IT486 v3.0: Blockchains and Cryptocurrencies

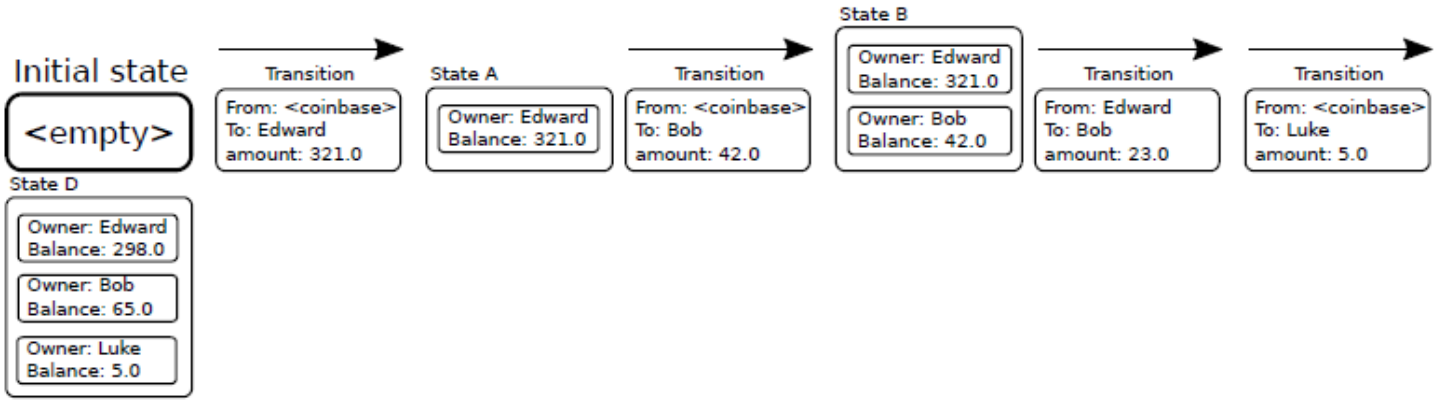
Blockchains, POW, and mining

State transition systems

- We can map any financial system to a state-transition system
- E.g., for a bank the mapping looks something like this
  - account - owner and associated amount
  - state - collection of all accounts
  - transition - transaction (moving value from one account to another)
- Also works for other systems of ownership, e.g. estate

State transition systems

- No need to remember the intermediate states
- It is enough if (an empty) start state and all transitions are known



Consensus

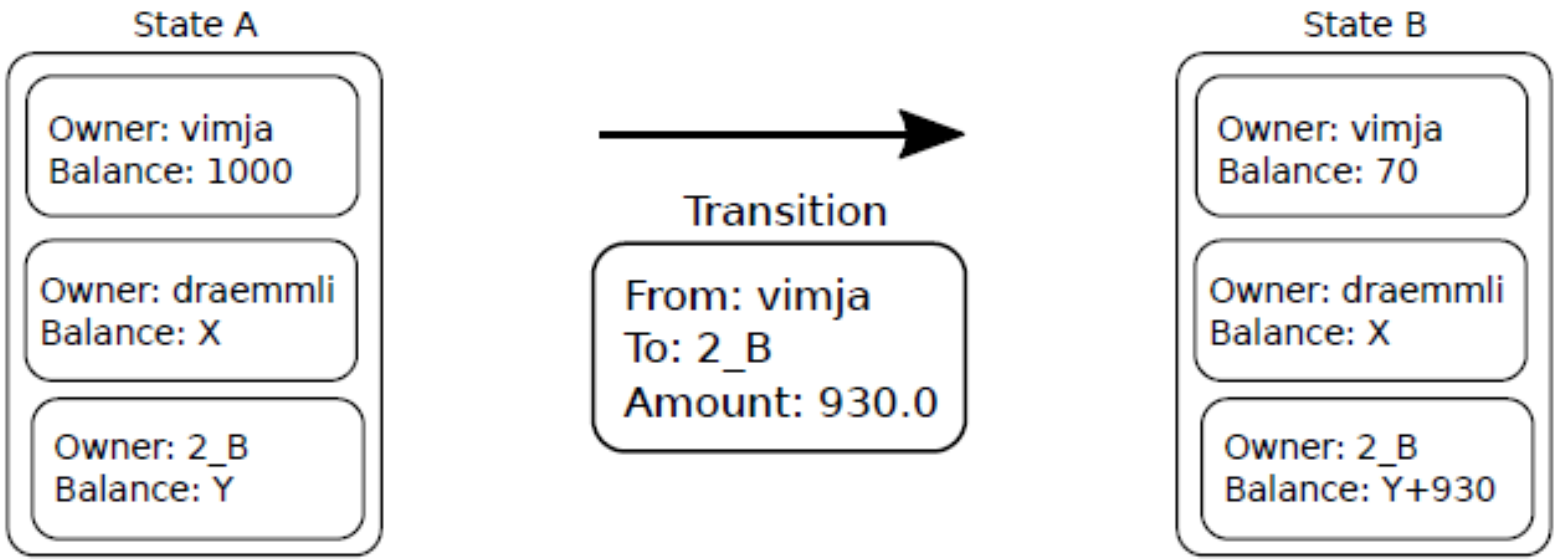
- Before every payment, all parties need to agree on the current state!
- The reasons for this: double spend
- Well known attack in the Bitcoin world

Double spend example

- vimja has 1000
- 2_B has for sale a cool bicycle for 930
- draemmli has for sale an old computer for 750
- I want to buy both!

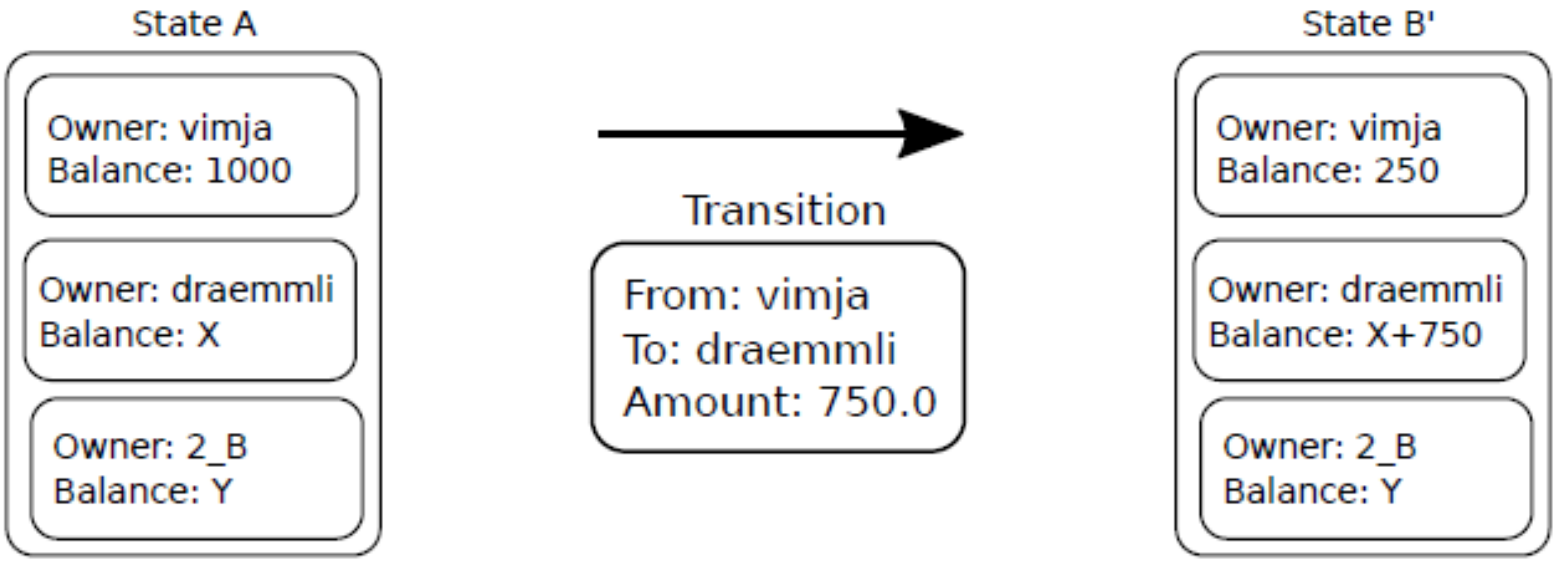
Example: 2_B's view

- 2_B has got the money. He gives me the bicycle



Example: draemmli's view

- Assume draemmli doesn't know about the transaction from me to 2_B
- He still thinks the state looks like this:



- draemmli has got the money. He now gives me the computer

Observations

- 2_B and draemmli do not agree on the state
- Their views on the state of the system are incompatible
- This breaks the system
- Before we consider a possible solution, lets consider another example

Real estate example

- I'm selling an estate at Pangong lake
  - At 14:00 hrs, I meet with 2_B
  - I sell him the estate, he gets a receipt
  - At 15:00 hrs, I meet with draemmli
  - I sell him the estate, he too gets a receipt
- But they can't both own the same estate!

Real estate example

- I'm selling an estate at Pangong lake
  - At 14:00 hrs, I meet with 2_B
  - I sell him the estate, he gets a receipt
  - At 15:00 hrs, I meet with draemmli
  - I sell him the estate, he too gets a receipt
- But they can't both own the same estate!
- The estate world has a simple solution for this

Real estate traditional solution

- A central authority (Registry of deeds) 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
  - either accepts or rejects it

Solution requirements

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

Solution requirements

- All parties need to agree on the current state
- All parties need to agree on whether a transition is valid
- A central authority controlling the state is one way of achieving these requirements
- In traditional monetary systems, banks control the state of the system

A blockchain solution

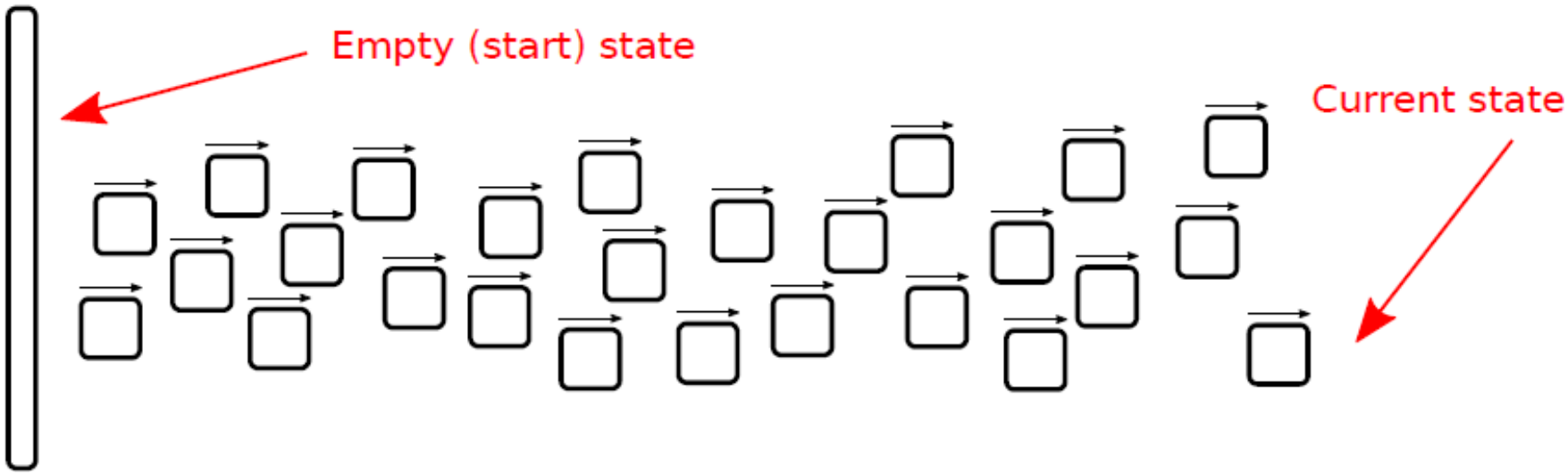
- Blockchains can serve as a solution to the same problem
- A blockchain solution provides amazing properties
  - No central authority
  - Parties do not need to trust each other
  - Parties need no information on who is participating
  - . . . not even any information on how many others are participating

A blockchain solution

- Blockchains can serve as a solution to the same problem
- A blockchain solution provides 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!

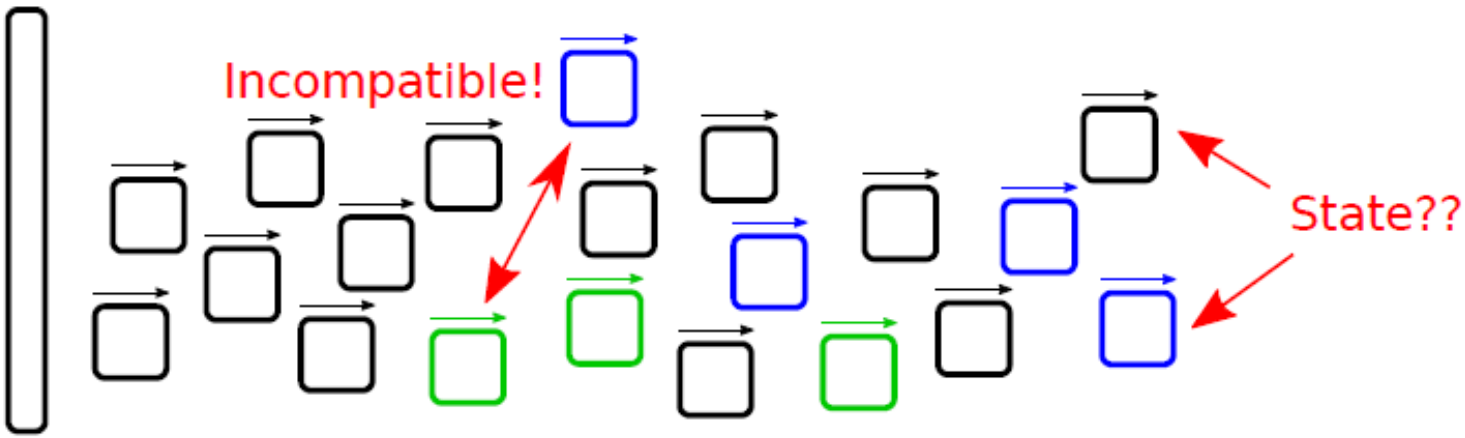
Basic idea

- We publish all transactions
- We define the current state as: all transactions applied to an empty state (valid transactions only)



Problems

- What order do we apply the transactions in?
- What happens on double-spends?
  - we would need to collectively decide which of the two transactions we accept

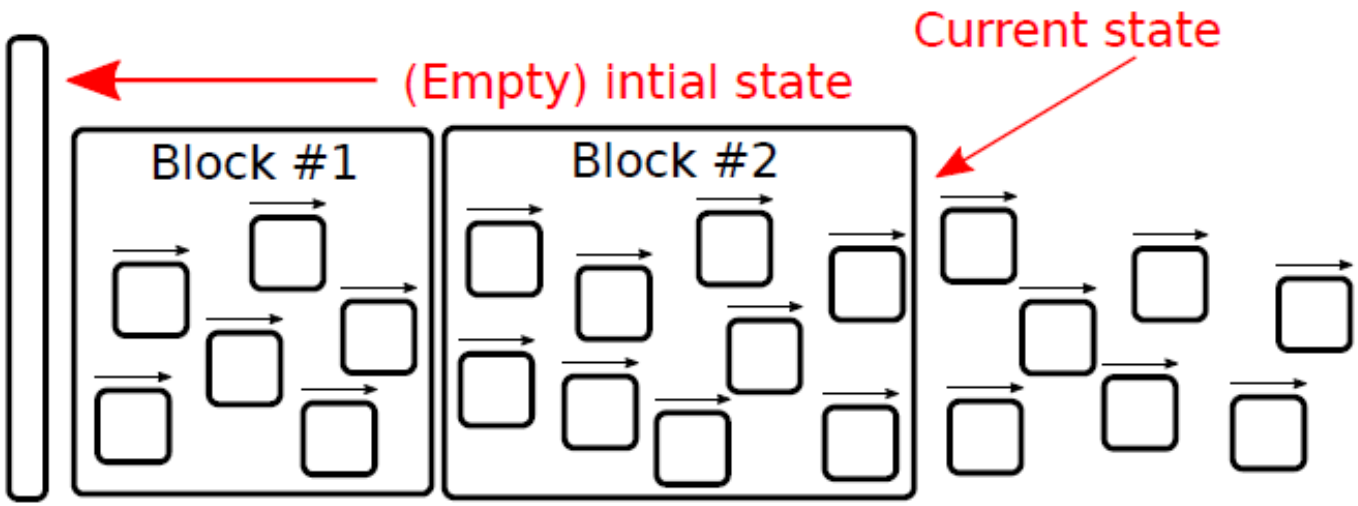


Blocks

- We group the transactions into blocks
- Blocks depend on one-another
- Blocks have to meet certain criteria
  - all transactions with 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
- Anyone can form a new block at any time

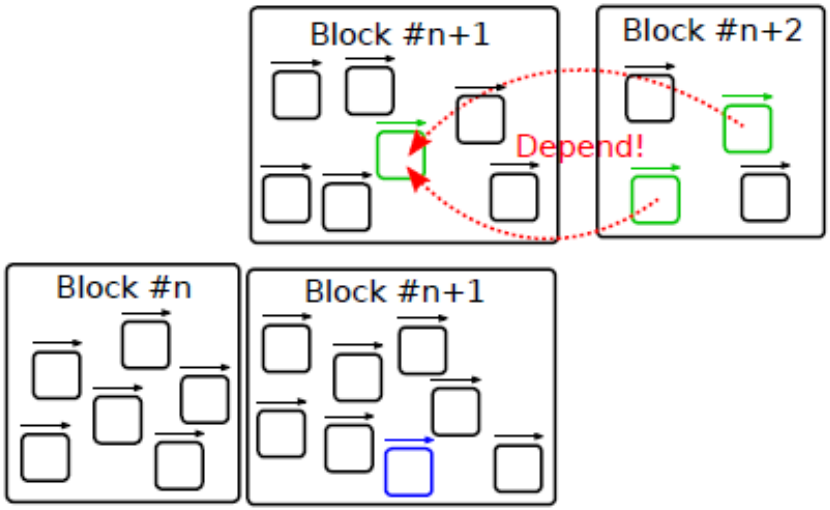
Blocks

- We define the current state as: all transactions within the longest branch of blocks applied to an empty state
- The state is at the end of the last block
- txns only become part of the state once they are inside a block



Blockchain

- If there are two blocks that both have the same number but different content, a problem arises
- Which of the two $n+1$ blocks the block $n+2$ depends on?

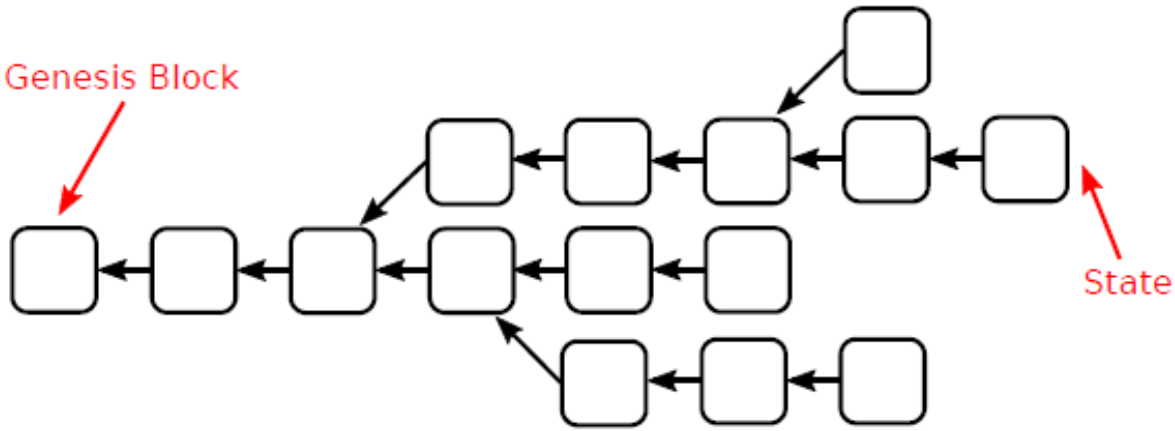


Blockchain

- As a solution, each block points to its previous block/references the previous block
- This reference is distinct and thus it is now possible to tell exactly what a block a block depends on
- Each block can only depend on exactly one prior block
- This is a bit like a linked list

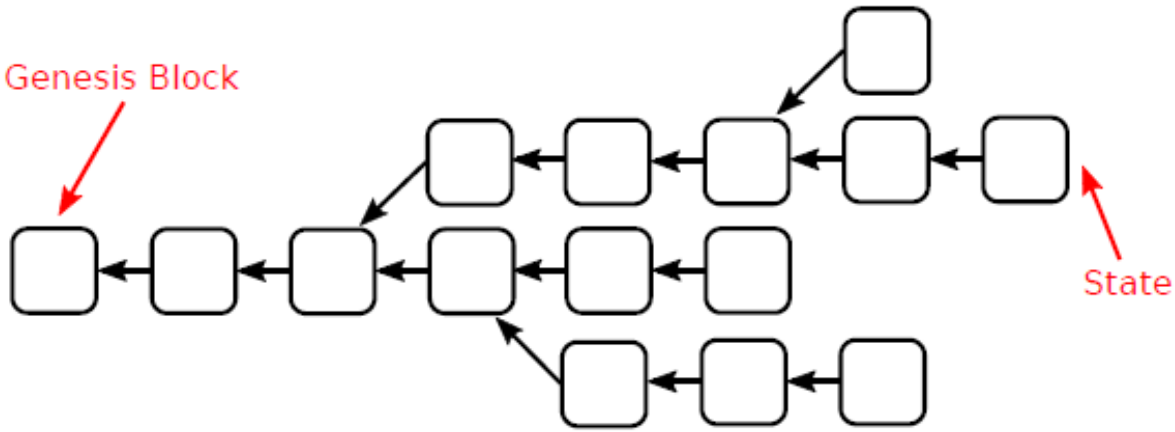
Longest branch (longest chain)

- The longest chain of blocks represents the current state
- If we display the chain as a tree, then that it is the longest branch
- The root of the tree is called the genesis block



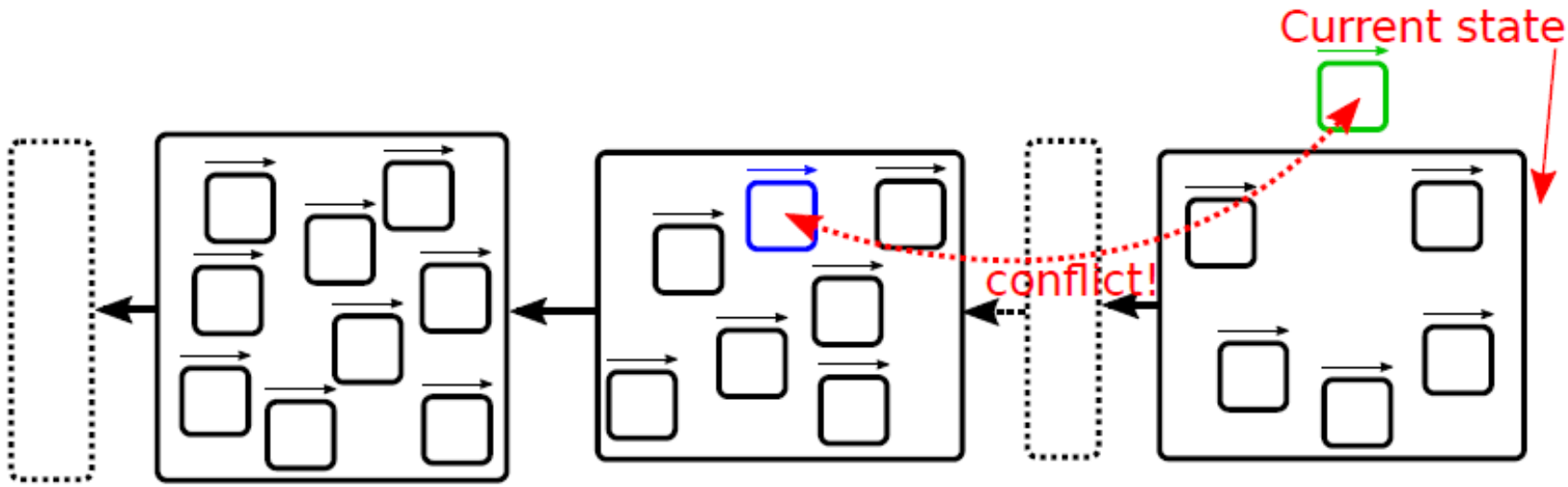
Longest branch (longest chain)

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



Solving double spend

- Recall txns within each block have to be compatible with txns in earlier blocks
- This is necessary, but not sufficient to solve the double spend problem



Solving double spend

- I make a transaction to draemmli, which is in conflict with the txn to 2_B I made earlier
- Because this new txn is in conflict with a txn already on the chain, it will not be included in the chain
- Since the txn never becomes part of the state, draemmli never receives the money and thus never gives me the computer

Solving double spend

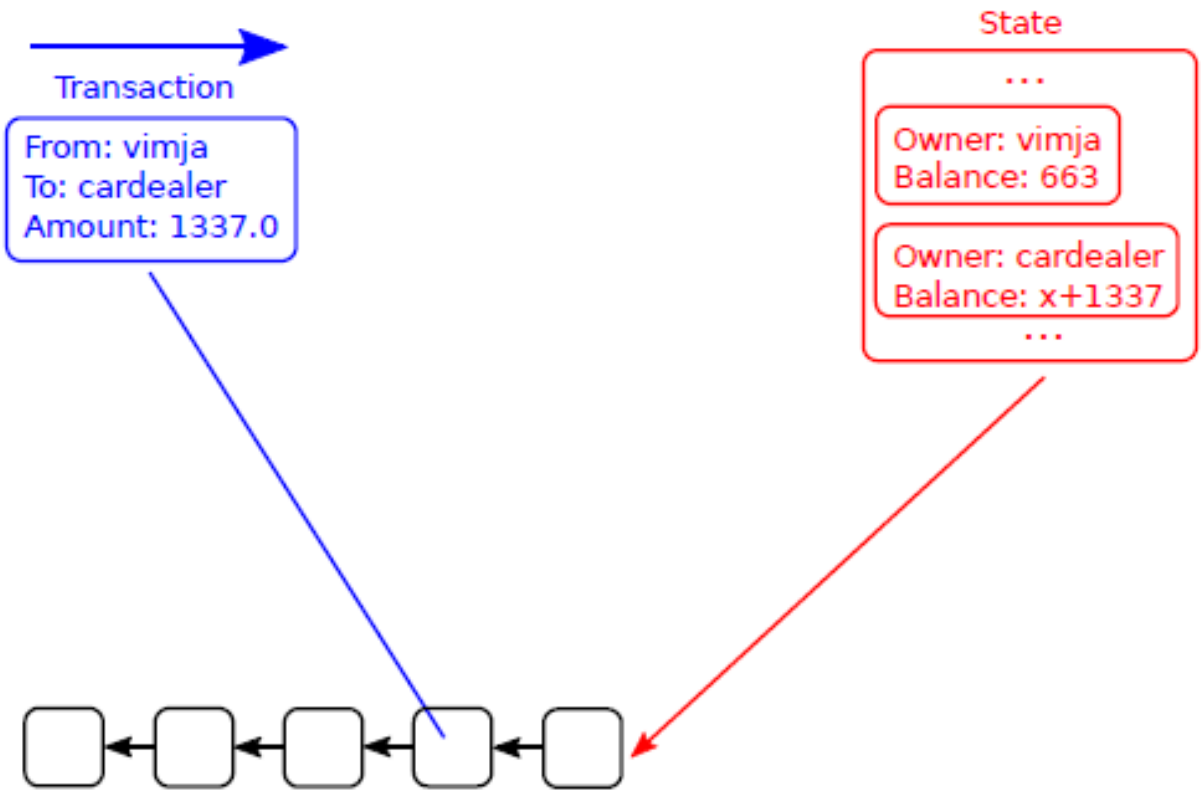
- I make a transaction to draemmli, which is in conflict with the txn to 2_B I made earlier
- Because this new txn is in conflict with a txn already on the chain, it will not be included in the chain
- Since the txn never becomes part of the state, draemmli never receives the money and thus never gives me the computer
- Unfortunately, this is not the end of the story

Solving double spend

- Double spend is still possible . . .
- The attacks are a little more difficult
- Lets steal a Tesla car this time

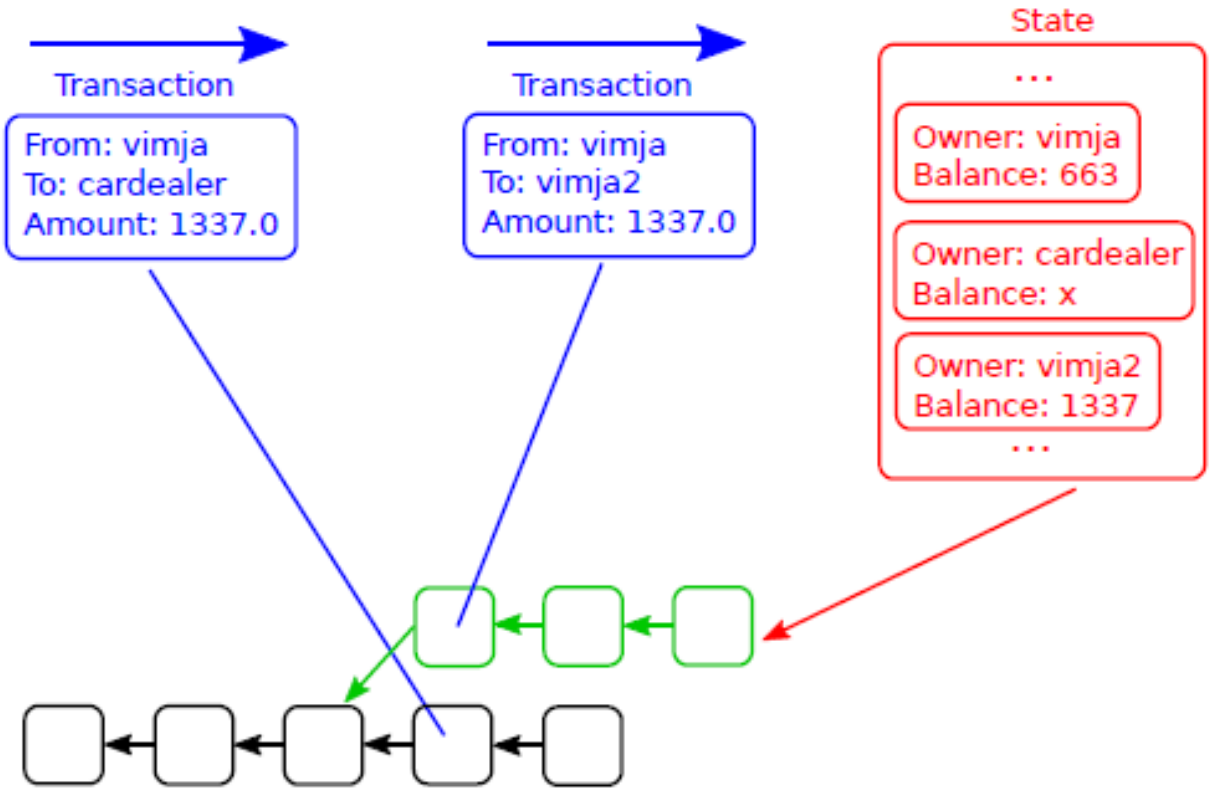
Classic double spend attack

- I create a txn, transfering the money to the car dealer
- After someone creates a block containing this txn, it becomes part of the state
- Potentially, further blocks are created



Classic double spend attack

- I create another txn, transfering the same money to my second account, put this in a new block, and add more blocks until my branch is longer than every other branch in the system
- The first txn can't become part of the same branch again . . .



p2p networking attack setup

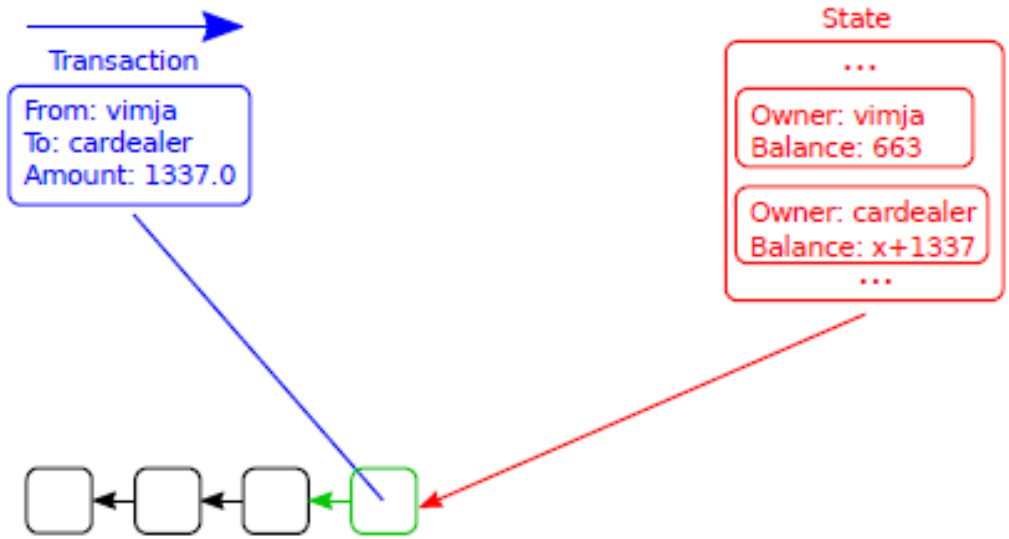
- Most blockchain based currencies use a p2p network
  - to distribute the transactions
  - to distribute the blocks

p2p networking attack setup

- Most blockchain based currencies use a p2p network
  - to distribute the transactions
  - to distribute the blocks
- An attacker p2p network attacker can control our view of the network
  - Present to us blocks and transactions he does not present to the rest of the network
  - Hide from us blocks and transactions the rest of the network sees

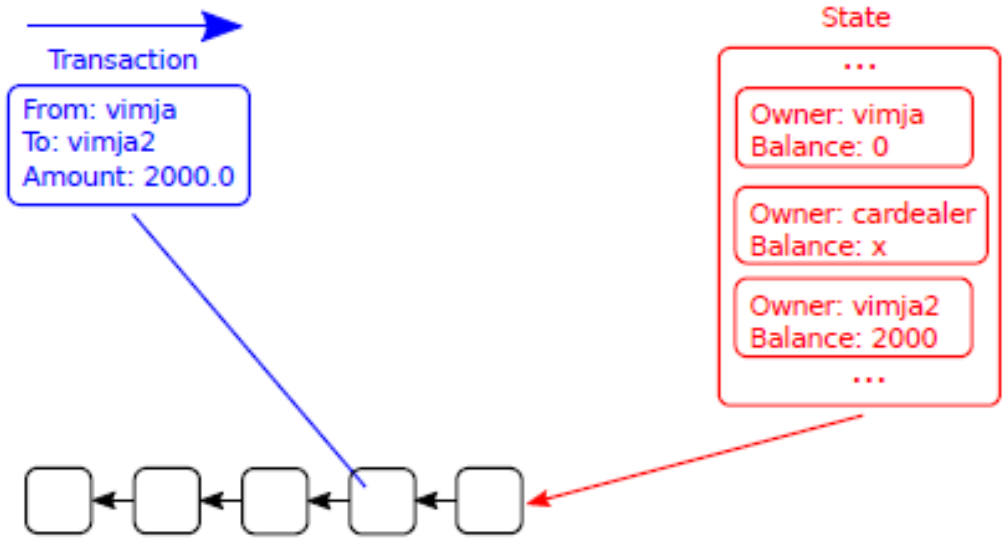
Double spend network attack

- I create a txn transferring the money to the car dealer
- Present this txn to the car dealer, but not to the network
- Create a block containing this txn, and present this block to the car dealer, but not to the network



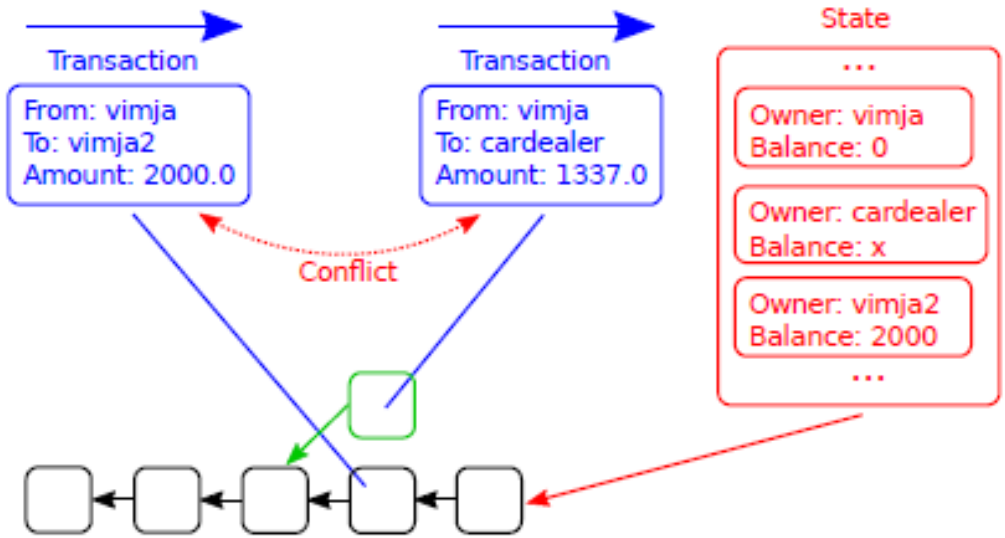
Double spend network attack

- I create a txn transferring the money to my second account
- Hide this txn from the car dealer
- Wait for this txn to be included in a block and wait for yet another block to be formed
- Hide both those blocks from the car dealer



Double spend network attack

- The txn to the car dealer is not part of the state



Double spend network attack

- Once I stop the attack, the car dealer becomes a part of the network again
- The car dealer and the rest of the network synchronize their blocks and transactions
- The car dealer's branch is the shorter one
- In the other branch, which is the state of the system, he has never received the money

Remaining problems

- Why are the attacks just described possible?
- Because it is easy and cheap to create blocks
- Anyone can create any number of blocks at no cost

Proof of Work (PoW)

- Solution: Make creating blocks hard
- To create a new block, one has to solve a computational problem
  - The solution gets included in the block
- The solution is called a Proof of Work

PoW

- We need to slightly adjust the rules for valid blocks
- They need to contain a valid solution to a problem
- This prevents someone from creating a new long chain of blocks quickly

PoW

- We need to slightly adjust the rules for valid blocks
- They need to contain a valid solution to a problem
- This prevents someone from creating a new long chain of blocks quickly
- We redefine the state
  - The branch with the highest accumulated difficulty

PoW

- The problem has to meet certain criteria
  - be hard! solvable only by brute force
  - be adaptable in difficulty (to the amount of power available to the network)
  - depend on the input (the block in question)
    - preven pre-compute attacks

Computationally Difficult Puzzle

- find a nonce such that hash(nonce || transaction data) is sufficiently close to 0
- By sufficiently close to 0 we mean that the leading k bits of hash(nonce || transaction data) should all be 0, where k is a parameter

Computationally Difficult Puzzle

- find a nonce such that hash(nonce || transaction data) is sufficiently close to 0
- By sufficiently close to 0 we mean that the leading k bits of hash(nonce || transaction data) should all be 0, where k is a parameter
- The bigger the choice of k, the harder the problem

Block reward

- There needs to be an incentive for creating new blocks
- We have no way of creating any new money yet
- Where does it come from? (we start with an empty state)

Block reward

- We solve both problems at once
- The miner of a block gets brand new money
- Done via a coinbase transaction
  - transfers money (called reward) from nowhere to the miner's account
- Additional incentive: transaction fees

Mining

- Which branch should a miner work on?
- For the reward to be spendable, it need to be part of the state
- This state is the longest branch
- So it only makes sense to work on the longest branch

Randomization

- If all miners are working on the exactly same problem, the one with the fastest computer would always find the solution first
- This is undesirable (why?)

Randomization

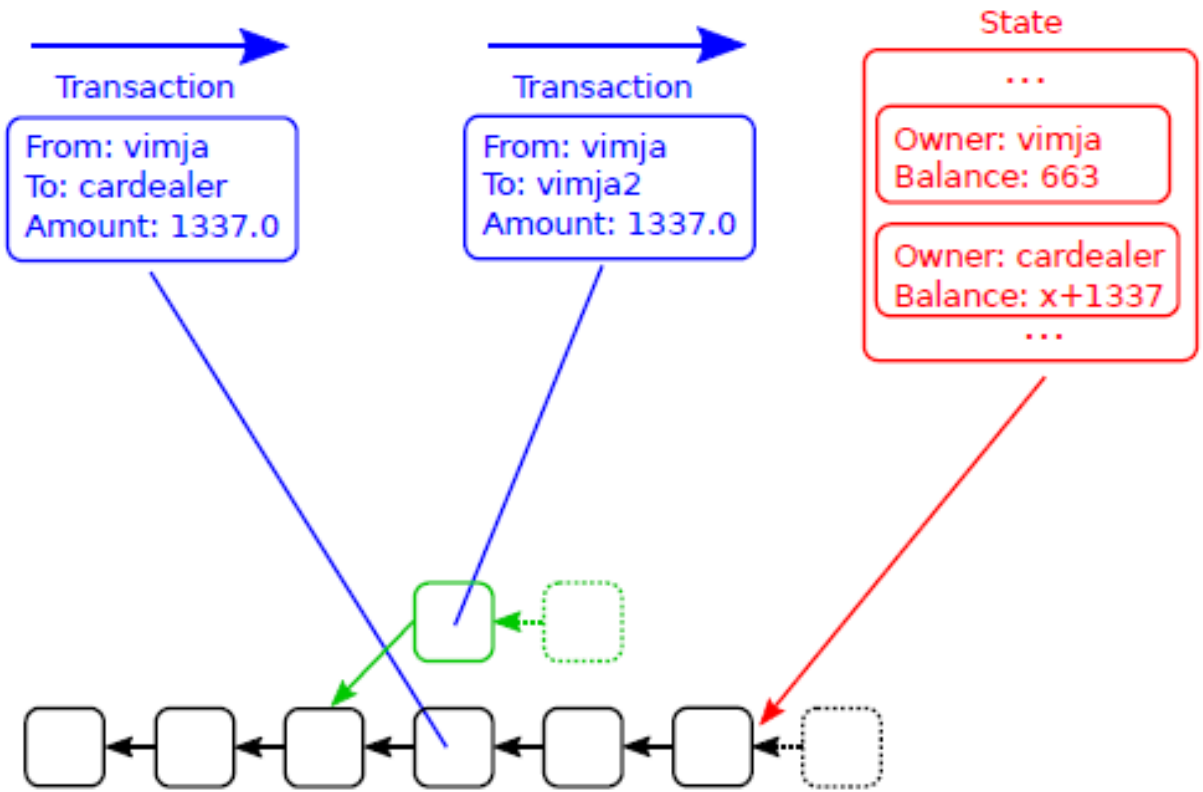
- In practice, the miners will not all be working on the same problem
  - coinbase transaction
  - different selection/order of transactions
- This equally distributes the success rate according to the actual computing power

Attacks revisited

- Lets revisit the double spend attacks from before
- Now we have PoW to protect from the attacks

Double spend attack

- Since the attacker's mining power is less than the combined power of all the other miners, the attacker creates new blocks at a slower rate
- Now the attacker can't easily create a longer branch
- So, the attacker's branch never becomes the state of the system

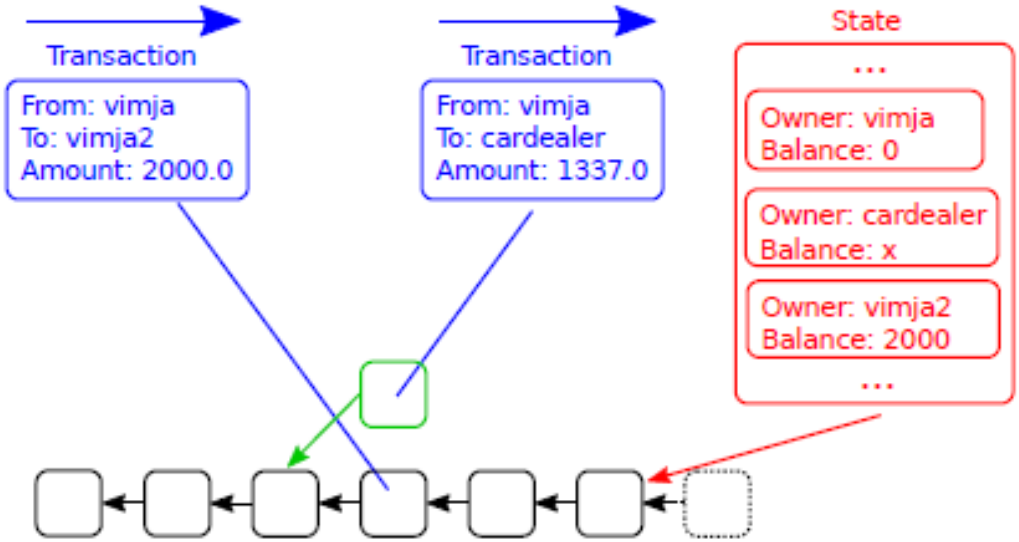


Double spend attack

- The only way attack could succeed is if the attacker had more mining power than all the other miners combined
- Thus the term 51% attack
- Such an attack breaks the system

Network attack

- Since the rest of the miners are faster at creating at new blocks than the attacker is, the branch on the public network is the longer one
- However, since the car dealer can't see the network's branch, he accepts the attacker's block as the longest chain



Network attack

- When the attack stops and the car dealer synchronizes with the rest of the network, it becomes clear that the car dealer's branch is the shorter one
- The attack still worked despite PoW
- However, for the attacker such an attack is not worth the effort

The cost of the attack

- To perform the attack the attacker has to create the block he presents to the car dealer
- During this time, the attacker can't use the computational power to mine on the main network
- So he has to do without the block rewards he could get during that time

Example

- Attacker power: $20\% = 1/5$ of the network
- Network avg. block time: 10 min
- Block reward: 25 Btc
- Time attacker needs to create one block $= 50$ min/block
- So the attacker spent 50 min on the attack
- Reward in 50 min mining $= 25$ btc

Example

- So the attacker has to decide: should he perform the attack, or should he earn 25 Btc
- Obviously, he will go for the more profitable choice
- Performing the attack is only worth it if attacker can steal more than 25 Btc in these 50 minutes