Introduction to Blockchains

33c3

Niklaus 'vimja' Hofer
niklaus@mykolab.ch

December 28, 2016

/me

- 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

Content I

Content II

- Double-spend
- Mining
- Pruning client

Section 1

Systems for representing ownership

Subsection

Systems of ownership as state-transition-systems

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

- Finance
- Estate
- ...

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

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

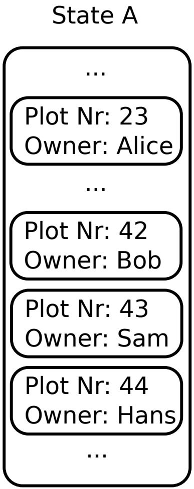
Example for financial system:

State Collection of all accounts

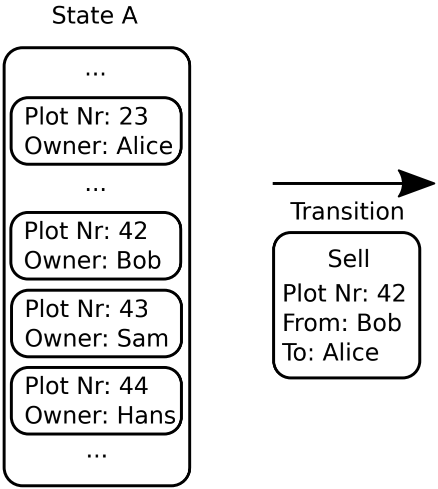
Account Owner and associated amount

Transition Transaction (Moving value from one account to another)

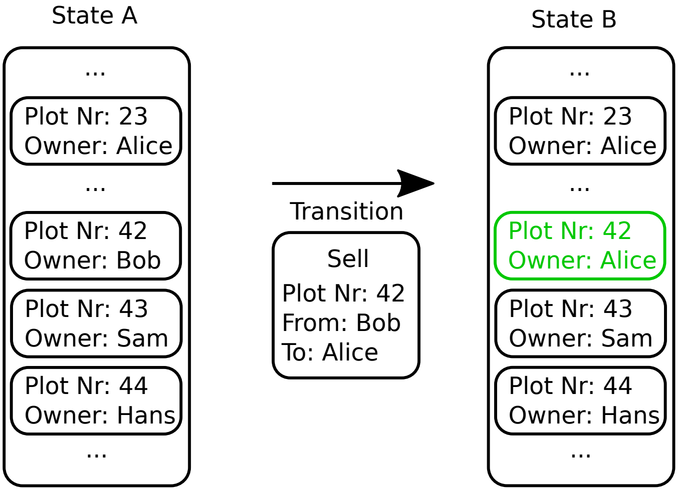
Showcase


State A
...
Plot Nr: 23
Owner: Alice
...
Plot Nr: 42
Owner: Bob
Plot Nr: 43
Owner: Sam
Plot Nr: 44
Owner: Hans
...

Showcase


State A
...
Plot Nr: 23
Owner: Alice
...
Plot Nr: 42
Owner: Bob
Plot Nr: 43
Owner: Sam
Plot Nr: 44
Owner: Hans
...
Transition
Sell
Plot Nr: 42
From: Bob
To: Alice

Showcase


State A
...
Plot Nr: 23
Owner: Alice
...
Plot Nr: 42
Owner: Bob
Plot Nr: 43
Owner: Sam
Plot Nr: 44
Owner: Hans
...
Transition
Sell
Plot Nr: 42
From: Bob
To: Alice
State B
...
Plot Nr: 23
Owner: Alice
...
Plot Nr: 42
Owner: Alice
Plot Nr: 43
Owner: Sam
Plot Nr: 44
Owner: Hans
...

Subsection

Consensus

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

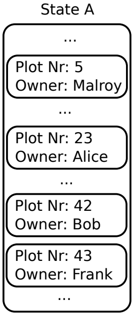
Double-spend

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

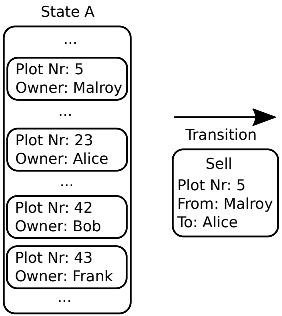
Double-spend estate example

- 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!

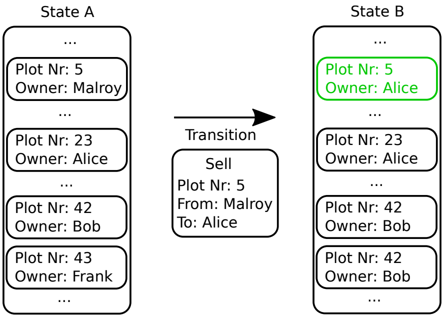
Example: Alice's view



Example: Alice's view



Example: Alice's view



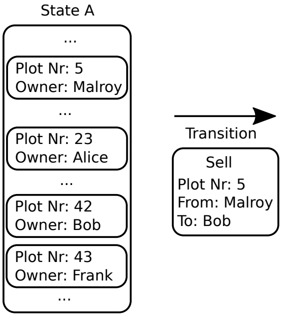
Alice has paid for the plot. She thinks it's now hers.

Example: Bob's view

- Bob does not know about the transfer of ownership from Malroy to Alice
- He still thinks the state looks like this:

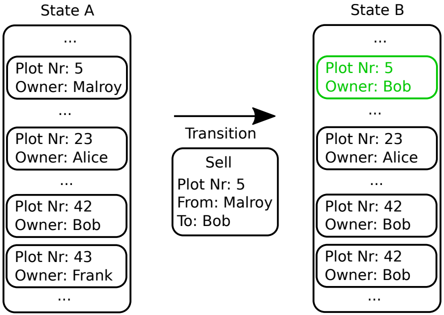
Example: Bob's view

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



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.

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

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

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.

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

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.

A Solution for the internet

Requirements:

- Decentralized
- Needs to work on the internet

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

Section 2

Blockchain (without PoW)

Another solution

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

Subsection

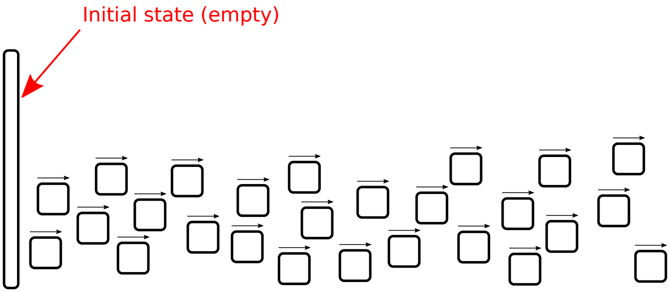
The general idea

- 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

Distributing transactions

All sorts of problems:

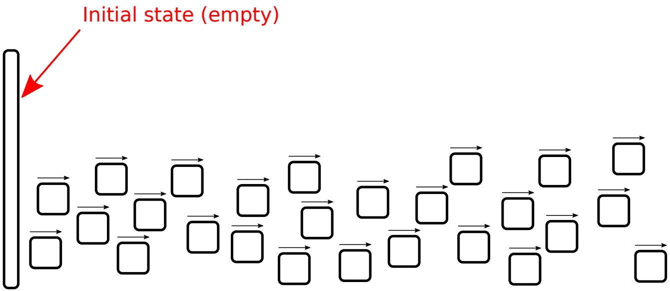
- Network out of sync



Distributing transactions

All sorts of problems:

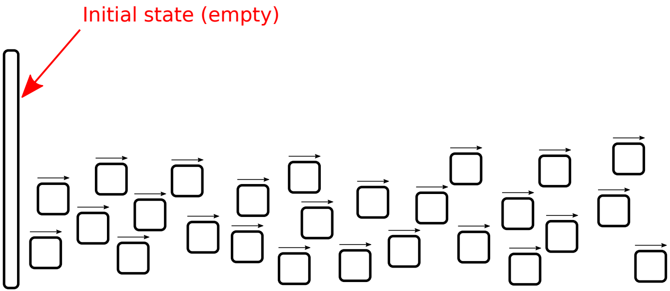
- Network out of sync
- Order unclear



Distributing transactions

All sorts of problems:

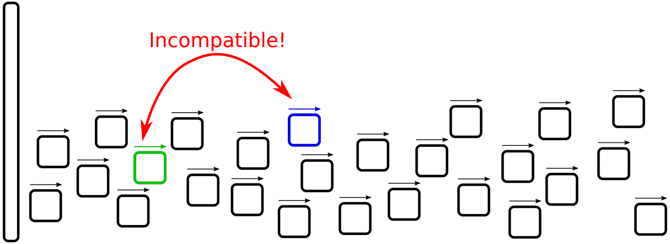
- Network out of sync
- Order unclear
- Double-spend attempts



Distributing transactions

All sorts of problems:

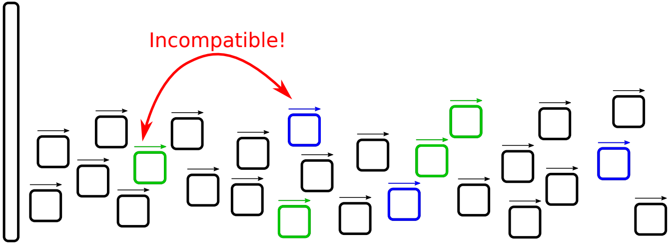
- Network out of sync
- Order unclear
- Double-spend attempts
- Conflicting transactions



Distributing transactions

All sorts of problems:

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



Subsection

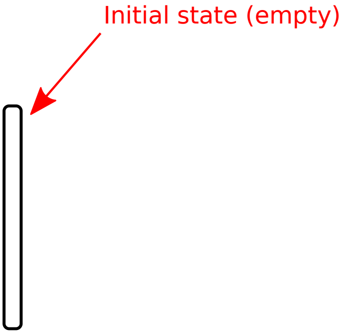
Introducing: the Blockchain

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)

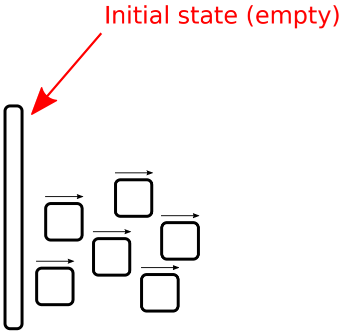
Blocks

- 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

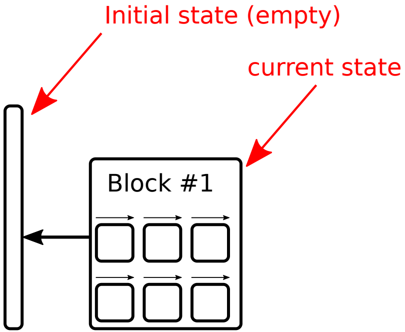
Visualization of Blockchain concepts



Visualization of Blockchain concepts

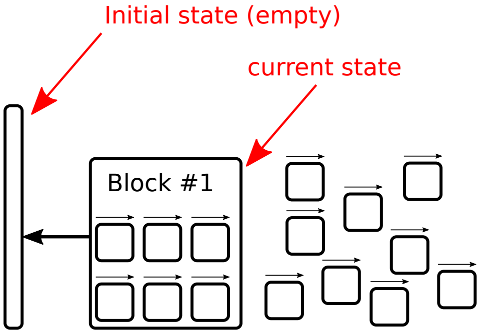


Visualization of Blockchain concepts



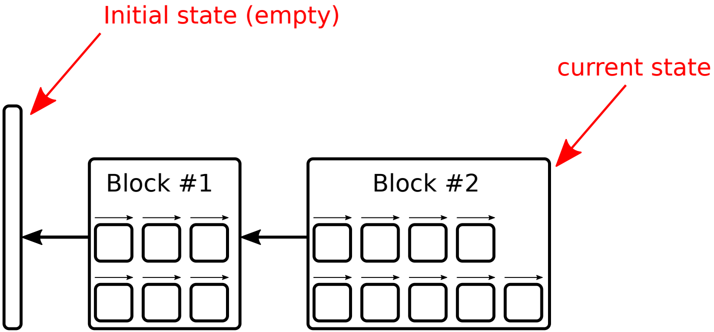
- Group transactions into blocks

Visualization of Blockchain concepts



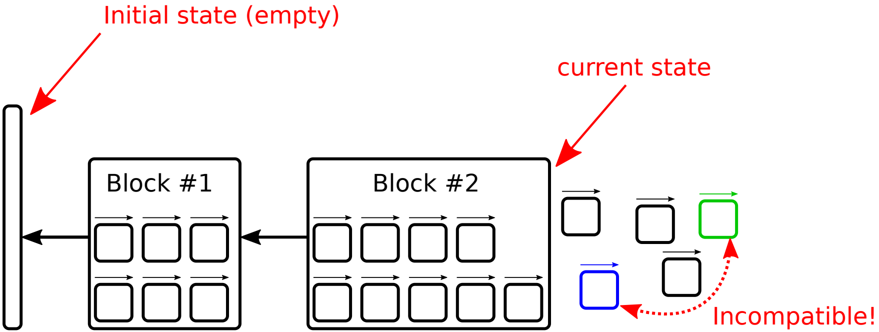
- Transactions only become part of the state once they are are inside a block

Visualization of Blockchain concepts

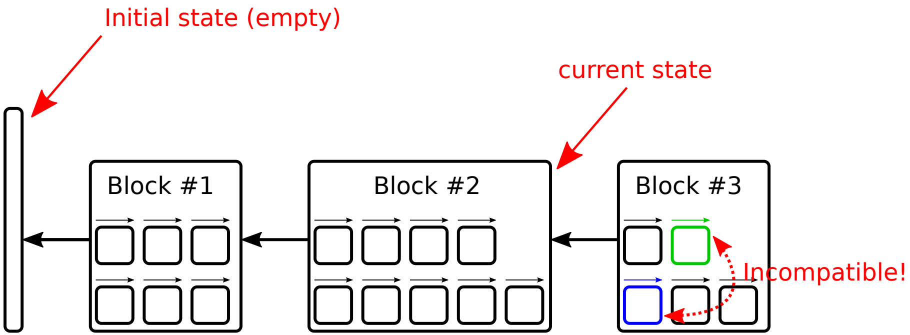


- Blocks depend on one-another
- Anyone can form a new block at any time

Visualization of Blockchain concepts

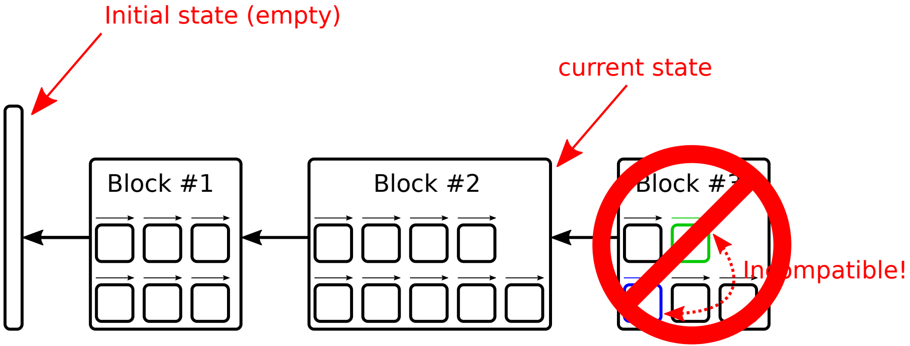


Visualization of Blockchain concepts



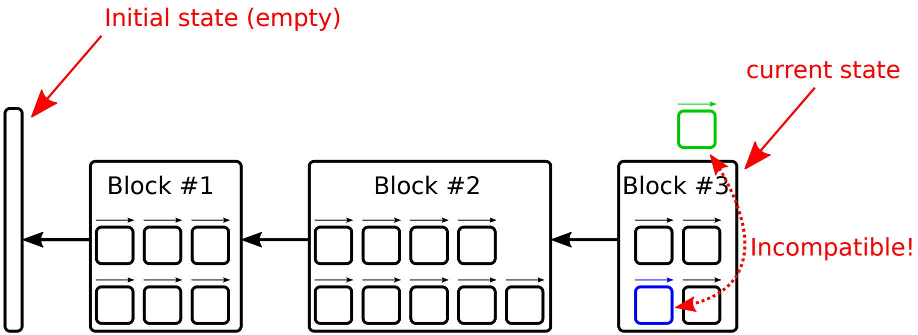
- Blocks have to meet certain criteria:
  - Transactions within the block have to be compatible with one another

Visualization of Blockchain concepts



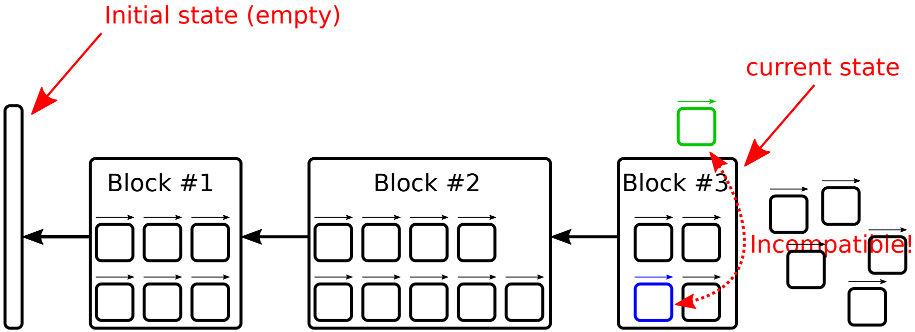
- Blocks have to meet certain criteria:
  - Transactions within the block have to be compatible with one another

Visualization of Blockchain concepts

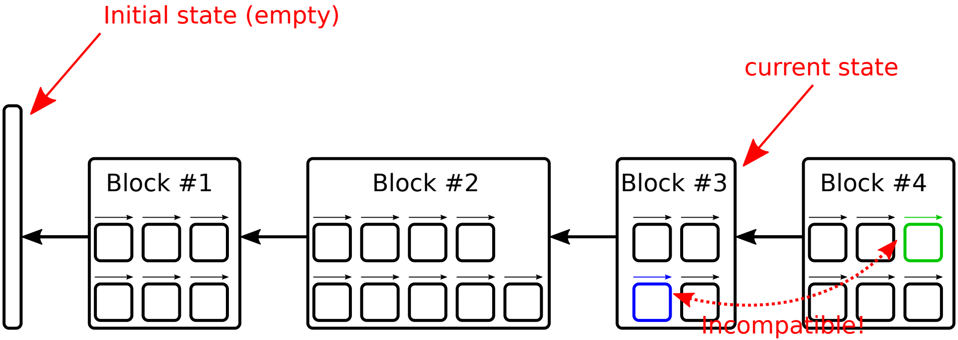


- Blocks have to meet certain criteria:
  - Transactions within the block have to be compatible with one another

Visualization of Blockchain concepts

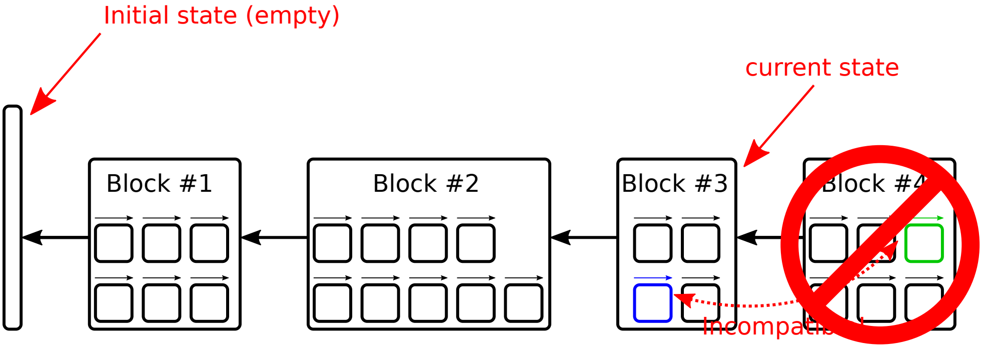


Visualization of Blockchain concepts



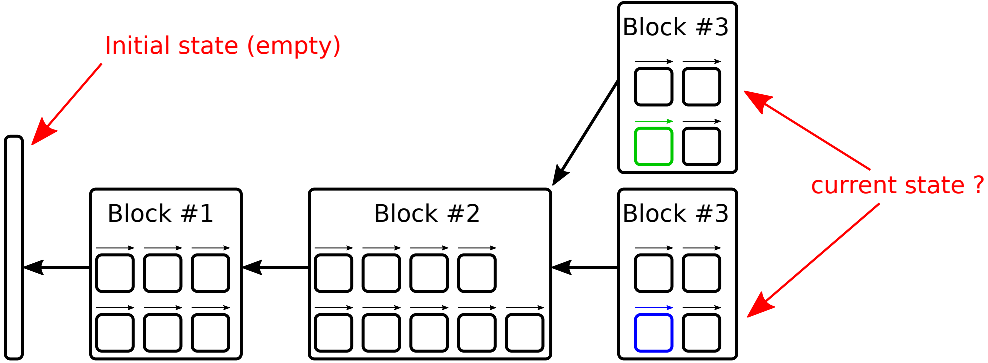
- Blocks have to meet certain criteria:
  - Transactions within the block have to be compatible with transactions in earlier blocks

Visualization of Blockchain concepts



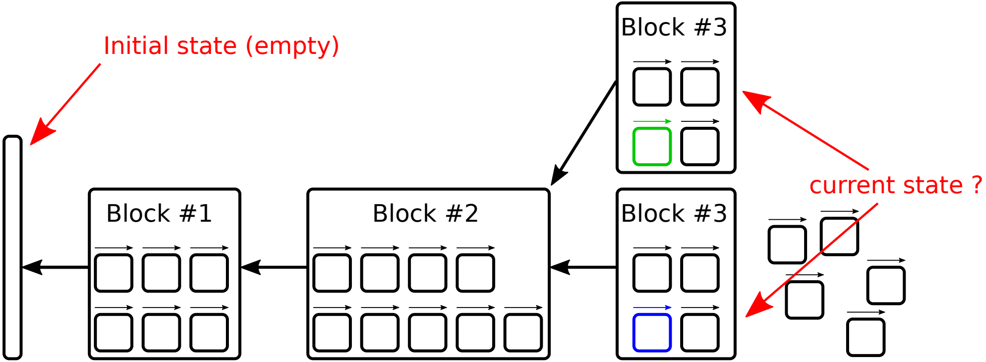
- Blocks have to meet certain criteria:
  - Transactions within the block have to be compatible with transactions in earlier blocks

Visualization of Blockchain concepts



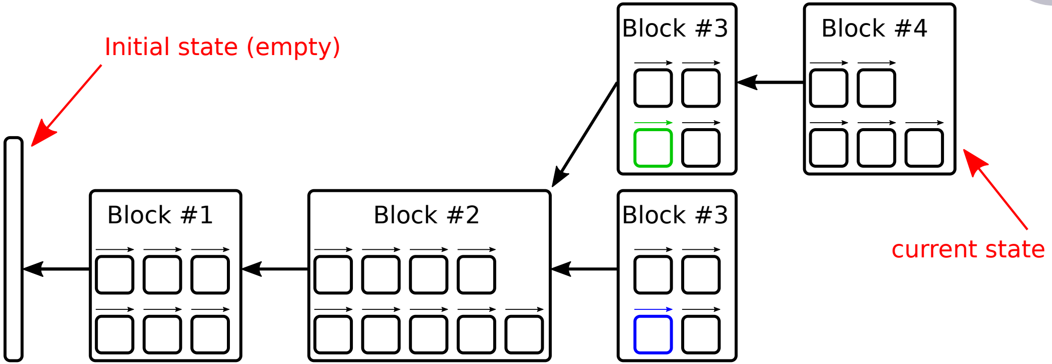
- We define the current state as: All transactions within the longest branch of blocks applied on an empty state

Visualization of Blockchain concepts



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

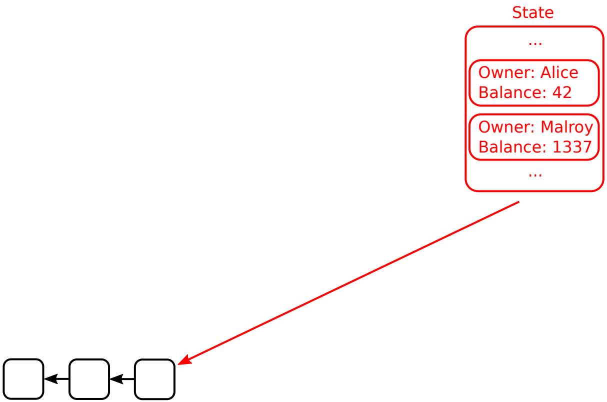
Visualization of Blockchain concepts



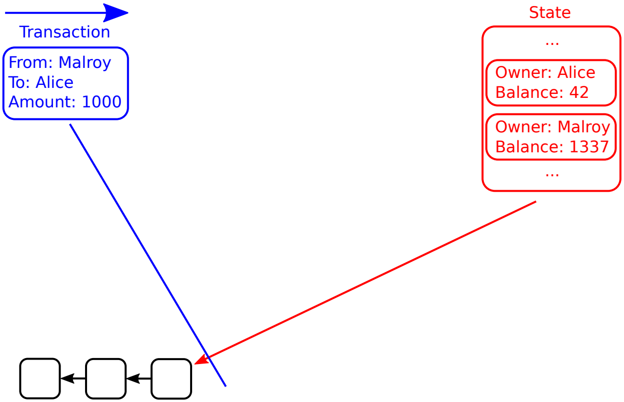
- We define the current state as: All transactions within the longest branch of blocks applied on an empty state

Subsection

Double-spend on a Blockchain without POW

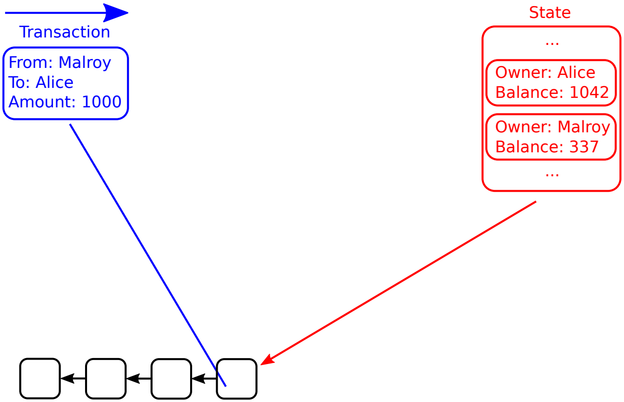


Double-spend on a Blockchain without POW



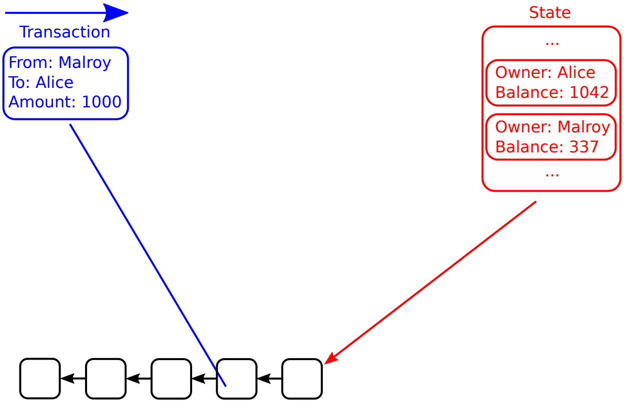
- Malroy creates the transaction to buy the bicycle

Double-spend on a Blockchain without POW

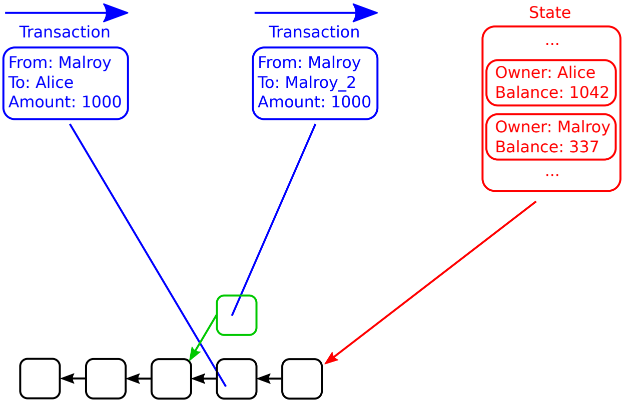


- The transaction becomes part of the state
- Alice now has the money
- Malroy gets the bicycle

Double-spend on a Blockchain without POW

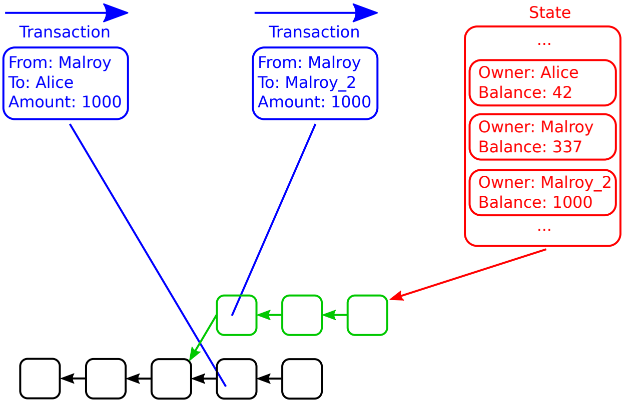


Double-spend on a Blockchain without POW



- Malroy creates a new block with a different transaction

Double-spend on a Blockchain without POW



- Malroy creates a lot of blocks very quickly
- Malroy's branch becomes the state
- Alice does not have the money
- Malroy still has the bicycle

Problems

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

Section 3

Blockchain

Subsection

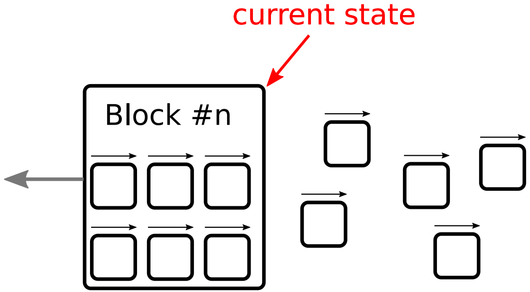
The problem

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

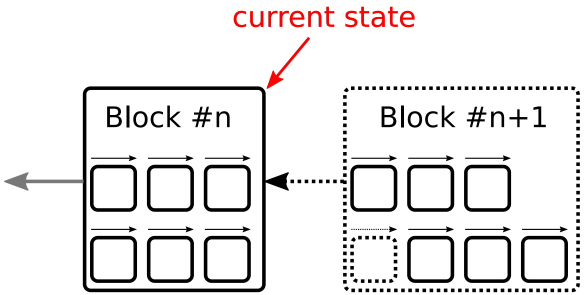
PoW

- 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

Creating PoW protected blocks

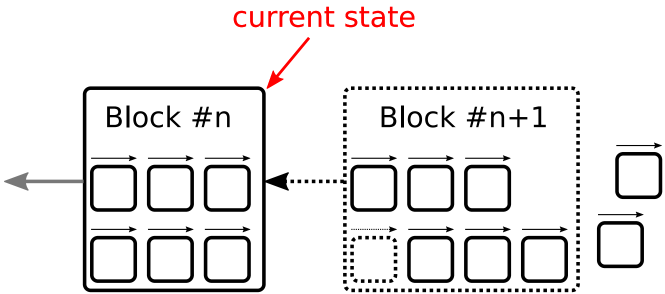


Creating PoW protected blocks



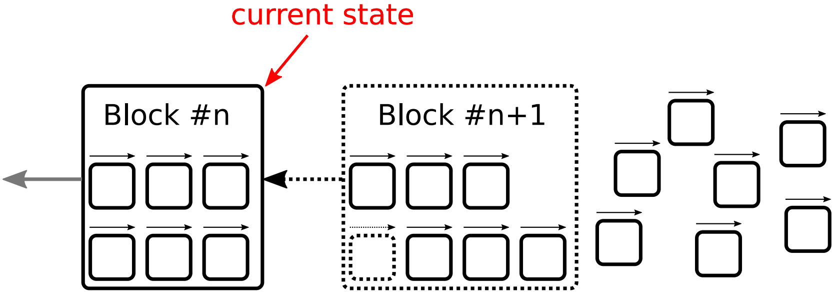
- (The additional transaction is a Coinbase (see below))

Creating PoW protected blocks



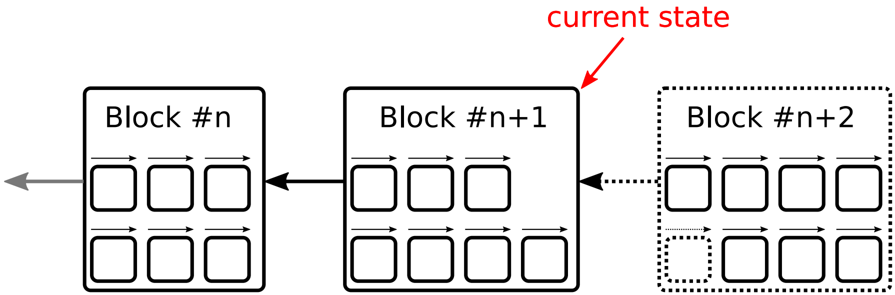
- Work on the block is going on
- New transactions arrive

Creating PoW protected blocks



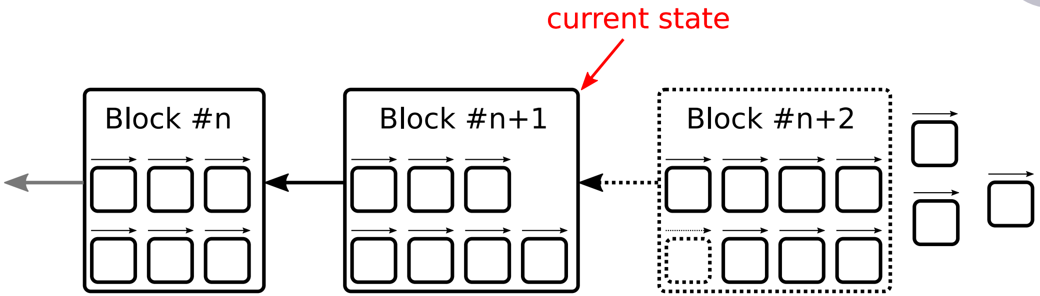
- Work on the block is going on
- New transactions arrive

Creating PoW protected blocks



- The block is completed
  - PoW created successfully
- Work on the next block starts immediately

Creating PoW protected blocks



- The block is completed
  - PoW created successfully
- Work on the next block starts immediately

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

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

Reward

For mining to be worthwhile, a reward is needed:

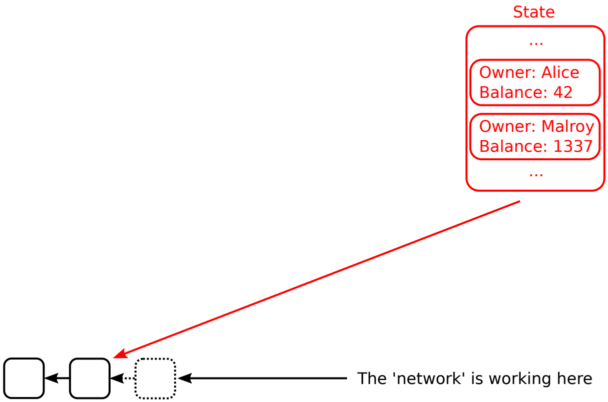
- A transaction from nowhere to the miner
  - Transaction fees
  - Coinbase / new money
- Included in the block
- Makes blocks individual
  - Ensures distribution of success amongst all miners

Mining I

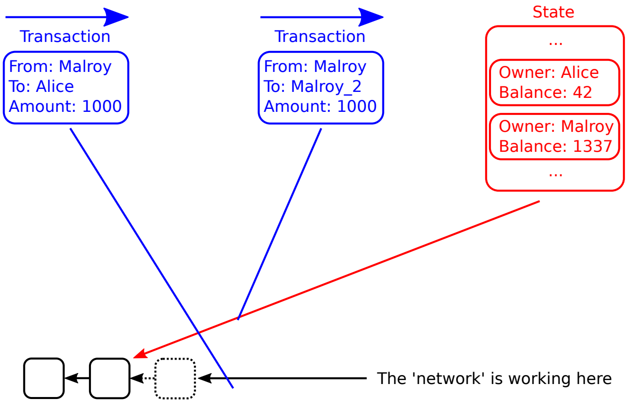
- Which branch should a miner work on?
- For the reward to be spendable, it needs to be part of the state
- The state is the "longest" branch
- So it only makes sense to work on the "longest" branch
- Works without any coordination!

Subsection

Double-spend

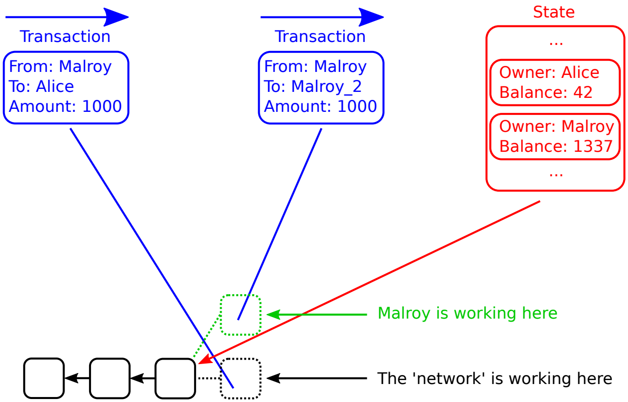


Double-spend



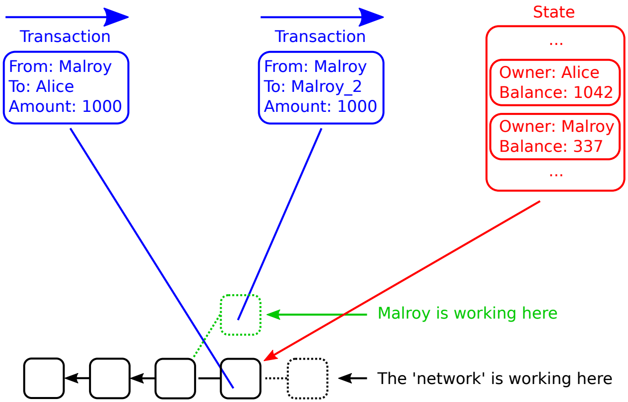
- Malroy creates two transactions
- Only the legitimate one gets published

Double-spend



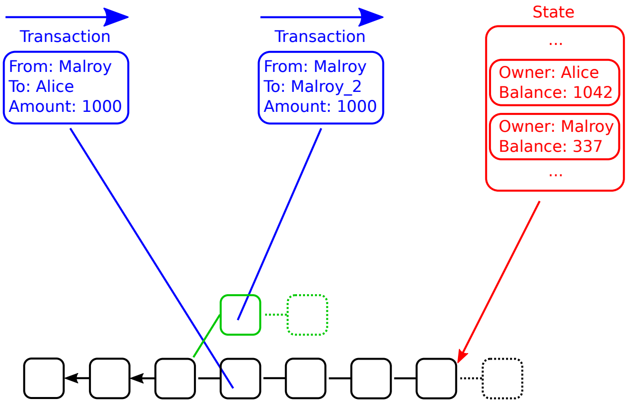
- Work on the next block starts
- Malroy is the only one working on a different block

Double-spend



- The network has more power than Malroy
- Thus they create new blocks faster

Double-spend

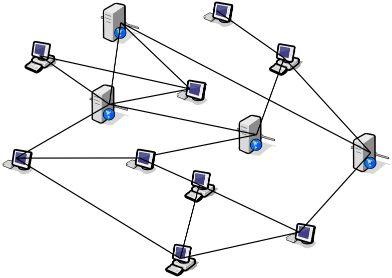


- Malroy's chain never becomes the longest

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

p2p networking attack setup

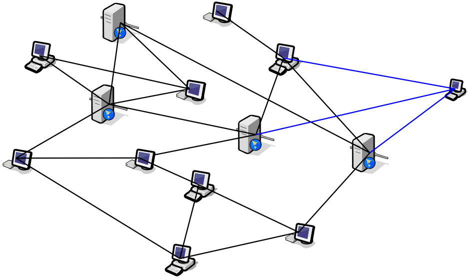


Most Blockchain currencies use a p2p network

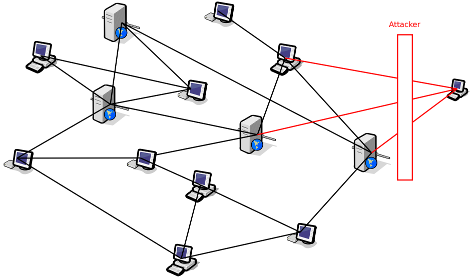
- to distribute the transactions
- to distribute the blocks

p2p networking attack setup

Malroy wants to control the victim's (Bob's) view of the network by...



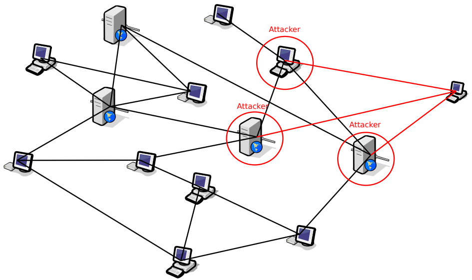
p2p networking attack setup



...Controlling Bob's network connection. That can happen sometimes

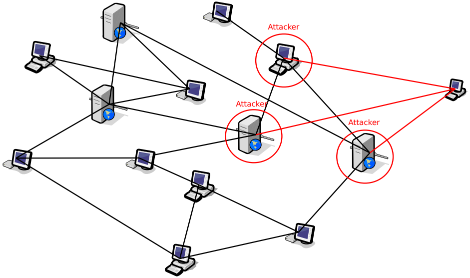
- network provider
- public wifi

p2p networking attack setup



...Controlling the peers Bob is connected to.

p2p networking attack setup



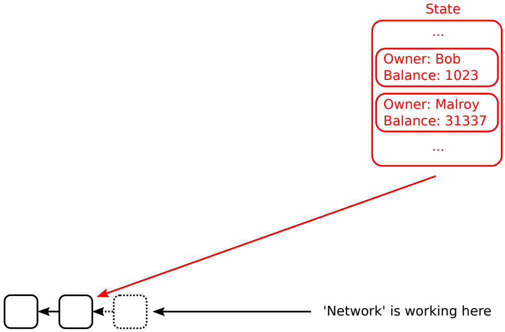
This allows Malroy to:

- Hide transactions from Bob
- Hide blocks from Bob
- Present extra transactions and / or blocks to Bob

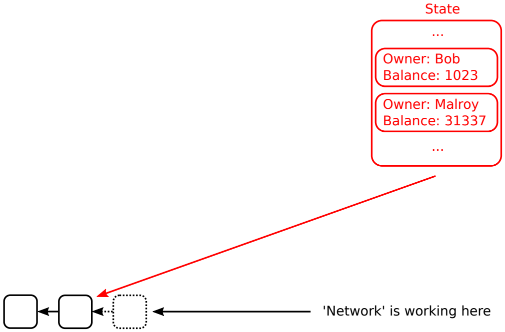
In short, Malroy can maintain a fork of the Blockchain just for Bob. This allows Malroy to stage a different kind of double-spend attack:

Network attack

Public network view:

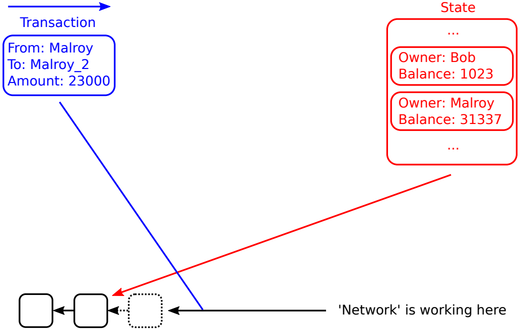


Bob’s view:

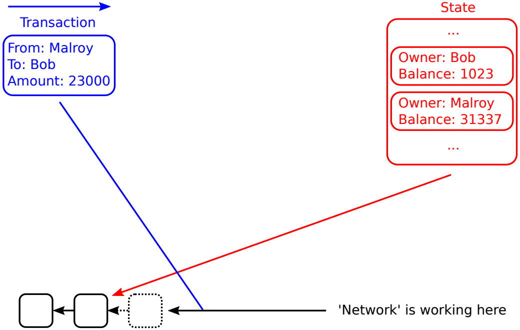


Network attack

Public network view:



Bob's view:



Malroy:

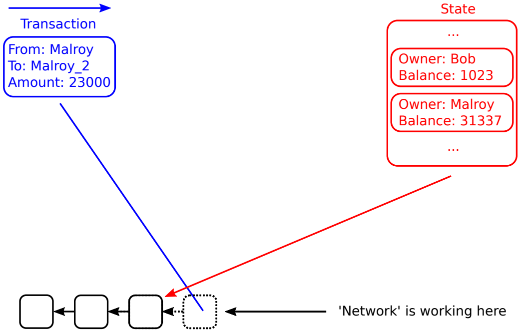
- Creates a transaction to Malroy_2
- Puts it onto the network

Malroy:

- Creates a transaction to Bob
- Sends it to Bob only

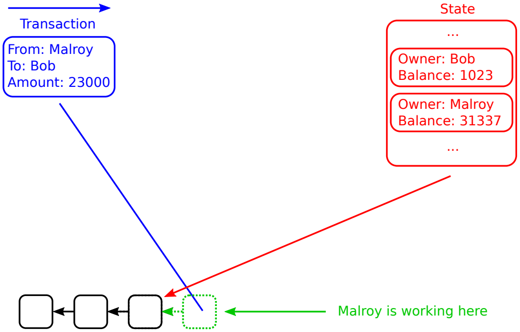
Network attack

Public network view:



- Work on the next block starts

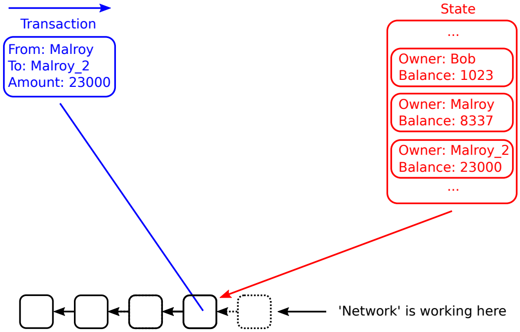
Bob's view:



- Malroy starts to work on the new block

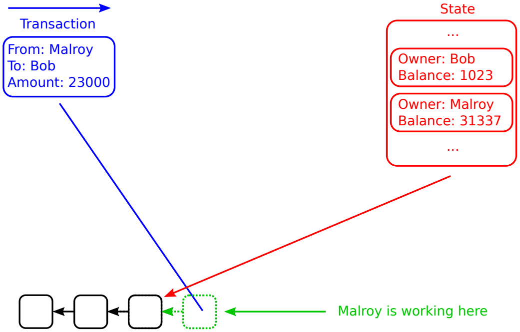
Network attack

Public network view:



- The block gets created

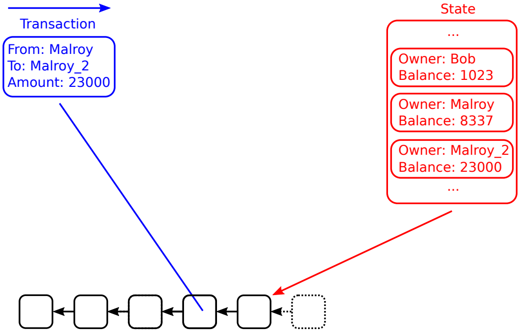
Bob’s view:



- Malroy is the only one working on this branch

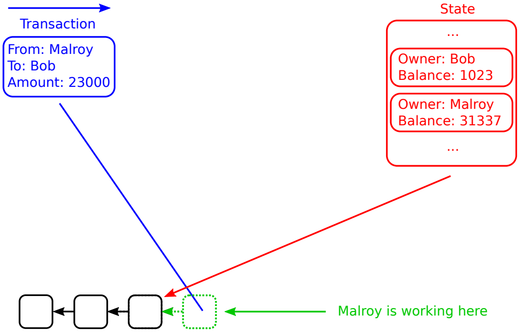
Network attack

Public network view:



- New blocks get added to the chain

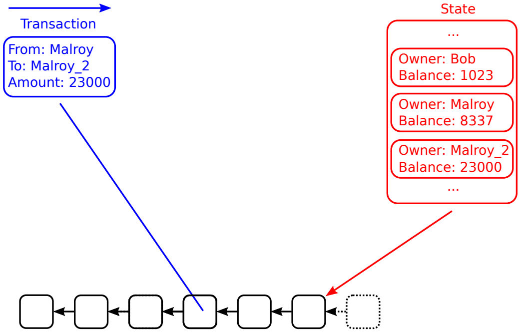
Bob's view:



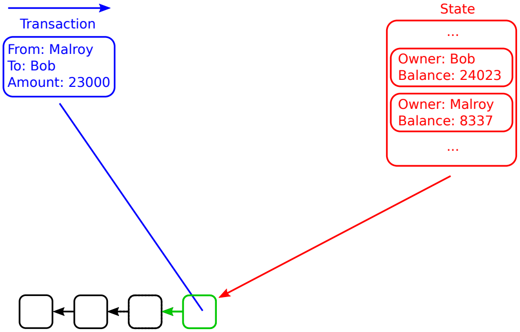
- Malroy is the only one working on this branch

Network attack

Public network view:



Bob's view:



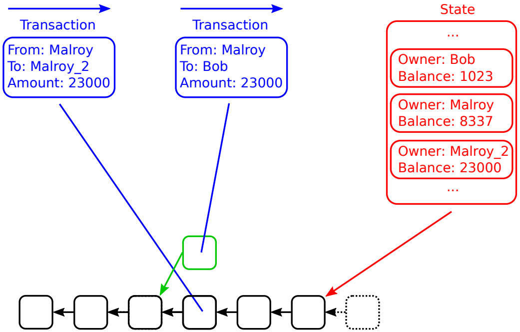
- Bob can't see the other, longer branch
- Thus in his view, the transaction becomes part of the state
- He has the money, hands over the car

Network attack

- The attack stops
- Bob synchronizes with the rest of the network
- The two chains merge

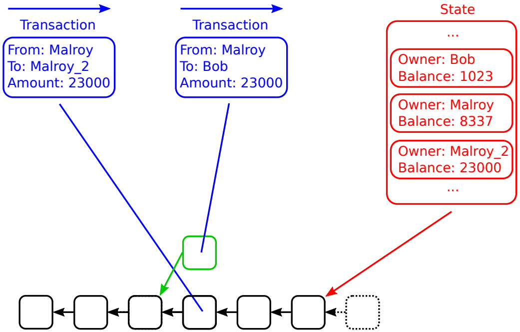
Network attack

- The attack stops
- Bob synchronizes with the rest of the network
- The two chains merge



Network attack

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



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

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?

Cost of the attack

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

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 to create one block

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

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 to create one block

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

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

Reward in 50min mining

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

Protection from the attack

Malroy can chose:

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

Protection from the attack

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.

Waiting

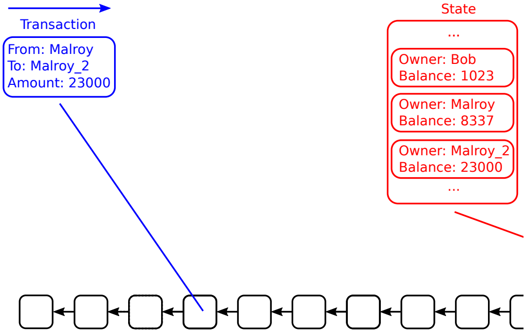
By waiting another block, Bob makes the attack twice as expensive.
So the larger a transaction we protect, the longer we have to wait. You can easily calculate for how long you have to wait to be secure:

How long to wait?

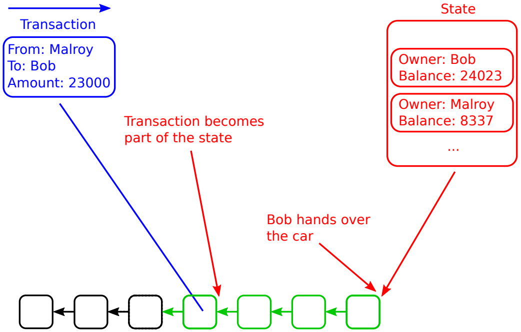
$$\lceil amount \div Blockreward \rceil + 1$$

Bob’s protection

Public network view:



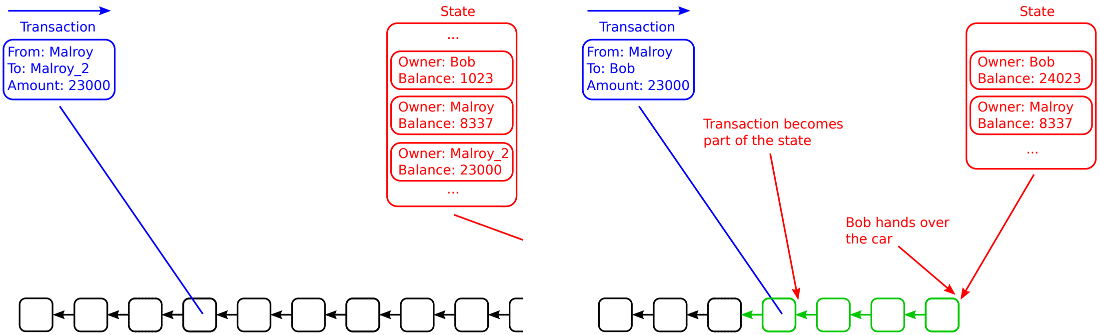
Bob’s view:



Bob's protection

Public network view:

Bob's view:



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

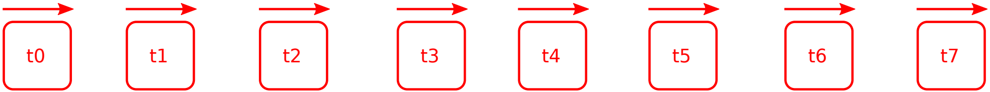
Section 4

Bitcoin

Subsection

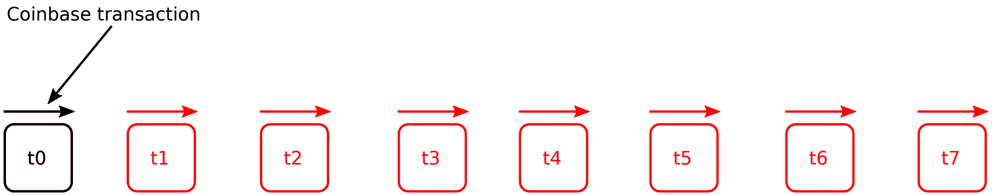
Body

- Contains transactions
- Ordered
  - Dependant transactions are in the correct order



Body

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

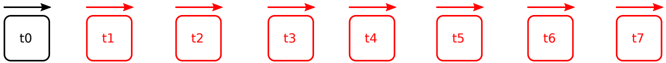


Merkle tree

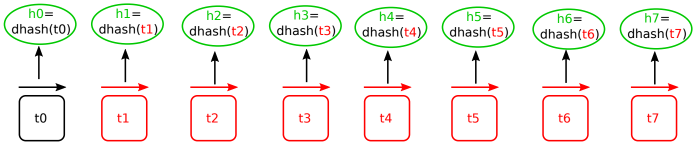
Bitcoin uses a Merkle tree to secure the transactions. Root of the Merkle tree is part of the Block header.

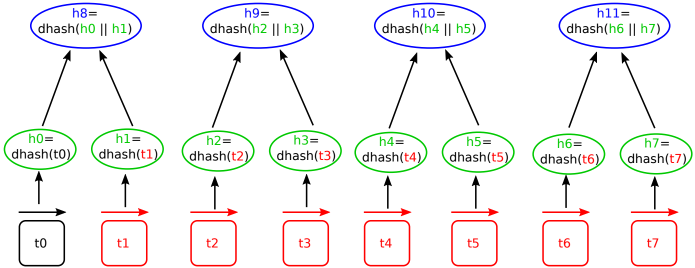
- Binary tree
- A node is the hash of the two child nodes
- Bitcoin uses sha256 double hashes
    - aka. dhash, double-sha256
    - $sha256(sha256(...))$

Merkle tree visualization



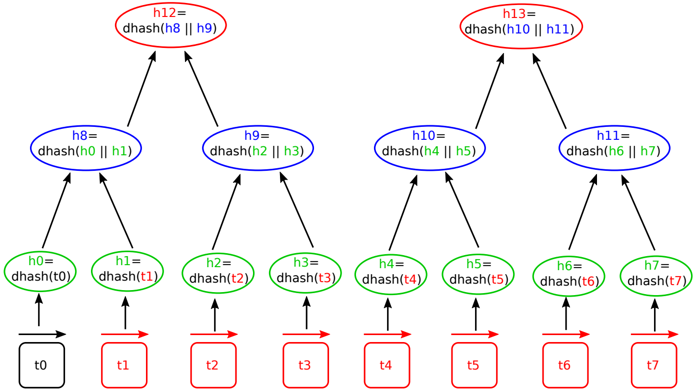
Merkle tree visualization



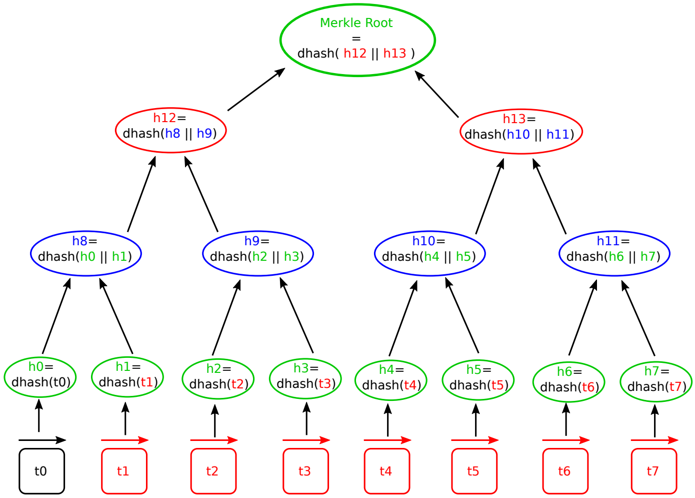


h8=
dhash(h0 || h1)
h9=
dhash(h2 || h3)
h10=
dhash(h4 || h5)
h11=
dhash(h6 || h7)
h0=
dhash(t0)
h1=
dhash(t1)
h2=
dhash(t2)
h3=
dhash(t3)
h4=
dhash(t4)
h5=
dhash(t5)
h6=
dhash(t6)
h7=
dhash(t7)
t0
t1
t2
t3
t4
t5
t6
t7

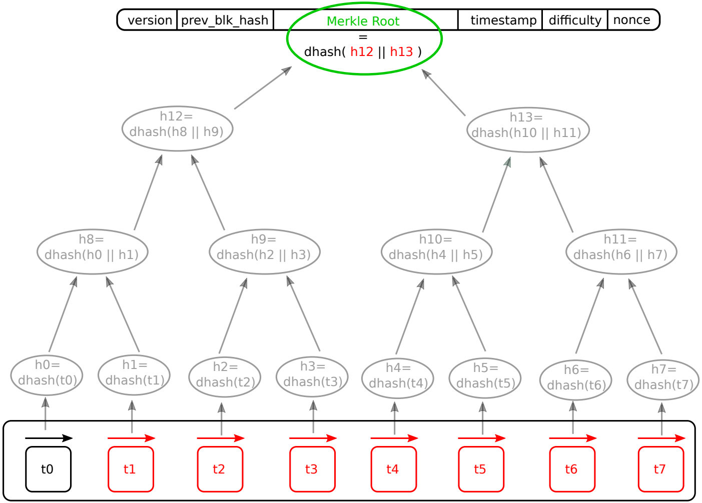
Merkle tree visualization



Merkle tree visualization



Merkle tree visualization



Header

version	prev_blk_hash	mrkl_root	timestamp	difficulty	nonce
32 Bit	256 Bit	256 Bit	32 Bit	32 Bit	32 Bit

- **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.

Bitcoin mining

PoW:

- Bitcoin uses the Block Hash for PoW
  - $dhash(block\_header)$
- PoW: First n Bits of the Block Hash need to be 0

Bitcoin mining

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

Bitcoin mining

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

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

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

Subsection

Types of Bitcoin clients

- Full node
  - Stores the entire Blockchain
  - Seeds the Blockchain to the network
  - Validates every block
  - Validates every transaction

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

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

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!

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!

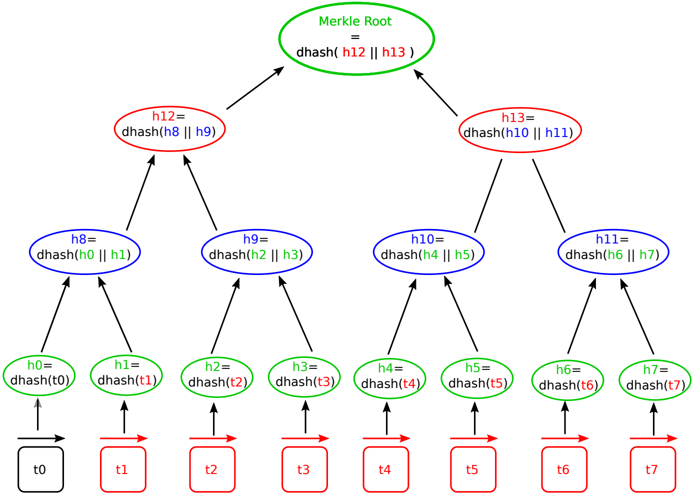
Advantage:

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

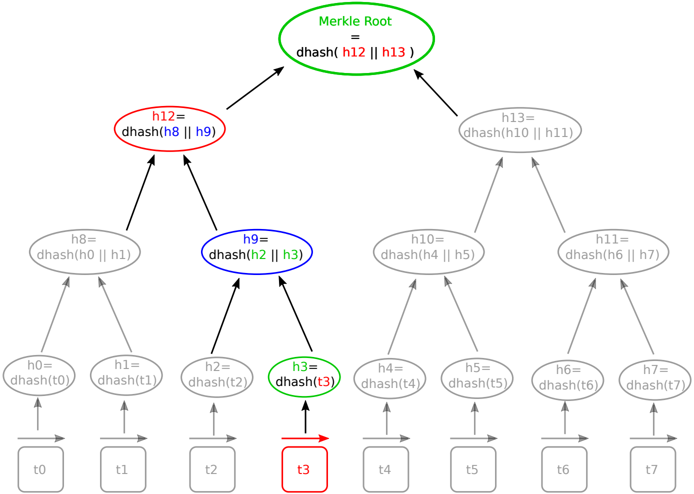
Merkle branch

- 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

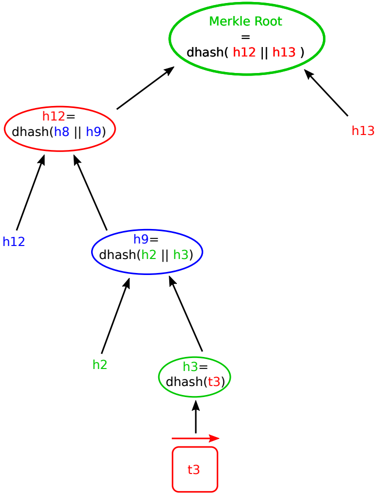
Merkle branch visualization



Merkle branch visualization



Merkle branch visualization



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

SPV

- Simple Payment Verification
- Proof that a transaction is part of the chain
- Avoid downloading the entire chain

SPV

- Simple Payment Verification
- Proof that a transaction is part of the chain
- Avoid downloading the entire chain

1. Retrieve all block headers

SPV

- 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

SPV

- 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

SPV

- 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

SPV

- 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

Questions?

Ask them now or contact me:

33c3 DECT / GSM

- 5346 (LEGO)
- 7869 (PUNX)

email

- niklaus@mykolab.ch

XMPP

- vimja@xmpp.honet.ch

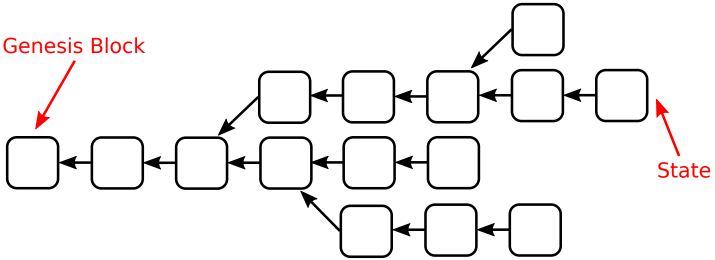
Section 5

Bonus Slides

Subsection

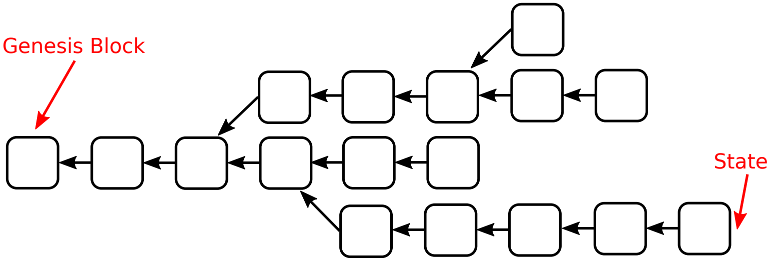
Status switch between branches

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.



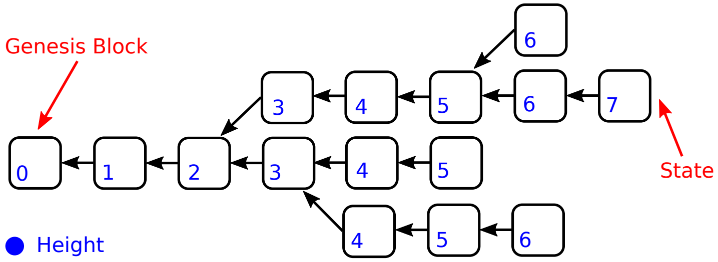
Status switch between branches

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.



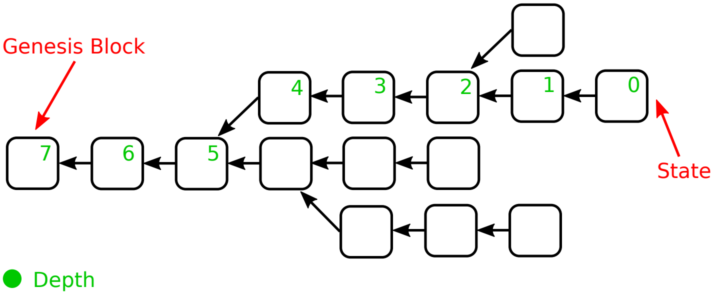
Block height and depth

Height identifier



Block height and depth

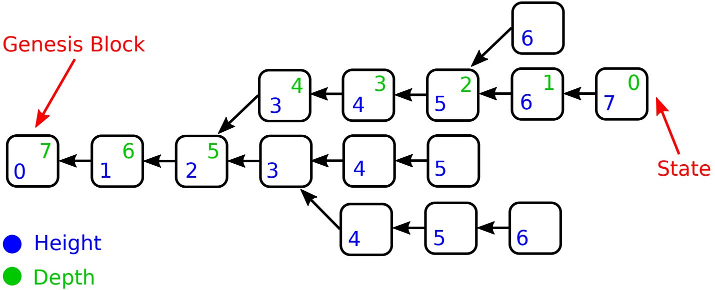
Depth expression of security



Block height and depth

Height identifier

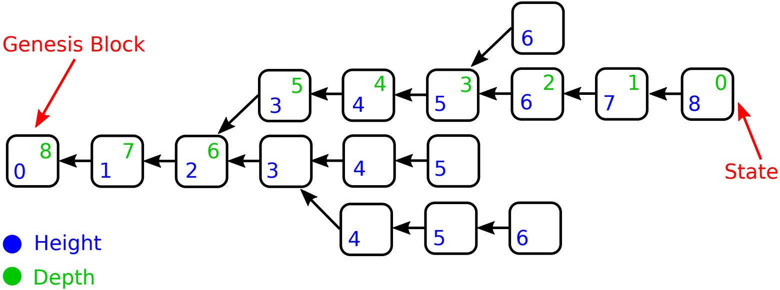
Depth expression of security



Block height and depth

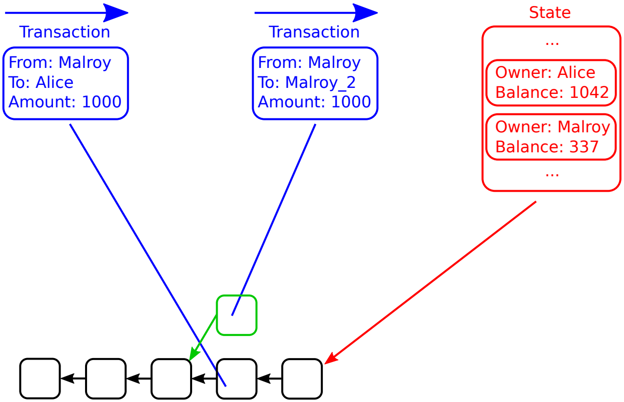
Height identifier

Depth expression of security



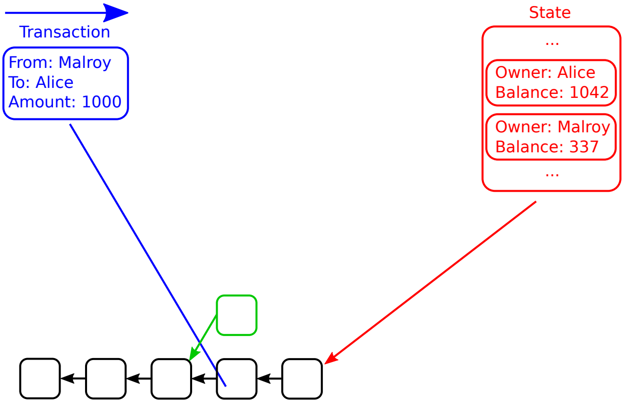
Subsection

Conflicting transaction in double-spend attacks



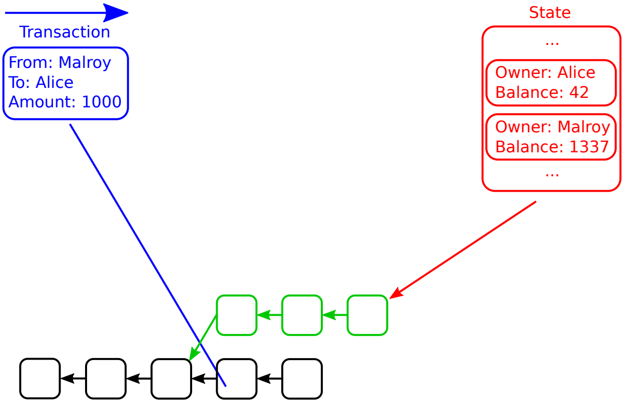
- What is the additional transaction for?

Conflicting transaction in double-spend attacks

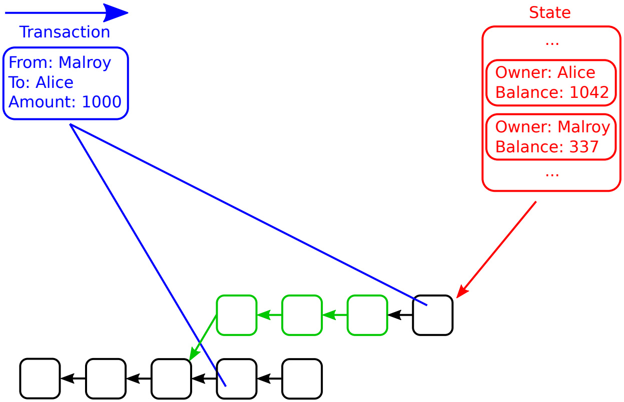


- Let's see what happens without it

Conflicting transaction in double-spend attacks

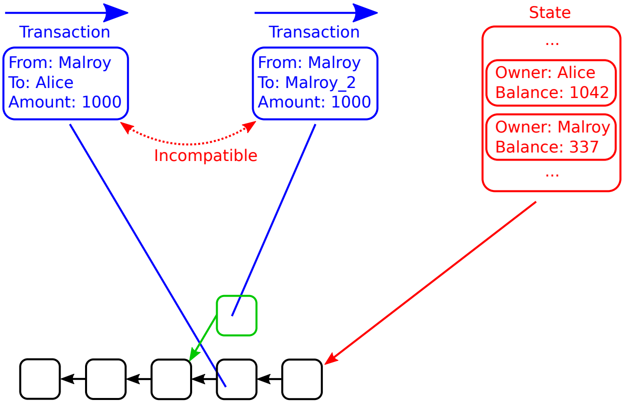


Conflicting transaction in double-spend attacks



- Someone could create a new block containing the transaction
- This would make a valid block
  - It is not in conflict with any previous block
- The transaction becomes part of the new state
- Malroy's money is gone

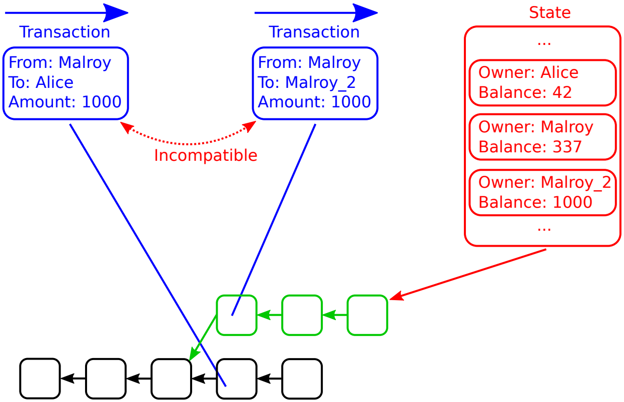
Conflicting transaction in double-spend attacks



So to prevent that:

- Malroy creates a conflicting transaction
- Puts it into the state

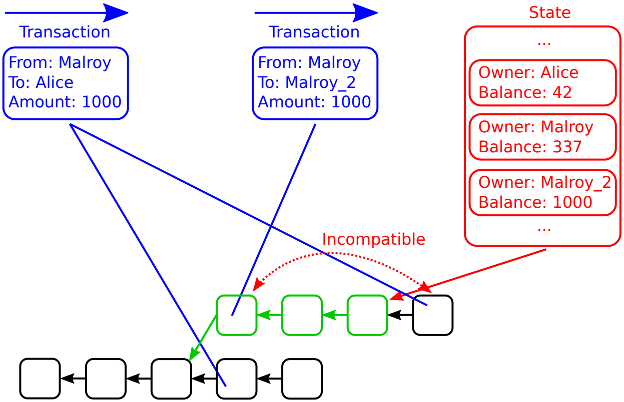
Conflicting transaction in double-spend attacks



So to prevent that:

- Malroy creates a conflicting transaction
- Puts it into the state

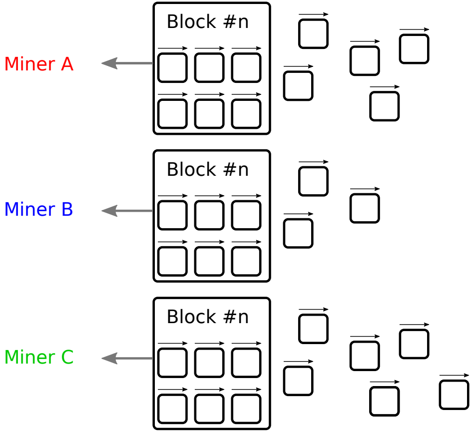
Conflicting transaction in double-spend attacks



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

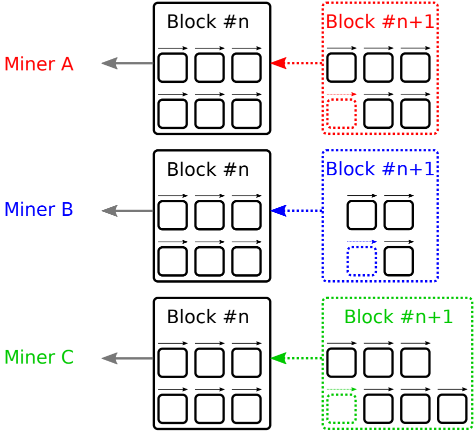
Subsection

Example scenario with multiple miners



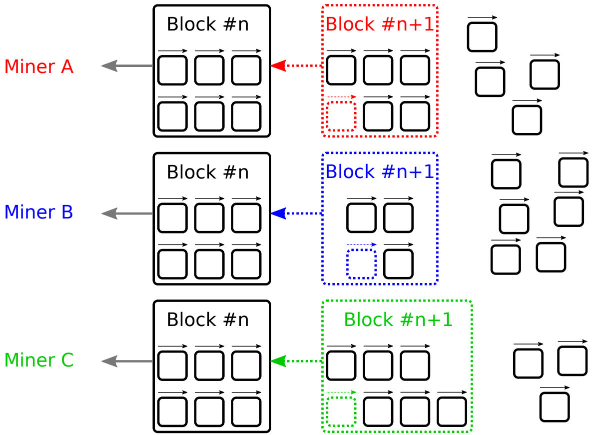
- Miners have different sets of transactions in their transaction pools
- Due to network delays

Example scenario with multiple miners



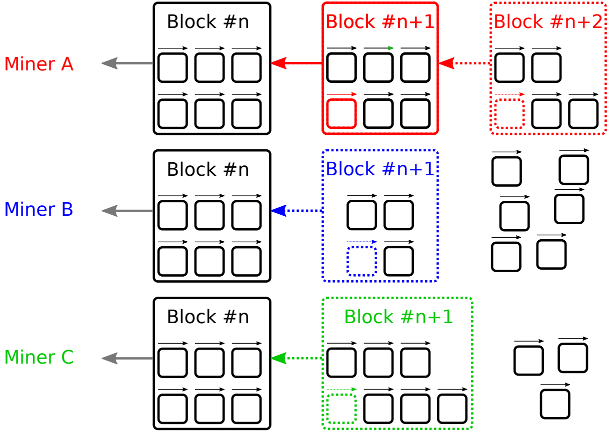
- Block #n gets completed
- All miners immediately start work on Block #n+1

Example scenario with multiple miners



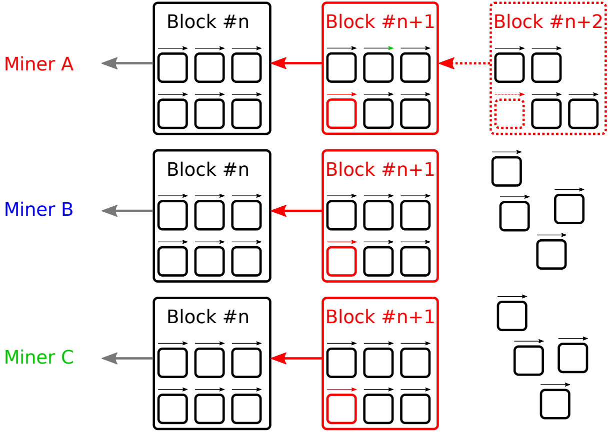
- New transactions keep arriving
- Miners add them to their transaction pools

Example scenario with multiple miners



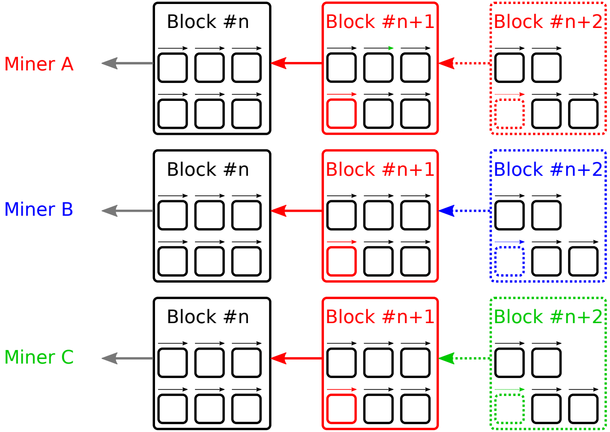
- Miner A finishes Bock #n+1
- Miner A immediately starts work on #n+2

Example scenario with multiple miners



- 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)

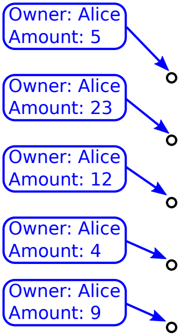
Blockchain

Bitcoin

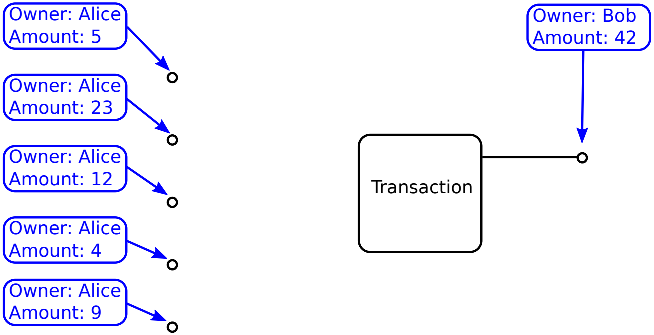
Bonus Slides

- Branches in the chain
- Double-spend
- Mining
- Pruning client

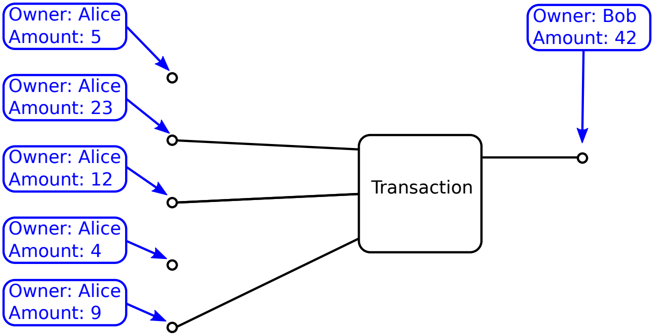
Bitcoin transaction



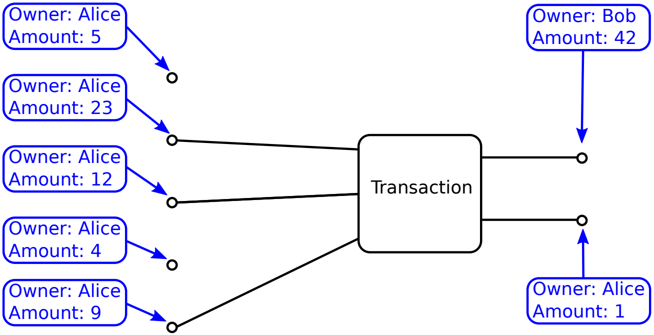
Bitcoin transaction



Bitcoin transaction



Bitcoin transaction



Bitcoin transaction