotted line. The transactions that will be part of that block are not yet part of the state.

During the time a new block is worked on, new transactions arrive at the miner's machines. The miner stores them in what is called the transaction pools. Transactions in the transaction pool are potential candidates for inclusion in the next block. Each time a new valid block is announced on the network, the miners immediately start work on the next block. Which transactions a miner wants to include in the block they create is their own decision.



The function / challenge used for the PoW has to meet certain requirements:

- ▶ Be hard! Solvable only by brute force
- ▶ Validation needs to be fast and easy
- ▶ Dependant on the exact block it is produced for
 - ▶ To prevent pre-compute attacks
- ▶ Variable difficulty

Blockchain

PoW and mining

Requirements for PoW

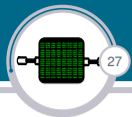
The function / challenge used for the PoW has to meet certain requirements:

- Be hard! Solvable only by brute force
- Validation needs to be fast and easy
- Dependant on the exact block it is produced for
 - To prevent pre-compute attacks
- Variable difficulty

A client receiving a block and supposedly matching PoW needs to be able to verify the PoW quickly so it can decide whether that block is actually part of the state. So even though creating a PoW needs to be hard, checking it needs to be simple.

The challenge being dependant on the block, that is to say the collection of transactions, it was created for, is very important. If it was not, Malroy could pre-compute a large number of PoWs over a long time and later attach them to some independant blocks to quickly create a long chain of blocks. If however, the PoW is fully dependant on the block it is created for, work on the PoW can only begin once the exact content of the block is known. So in the best case, work on the PoW for a new block, can only start when the block before that is known.

If the system is used for financial transactions, the time it takes for new blocks to appear needs to be stable. For those making transactions, knowing how long it takes for those transactions to become validated is important. But if the network's total compute power increases, the time it takes until a miner successfully creates a new block becomes lower. This is why the difficulty of the problem to be solved should be variable, so it can be changed over time as the network's compute power changes.



For mining to be worthwhile, a reward is needed:

- ▶ A transaction from nowhere to the miner
 - ▶ Transaction fees
 - ▶ Coinbase / new money
- ▶ Included in the block
- ▶ Makes blocks individual
 - ▶ Ensures distribution of success amongst all miners

Blockchain

- PoW and mining
- Reward

For mining to be worthwhile, a reward is needed:

- ▶ A transaction from nowhere to the miner
 - Transaction fees
 - Coinbase / new money
- ▶ Included in the block
- ▶ Makes blocks individual
 - Ensures distribution of success amongst all miners

Now that Mining is costly (taking time and using compute power), we need an incentive, otherwise no one is going to do it. The solution used in PoW based Blockchains is called a block-reward. It's a reward going to the person who creates the block.

The easy way to do this is to allow the miner to add a special transaction in the block they create. This transaction transfers money to the miner. It is called a Coinbase transaction. This money can come from various sources:

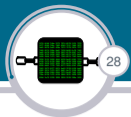
- The transaction fees of the transactions included in the block
- New money that is introduced into the system

The second option is interesting with the above example where we defined the initial state as an empty state where no one owned any money. This gives us a fair and independent means of bringing money into the initially empty system.

Of course, each miner will transfer the money in the Coinbase transaction to themselves. Because of that, each Coinbase transaction is different. That also means that the exact problem the miner is working on is individual to them, since it depends on the exact input which includes the Coinbase transaction. This ensures another important property:

Imagine if all miners were working on the same exact problem in the same manner. The first miner to find the solution would be the one with the most compute power. Always. That is bad since we want many people to create blocks to prevent censorship. By having all miners on slightly different problems, we give those who have less compute power a chance of finishing first by chance.

In practice, this ensures that the chance of finishing a block first, is roughly equal to the amount of compute power a miner has. That is to say, a miner with 20% of the network's power will generate about a fifth of all blocks.



- ▶ Which branch should a miner work on?
- ▶ For the reward to be spendable, it needs to be part of the state
- ▶ The state is the "longest" branch
- ▶ So it only makes sense to work on the "longest" branch
- ▶ Works without any coordination!

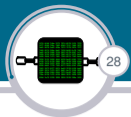
- Blockchain
 - PoW and mining
 - Mining

- Which branch should a miner work on?
- For the reward to be spendable, it needs to be part of the state
- The state is the "longest" branch
- So it only makes sense to work on the "longest" branch
- Works without any coordination!

We have seen in some of the examples above, that there might be multiple competing branches in a Blockchain. So each miner needs to decide which of those branches they want to work on.

The reward a miner gets, as mentioned above, is a special transaction. If the miner wants to spend the money they get from this transaction, then it needs to be part of the state. It is only part of the state, if the block it is included in is part of the state. For this reason, it's only interesting for miners to work on the longest branch. This makes sure, that all miners are working on the longest branch without any coordination being necessary.

This is important to ensure that the power of the miners gets combined to produce a single branch at the fastest speed possible. Malicious miners will have to compete against all of them at once when trying to create a separate branch.



Systems for representing ownership

Blockchain (without PoW)

Blockchain

- PoW and mining

- Solving double-spend

- Network attack

Bitcoin

Bonus Slides

2016-12-28

└ Blockchain
└ Solving double-spend
└ Subsection

Systems for representing ownership

Blockchain (without PoW)

Blockchain

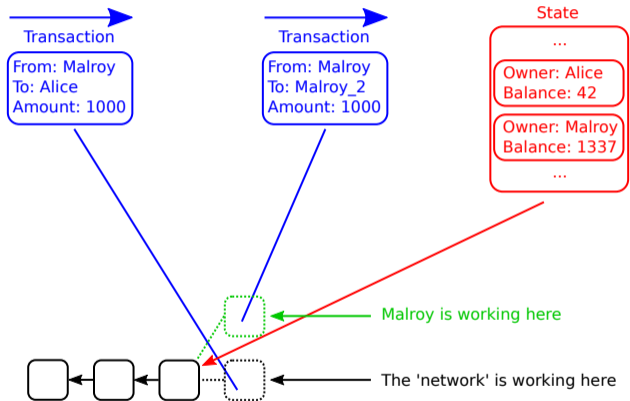
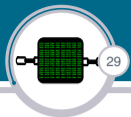
Proof and mining
Solving double-spend
Network attack

Blockchain

Bonus Slides

Let's revisit the double-spend attack from earlier. The attack remains the same, except that we now have PoW enabled on our Blockchain.

Double-spend



- ▶ Work on the next block starts
- ▶ Malroy is the only one working on a different block

Blockchain

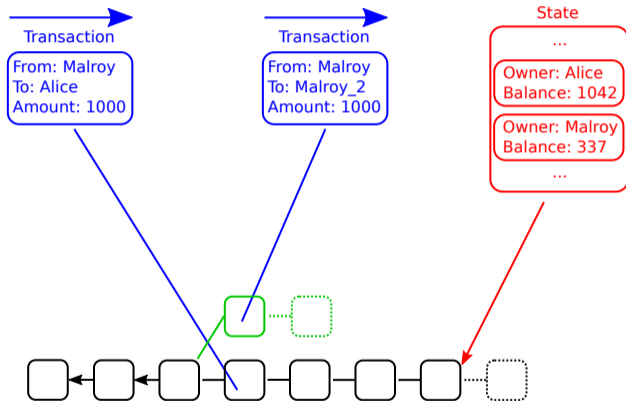
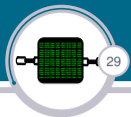
- Solving double-spend
- Double-spend



Note how blocks don't just suddenly appear anymore. Instead the miners are constantly working on new blocks. This is represented by the dotted blocks. The situation is largely the same as it was for the same attack in the last chapter. Malroy starts work on his branch as soon as the previous block has been announced to the network. However, this time, Malroy can't easily create a longer branch. He needs to mine each block. This is a lot of work. Since Malroy's mining power is less than the combined power of all the other miners in the network, Malroy creates new blocks at a slower rate. Thus, his branch never becomes the longest one and never becomes the state of the system.

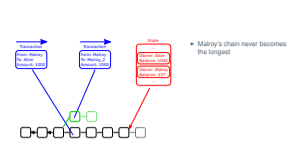
The attack has successfully been prevented!

Double-spend



- ▶ Malroy's chain never becomes the longest

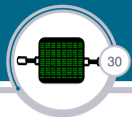
- Blockchain
 - Solving double-spend
 - Double-spend



Note how blocks don't just suddenly appear anymore. Instead the miners are constantly working on new blocks. This is represented by the dotted blocks. The situation is largely the same as it was for the same attack in the last chapter. Malroy starts work on his branch as soon as the previous block has been announced to the network. However, this time, Malroy can't easily create a longer branch. He needs to mine each block. This is a lot of work. Since Malroy's mining power is less than the combined power of all the other miners in the network, Malroy creates new blocks at a slower rate. Thus, his branch never becomes the longest one and never becomes the state of the system.

The attack has successfully been prevented!

50% attack

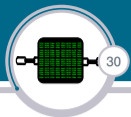


- ▶ Double-spend is possible
- ▶ >50% of network's power is needed
- ▶ With that, Malroy could produce new blocks faster than the rest of the network

- Blockchain
 - Solving double-spend
 - 50% attack

- Double-spend is possible
- >50% of network's power is needed
- With that, Malroy could produce new blocks faster than the rest of the network

The only way the attack could succeed is if Malroy had more mining power than all the other miners combined. That is to say, Malroy would need more than 50% of the network's mining power. Thus the term 50% attack. Malroy then would produce more blocks than all the other miners combined. This would allow him to create his own branches of the chain which would grow faster than any other branch and become the accepted state of the network eventually.



Systems for representing ownership

Blockchain (without PoW)

Blockchain

- PoW and mining

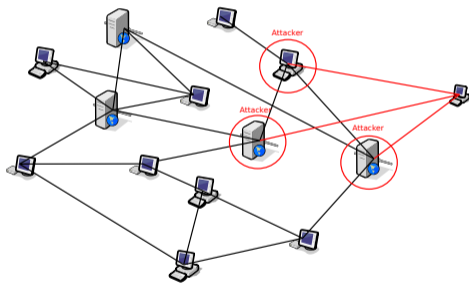
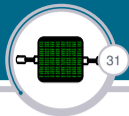
- Solving double-spend

- Network attack

Bitcoin

Bonus Slides

p2p networking attack setup



Most Blockchain currencies use a p2p network

- ▶ to distribute the transactions
- ▶ to distribute the blocks

- Blockchain
 - Network attack
 - p2p networking attack setup



Most Blockchain currencies use a p2p network

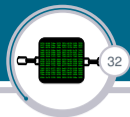
- to distribute the transactions
- to distribute the blocks

If Malroy controls Alice's access to the p2p network used to distribute transactions (either by controlling her network connection or by controlling the peers she connects to) he would have a lot of power. He could:

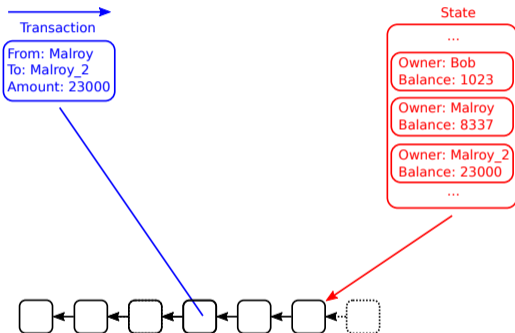
- Control Alice's view of the network
- Present to her blocks and transactions he does not present to the rest of the network
- Hide from her blocks and transactions the rest of the network sees

This can be used to perform a slightly more complicated and sophisticated form of double-spend attack.

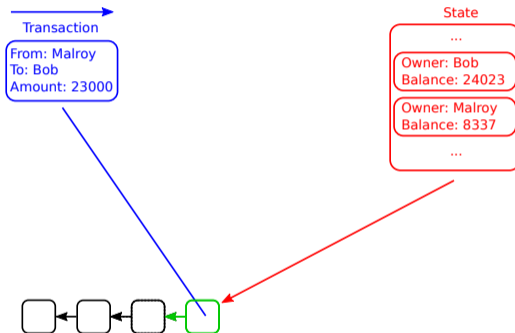
Network attack



Public network view:



Bob's view:

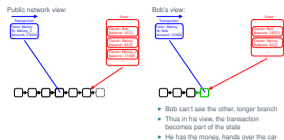


- ▶ Bob can't see the other, longer branch
- ▶ Thus in his view, the transaction becomes part of the state

Blockchain

Network attack

Network attack



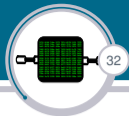
This attack is hard to perform, so Malroy will attempt to steal something more valuable this time. He will steal a car. For that, he goes to Bob who happens to be a car dealer. He will steal a 23000 Euro car.

Malroy controls all the nodes to which Bob is connected on the P2P network. When paying for the car, Malroy creates two different transactions which are not compatible with one-another. The first one is a transaction to Bob. Malroy sends this transaction to Bob only, the rest of the network does not see this transaction. The second transaction created by Malroy transfers the same money from his first to his second bank account. This transaction gets published to the network.

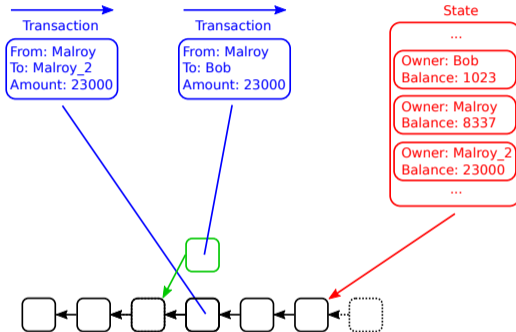
As soon as the previous block has been completed by the miners, work on the new block starts. The network will now create a block which contains the transaction from Malroy to his second account. Malroy alone is working on an alternative block which contains the transaction to Bob.

The network as a whole has more compute power than Malroy on his own and thus produces blocks at a faster pace. Malroy makes sure to block these new blocks so that Bob does not know about their existence. Eventually, Malroy succeeds in creating his block. He sends this block to Bob. Bob, who has not received any of the blocks recently created by the network, thinks that the block created by Malroy is in fact the longest branch. In Bob's opinion, the transaction transferring money from Malroy to Bob is now part of the state. He hands over the car to Malroy.

Network attack



- ▶ The transaction to Bob is not part of the state
- ▶ Bob does not have the money



- Blockchain
 - Network attack
 - Network attack

- The transaction to Bob is not part of the state
- Bob does not have the money



Malroy drives away. After a while he stops the attack. Bob gets unfiltered access to the network again. The two chains get merged. Bob's special chain is shorter than the actual network chain, which is why the transaction from Malroy to Bob is not part of the state.

So this kind of attack still works despite POW.

However, on the following slides, I will explain why such an attack is not worth the effort.

- Blockchain
 - Network attack
 - Network attack

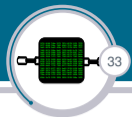
- The transaction to Bob is not part of the state
- Bob does not have the money



Short discussion: 10 minutes, the block time used by Bitcoin, is a long time. If you buy, say, an ice cream, you don't want to wait 10 minutes for your transaction to be processed. Because of this, some people will accept transactions as soon as they appear on the network - given of course, that the transaction is valid and not in conflict with any other transaction that is already part of the chain.

This will work fine in most cases, and if the ice cream seller gets tricked once or twice a week, he can probably still run his business. However, this practice is **NOT SECURE AT ALL**. Transactions should only be accepted as valid once they are included in a block.

This very network attack demonstrates why. If the car dealer were to accept transactions that are not yet in any block, all the attacker would have to do was to send a transaction to the car dealer. He would not have to put in the time to mine an extra block and thus the mechanisms described on the following slides would not work.



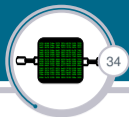
- ▶ The attack still worked
- ▶ It's now costly though
- ▶ Malroy's computer can either:
 - ▶ Perform the attack
 - ▶ Mine on the network
- ▶ For the time of the attack, Malroy has to decide
- ▶ This makes using the computer for the attack expensive

Just how expensive exactly?

- └ Blockchain
- └ Network attack
- └ What happened

- The attack still worked
 - It's now costly though
 - Malroy's computer can either:
 - Perform the attack
 - Mine on the network
 - For the time of the attack, Malroy has to decide
 - This makes using the computer for the attack expensive
- Just how expensive exactly?

The attack of course uses compute power and electricity, both of which have a cost. This cost however, is not what actually makes the attack expensive. Much more expensive than computers or the electricity used, is the time wasted. While the computer is trying to create a fake block for Bob, it can't simultaneously mine on the actual chain. The block rewards Malroy could earn but doesn't due to using the compute power for the attack is what makes the attack so expensive.



- ▶ Malroy's compute power: $20\% = \frac{1}{5}$ of the network
- ▶ Network average block time: 10min
- ▶ Block reward: 1000 Euro

Malroy needs to create one block:

Time Malroy needs to to create one block

$100\%Power \Leftrightarrow 10min \implies 20\%Power \Leftrightarrow 50min$

What if Malroy was mining for the same 50 minutes instead?

Reward in 50min mining

$100\%Power \Leftrightarrow 5Blocks \implies 20\%Power \Leftrightarrow 1Block \implies 1000Euro$

Blockchain

Network attack

Cost of the attack

- Malroy's compute power: $20\% = \frac{1}{5}$ of the network
- Network average block time: 10min
- Block reward: 1000 Euro

Malroy needs to create one block:

Time Malroy needs to create one block

100%Power \leftrightarrow 10min \implies 20%Power \leftrightarrow 50min

What if Malroy was mining for the same 50 minutes instead?

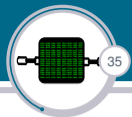
Reward in 50min mining

100%Power \leftrightarrow 5Blocks \implies 20%Power \leftrightarrow 1Block \implies 1000Euro

The values in the slide above are example values we are going to use for the calculations. If the produces 1 Block every 10 minutes, then it will take Malroy on his own 50 minutes to create a single block. That means that for 50 minutes, he can't mine on the actual chain.

During 50 minutes, the network produces 5 blocks, one of which, on average, would be produced by Malroy. So by performing the attack, he misses one block reward, worth 1000 Euro.

This does not depend on how much power the attacker has - an attacker with 30% of the network compute power takes less time to perform the attack, but he loses more block rewards per time.



Malroy can chose:

- ▶ Do the attack
 - ▶ Steal a 23000 Euro car
- ▶ Not do the attack
 - ▶ Earn 1000 euro mining

Attacker's power

The power of the attacker does not matter. An attacker with more power needs less time, but he looses more money per time.

- Blockchain
 - Network attack
 - Protection from the attack

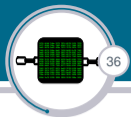
Melroy can choose:

- Do the attack
 - Steal a 23000 Euro car
- Not do the attack
 - Earn 1000 euro mining

Attacker's power

The power of the attacker does not matter. An attacker with more power needs less time, but he loses more money per time.

In our example, performing the attack makes sense, since the car, at 23000 Euro is worth a lot more than the single block reward. So Bob needs to take additional measures to be secure.



By waiting another block, Bob makes the attack twice as expensive.
So the larger a transaction we protect, the longer we have to wait. You can easily calculate for how long you have to wait to be secure:

How long to wait?

$$\lceil amount \div Blockreward \rceil + 1$$

- Blockchain
 - Network attack
 - Waiting

By waiting another block, Bob makes the attack twice as expensive. So the larger a transaction we protect, the longer we have to wait. You can easily calculate for how long you have to wait to be secure:

How long to wait?

$(\text{amount} + \text{Blockreward}) / 1$

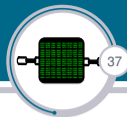
To protect himself from an attacker, Bob doesn't hand over the car immediately. Instead, once the transaction has been included in a block, and thus become part of the state, Bob waits until some more blocks have been added to the same branch.

If Bob was under attack, the attacker would have to create those additional blocks as well. It would cost him even more time to do so. All this time the attacker spends on creating the additional blocks he can't spend mining. Thus it costs him money. The longer we wait, the more expensive an attack becomes.

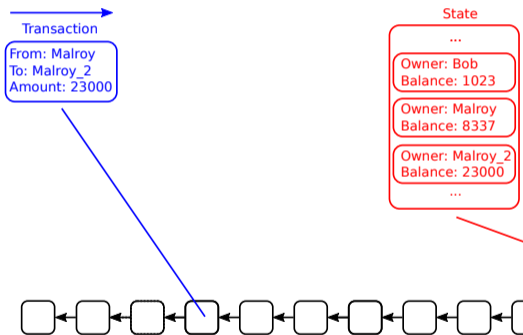
Bob can easily calculate how long exactly he has to wait in order to be secure, based on how expensive the product is he is selling and how high the block reward is.

By doing so, Bob makes sure attacking him is never worthwhile. Noone will have an incentive to attack him since doing so makes no sense at all economically speaking.

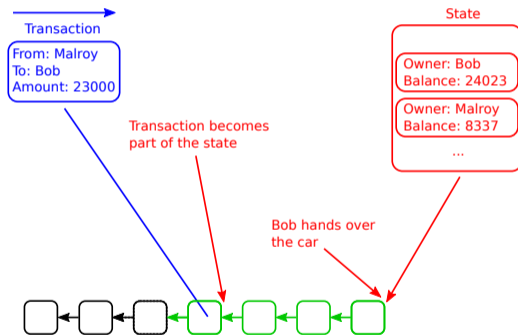
Bob's protection



Public network view:

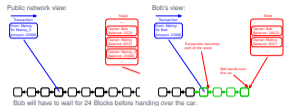


Bob's view:

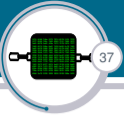


Bob will have to wait for 24 Blocks before handing over the car.

- Blockchain
- Network attack
- Bob's protection



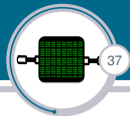
At 1000 Euro per block, Bob should wait 23, probably more like 24 blocks before handing the car to Malroy (or any other customer for that). The graphic above is not accurate, since I can't fit that many blocks onto a single slide ;)



Section 4

Bitcoin

I talked a lot about generic Blockchain designs and concepts. To get a better idea of how these concepts are implemented, we will now take a look at the Bitcoin Blockchain.



Systems for representing ownership

Blockchain (without PoW)

Blockchain

Bitcoin

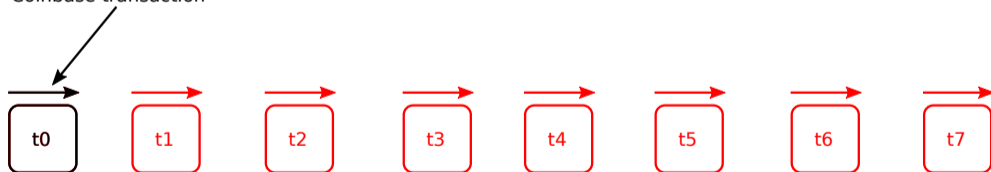
Blocks

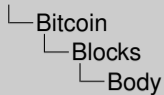
Light clients

Bonus Slides

- ▶ Contains transactions
- ▶ Ordered
 - ▶ Dependant transactions are in the correct order
- ▶ The first transactions is Coinbase

Coinbase transaction





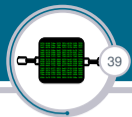
- Contains transactions
- Ordered
 - Dependent transactions are in the correct order
- The first transaction is Coinbase



In IT, we like to separate metadata from data by putting the data in what we call a body and the metadata into a header. We do this for messages, network protocols and file formats. We also do it for Blockchains. In Bitcoin, a block's body is comprised of the blocks it contains. The header contains some metadata.

The transactions inside the block's body are ordered. They get ordered by the miner who creates the block. The miner has to follow few rules for that. If Message "b" depending on message "a" is in the same block as "a", then they have to be in the block in the correct order. Other than that, the miners is basically free to order the transactions whichever way he likes. But once the order has been decided on, it can't be changed anymore.

The leftmost transaction (the first transaction inside the block), is the Coinbase transaction.



Bitcoin uses a Merkle tree to secure the transactions. Root of the Merkle tree is part of the Block header.

- ▶ Binary tree
- ▶ A node is the hash of the two child nodes
- ▶ Bitcoin uses sha256 double hashes
 - ▶ aka. dhash, double-sha256
 - ▶ $sha256 sha256(\dots)$

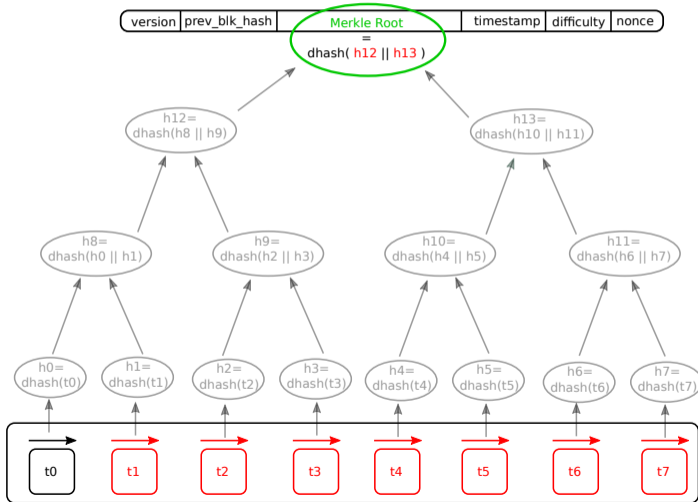
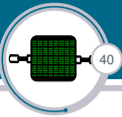


Bitcoin uses a Merkle tree to secure the transactions. Root of the Merkle tree is part of the Block header.

- Binary tree
- A node is the hash of the two child nodes
- Bitcoin uses sha256 double hashes
 - aka. `dhash, double-sha256`
 - `sha256(sha256(...))`

The Merkle tree, as used by Bitcoin, is a binary tree. Every node, except the leaf nodes, is formed by creating the doublehash of the concatenation of the two child nodes.

Merkle tree visualization



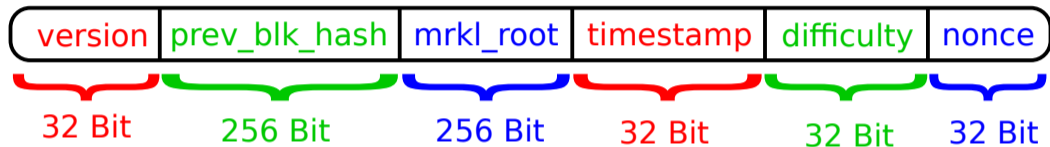
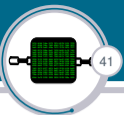
- Bitcoin
 - Blocks
 - Merkle tree visualization



The Merkle Root is put into the Block header. The collection of transactions become the block body. The rest of the Merkle tree is then no longer needed. If someone validates the block, they retrieve both the header and the body, then create the Merkle tree from the body they retrieved and compare the Merkle root to the value from the block header. If the two values match, everything is in order

It is important that the order of blocks in the body is preserved, otherwise the result of forming the Merkle tree would be a different one.

All transactions are secure. If one would be changed or replaced, the hash of that transaction would change, which would be reflected in the node above, which in turn would change that node's parent and so on until finally, the Merkle root would be a different one.



version Block version. Currently at 4

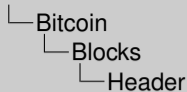
prev_blk_hash Hash of the previous block header. Reference to the previous block.

mrkl_root Reference to the transactions. Root node of the Merkle tree.

timestamp Standard UNIX timestamp. Complicated rules here.

difficulty Difficulty of the block. Network difficulty is recalculated every 2016 blocks.

nonce Nonce used to manipulate the block hash. Note: too short.



version Block version. Currently at 4
 prev_blk_hash Hash of the previous block header. Reference to the previous block.
 mrkl_root Reference to the transactions. Root node of the Merkle tree.
 timestamp Standard UNIX timestamp. Complicated rules here.
 difficulty Difficulty of the block. Network difficulty is recalculated every 2016 blocks.
 nonce Nonce used to manipulate the block hash. Note: too short.

The image above shows the header of a Bitcoin Block.

The prev_blk_hash stands for Previous Block Hash and is just that - the hash of the previous block. In our generic Blockchain construct from the chapters before, we have seen that each block references it's previous block in order to form a chain - this is Bitcoin's way of doing that.

Rules for block Timestamp:

"A timestamp is accepted as valid if it is greater than the median timestamp of previous 11 blocks, and less than the network-adjusted time + 2 hours.

'Network-adjusted time' is the median of the timestamps returned by all nodes connected to you." - The Bitcoin Wiki: https://en.bitcoin.it/w/index.php?title=Block_timestamp&oldid=51392

The difficulty is an expression of how hard it was to create the PoW for this block.

The nonce is a field used for mining, see below.

2016 Blocks equals 14 days.



PoW:

- ▶ Bitcoin uses the Block Hash for PoW
 - ▶ *dhash(block_header)*
- ▶ PoW: First n Bits of the Block Hash need to be 0
- ▶ Mining by incrementing the `nonce` field
 - ▶ Too short (32bit nonce for 256bit Hash)
- ▶ Secondary nonce
 - ▶ tx input / input script of Coinbase transaction
 - ▶ Change in transaction changes Merkle root

Difficulty:

- ▶ Difficulty is adjusted every 2016 Blocks (14 days)
- ▶ Network speed target: one Block ever 10 minutes

Bitcoin

Blocks

Bitcoin mining

PoW:

- Bitcoin uses the Block Hash for PoW
 - `sha256(block_header)`
- PoW: First n Bits of the Block Hash need to be 0
- Mining by incrementing the `nonce` field
 - Too short (32bit nonce for 256bit Hash)
- Secondary nonce
 - tx input / input script of Coinbase transaction
 - Change in transaction changes Merkle root

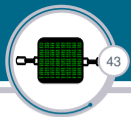
Difficulty:

- Difficulty is adjusted every 2016 Blocks (14 days)
- Network speed target: one Block over 10 minutes

Here is how mining in Bitcoin works: The Hash of the block header is used as a PoW. The difficulty in creating it is that the first n bits of the header need to be valud 0. A miner can only achieve this by modifying the header, hashing it and testing if the hash meets the criteria. If it does not, the miner has to modify the header again and repeat the process. The way the miner modifies the header, is by incrementing the `nonce` field.

The nonce field is only 32 Bit though and the Hash is 256 bit, so the nonce field might not always be enough to find a matching Hash, depending on the difficulty target. So the miner modifies another value under their control - the Coinbase transaction. More precisely, the field used to reference where the money is coming from is not used in the Coinbase transaction and can be filled with arbitrary data.

Modifying the Coinbase transaction changes its hash which eventually changes the Merkle root and thus the header.



The miner of a new block gets a reward. The reward is the sum of:

- ▶ Fixed reward/ newly generated money
 - ▶ Started at 50 Btc per Block
 - ▶ Halves every 210000 Blocks (ca. 4 years)
- ▶ The transaction fees of all transactions in the block

To do this, the miner creates an additional transaction:

- ▶ So called Coinbase transaction
- ▶ First (left most) transaction in the Block

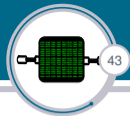
└ Bitcoin
└ Blocks
└ Coinbase: Creating money

The miner of a new block gets a reward. The reward is the sum of:

- Fixed reward/ newly generated money
 - Started at 50 btc per block
 - Halves every 210000 blocks (ca. 4 years)
 - The transaction fees of all transactions in the block
- To do this, the miner creates an additional transaction:
- So called Coinbase transaction
 - First (left most) transaction in the Block

Adjusting the difficulty target works like this:

Every Bitcoin client compares the actual time it took to generate these blocks with the two week goal and modifies the target by the percentage difference. Each node calculates the network difficulty independently from all other nodes. It is then, important that they all get the same result and thus that they all use the same algorithm.



Systems for representing ownership

Blockchain (without PoW)

Blockchain

Bitcoin

Blocks

Light clients

Bonus Slides

Types of Bitcoin clients



- ▶ Full node
 - ▶ Stores the entire Blockchain
 - ▶ Seeds the Blockchain to the network
 - ▶ Validates every block
 - ▶ Validates every transaction
- ▶ Pruning client
 - ▶ Validates all blocks
 - ▶ Validates all transactions
 - ▶ Only stores parts of the Blockchain
 - ▶ See the bonus slides
- ▶ Light client / SPV client
 - ▶ See below

Bitcoin

- Light clients
- Types of Bitcoin clients

- Full node
 - Stores the entire Blockchain
 - Seeds the Blockchain to the network
 - Validates every block
 - Validates every transaction
- Pruning client
 - Validates all blocks
 - Validates all transactions
 - Only stores parts of the Blockchain
 - See the bonus slides
- Light client / SPV client
 - See below

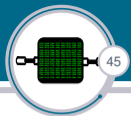
To my knowledge, there are three types of Bitcoin clients:

Full clients store the entire blockchain and seed the blocks into the network. They validate each transaction and each block.

Pruning clients also check all transactions and all blocks but they don't store the entire blockchain. Instead, they get rid of blocks they no longer need to save storage.

Light clients finally only validate single transactions. They use little bandwidth and storage and are suitable for mobile clients.

Chain of Block headers



- ▶ Download just the headers
- ▶ PoW can be checked from just the header
- ▶ Previous block can be checked from just the header
- ▶ Creating a header is as hard as creating a block!

Advantage:

- ▶ 450000 blocks
- ▶ Blockchain: >95 GB (>100GB with indexing)
- ▶ Block headers: <100 MB

- └ Bitcoin
 - └ Light clients
 - └ Chain of Block headers

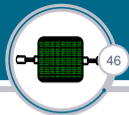
- Download just the headers
- PoW can be checked from just the header
- Previous block can be checked from just the header
- Creating a header is as hard as creating a block!

Advantage:

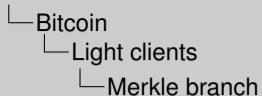
- 450000 blocks
- Blockchain: >95 GB (>100GB with indexing)
- Block headers: <100 MB

The first thing a light Client needs to validate a transaction is the list of all block headers. Each block header references the previous' block header by it's hash, thus the correct order of blocks can be determined from the headers alone. The PoW too can be checked from the header alone, since it's the header's hash. Creating a header is as hard as creating an entire block, so re-creating the chain like this is as secure as donloading the entire Blockchain, but much faster.

At the time of writing, there are almost 450000 blocks in the Blockchain which is in total about 95 GB in size. A list of all Block headers can be stored in well under 100 MB.



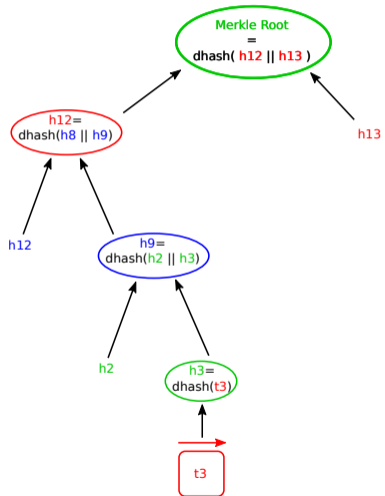
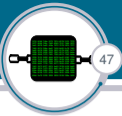
- ▶ We have the Merkle root
 - ▶ We can get it securely from the Block header
- ▶ We do not have the elements it consists of
- ▶ We are interested in a single element
 - ▶ A single transaction



- We have the Merkle root
 - We can get it securely from the Block header
- We do not have the elements it consists of
- We are interested in a single element
 - A single transaction

The Merkle tree allows us to secure all the transactions. Even more interesting though is another feature - the Merkle branch. The Merkle branch allows for a single branch to be extracted and then stored in a space efficient yet safe way. Despite that, it still proves that a transaction is part of a block.

Merkle branch visualization



- └ Bitcoin
 - └ Light clients
 - └ Merkle branch visualization



For the Merkle branch, instead of downloading all transactions, we only need the one transaction and then one hash from each level of the tree.

In the above example, we can prove that transaction t_3 is part of the block by downloading the transaction itself along with the hashes h_2 , h_{12} and h_{13} .

From these four values, we can form the Merkle branch and get the Merkle root.



Download a Merkle branch instead of all transactions:

- ▶ Less elements
- ▶ Smaller (hashes instead of transactions)
- ▶ The advantages get larger the bigger the tree

Example: Block with 1600 transactions:

- ▶ 1600 transactions
- ▶ 1 transaction + 11 hashes

- └ Bitcoin
 - └ Light clients
 - └ Merkle branch advantages

Download a Merkle branch instead of all transactions:

- Less elements
- Smaller (hashes instead of transactions)
- The advantages get larger the bigger the tree

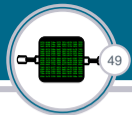
Example: Block with 1600 transactions:

- 1600 transactions
- 1 transaction + 11 hashes

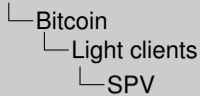
The Merkle branch will, especially for very large blocks, be smaller than the entire body.

Only the specific transaction plus \log_2 (number of transactions in the block) hashes need to be downloaded to form and verify the Merkle branch for a transaction.

Downloading and verifying the Merkle branch is just as secure as downloading the entire Block body.



- ▶ Simple Payment Verification
 - ▶ Proof that a transaction is part of the chain
 - ▶ Avoid downloading the entire chain
1. Retrieve all block headers
 2. Rebuild longest branch
 3. Retrieve a transaction
 4. Retrieve the Merkle branch for the transaction
 5. Match the Merkle branch to the chain of block headers



- Simple Payment Verification
 - Proof that a transaction is part of the chain
 - Avoid downloading the entire chain
1. Retrieve all block headers
 2. Rebuild longest branch
 3. Retrieve a transaction
 4. Retrieve the Merkle branch for the transaction
 5. Match the Merkle branch to the chain of block headers

Now we have everything needed to create a light client. This specific type of client is called an SPV client. SPV stands for Simple Payment Verification and was described in Satoshi's Bitcoin Whitepaper.

An SPV client downloads the block headers and forms the Blockchain from them. Then it receives a specific transaction. It requests the Merkle branch for this specific transaction. The Merkle branch connects the transaction to the block headers and thus proves that this transaction is actually part of the Blockchain.

This way, an SPV client can securely receive transactions.

Questions?

Ask them now or contact me:

33c3 DECT / GSM

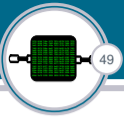
- ▶ 5346 (LEGO)
- ▶ 7869 (PUNX)

email

- ▶ niklaus@mykolab.ch

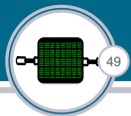
XMPP

- ▶ vimja@xmpp.honet.ch



Section 5

Bonus Slides



Systems for representing ownership

Blockchain (without PoW)

Blockchain

Bitcoin

Bonus Slides

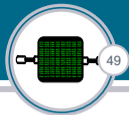
Branches in the chain

Double-spend

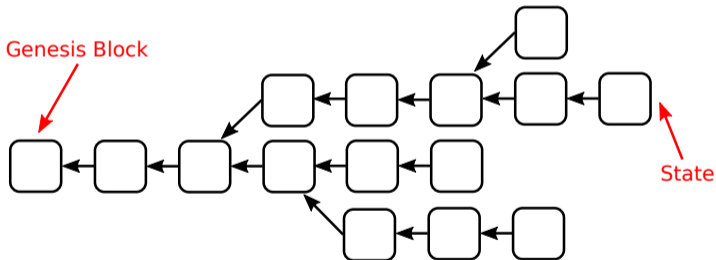
Mining

Pruning client

Status switch between branches

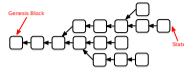


The longest chain of blocks represents the current state. If we display the chain as a tree, then that is the longest branch. The root of that tree is called the Genesis Block.



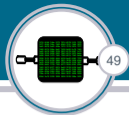
- Bonus Slides
 - Branches in the chain
 - Status switch between branches

The longest chain of blocks represents the current state. If we display the chain as a tree, then that is the longest branch. The root of that tree is called the Genesis Block.



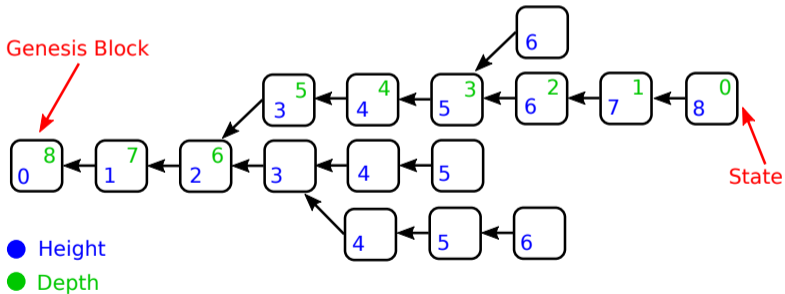
If suddenly, another branch were to become longer, then that branch would then be the state of the system.

Block height and depth



Height identifier

Depth expression of security



- Bonus Slides
 - Branches in the chain
 - Block height and depth

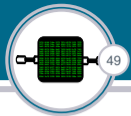


Even though we don't use numbers of blocks anymore to determine their dependencies, numbering them can still be useful. It allows for easy referencing of specific blocks. There are two important ways of doing this:

Height The block height expresses the block's distance from the Genesis Block. The height of a block is always the same and does not change when new blocks are added to the system. Every block in the system has a height.

Depth The depth of a block expresses the block's distance from the end of the branch. When a new block is appended to the branch, the depth of every block in the branch is increased by one. The depth is usually only expressed for blocks of the longest branch.

Later on when we introduce POW, the depth will become an important expression of security.



Systems for representing ownership

Blockchain (without PoW)

Blockchain

Bitcoin

Bonus Slides

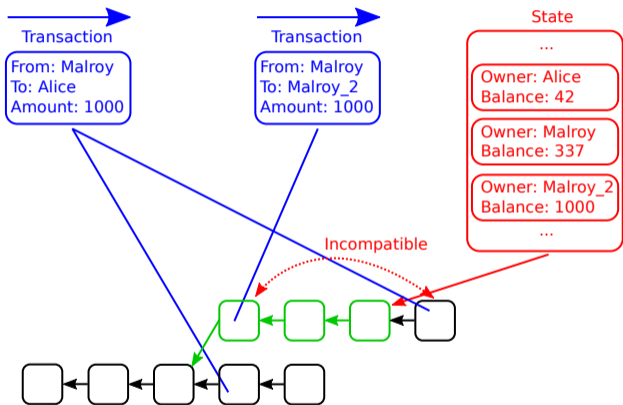
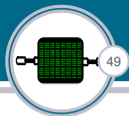
Branches in the chain

Double-spend

Mining

Pruning client

Conflicting transaction in double-spend attacks



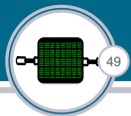
- ▶ New blocks containing the old transaction would conflict
- ▶ They would be invalid
- ▶ And they could not become part of the state

- Bonus Slides
 - Double-spend
 - Conflicting transaction in double-spend attacks



- New blocks containing the old transaction would conflict
- They would be invalid
- And they could not become part of the state

By creating a new transaction (the one transferring the money onto the vimja2 account) that is conflicting with the earlier transaction (the one transferring the money to the car dealer) and putting the new one into my chain/branch, I prevent the first transaction from ever becoming part of the same branch again. A block containing one of them will be in conflict with block containing the other, thus they can't become part of the same branch.



Systems for representing ownership

Blockchain (without PoW)

Blockchain

Bitcoin

Bonus Slides

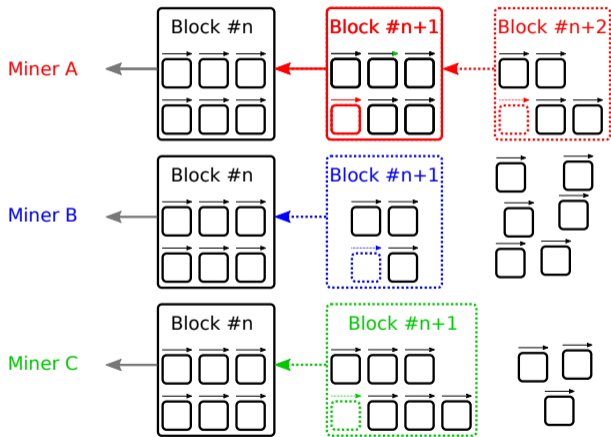
Branches in the chain

Double-spend

Mining

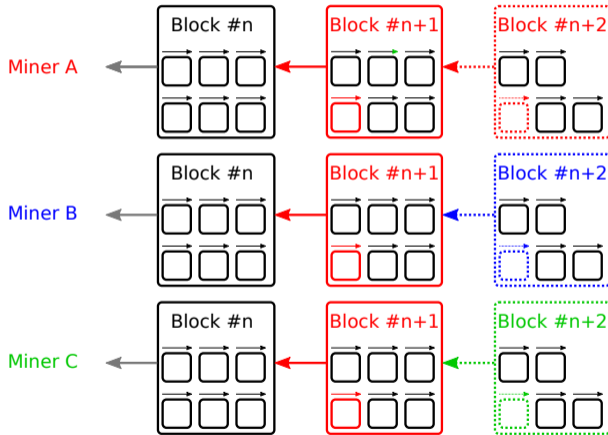
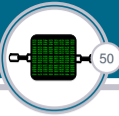
Pruning client

Example scenario with multiple miners



- ▶ Miner A finishes Block #n+1
- ▶ Miner A immediately starts work on #n+2
- ▶ Miner A broadcasts Block #n+1
- ▶ Miners B and C both receive the completed Block #n+1 from miner A
- ▶ They change their transaction pools accordingly

Example scenario with multiple miners



- ▶ Miners B and C immediately start work on Block #n+2



Systems for representing ownership

Blockchain (without PoW)

Blockchain

Bitcoin

Bonus Slides

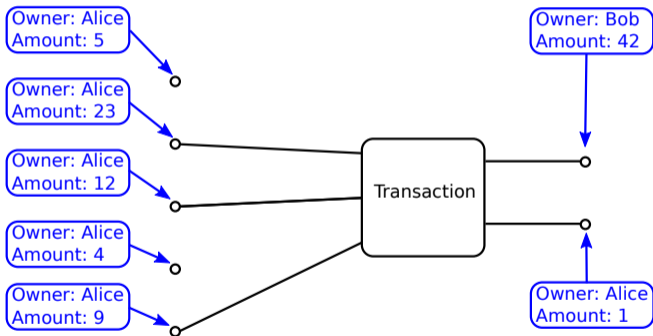
Branches in the chain

Double-spend

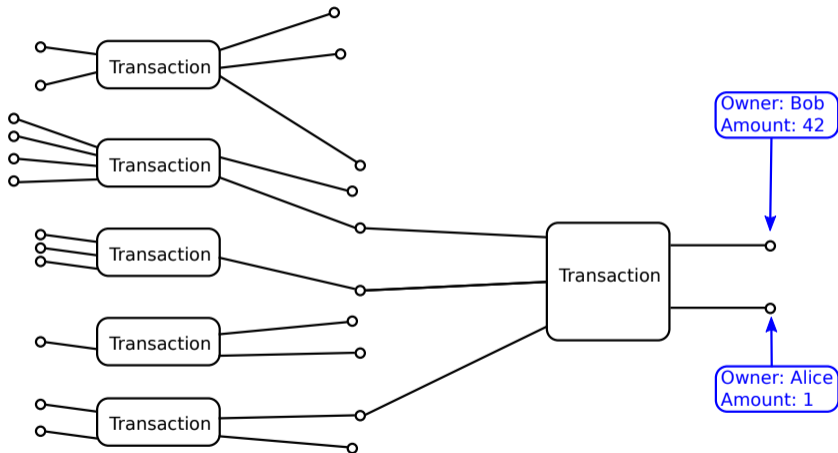
Mining

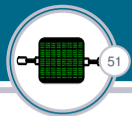
Pruning client

Bitcoin transaction



Bitcoin transaction





- ▶ This presentation is licensed under a Creative Commons Attribution 4.0 International License <http://creativecommons.org/licenses/by/4.0/>
- ▶ In the creation of this presentation, I used the Feather Beamer Theme by "Lilyana Vankova" which is released under the GPLv3 license.
 - ▶ As required by the GPLv3, I make the exact sources of the theme, as used by me, including all modifications I made, available to you. You can download them from https://gitlab.honet.ch/vimja/beamer_template (commit 1776ad90).