Research Group
Distributed Systems

Christian-Albrechts-Universität zu Kiel
Technische Fakultät

Blockchain & Bitcoin II

Olaf Landsiedel

Before we get started

- Happy New Year
  - And welcome back!

Course evaluations

- We sent out the TANs yesterday
  - To your email addresses registered in iLearn
  - Comment: Funny system
    - Not used to me sending TANs to you
  - Write me an email if you have not received one
- We really appreciate your feedback
- Help us to improve the course for next year's students

Labs

- Lab 4 deadline this week
  – How is it going?
- Project (=Lab 5)
  – How is it going?

Last time?

- Blockchain
- Bitcoin

BITCOIN

Central Principle

Give the good guys easy problems to solve…

(how should this sentence continue?)

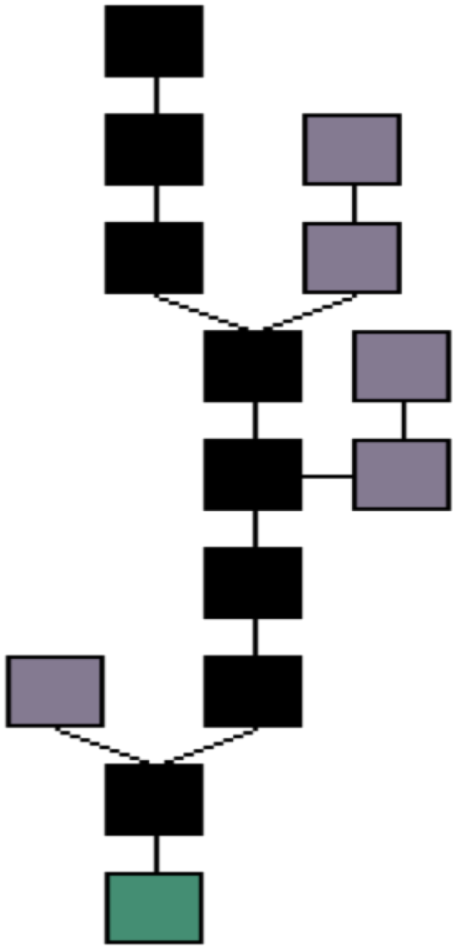
…and give the bad guys hard problems to solve.

Primitives

- SHA-256: hash
- Signatures: public / private key
- Tamper evident Merkle Trees
- Block chain
- Proof of Work

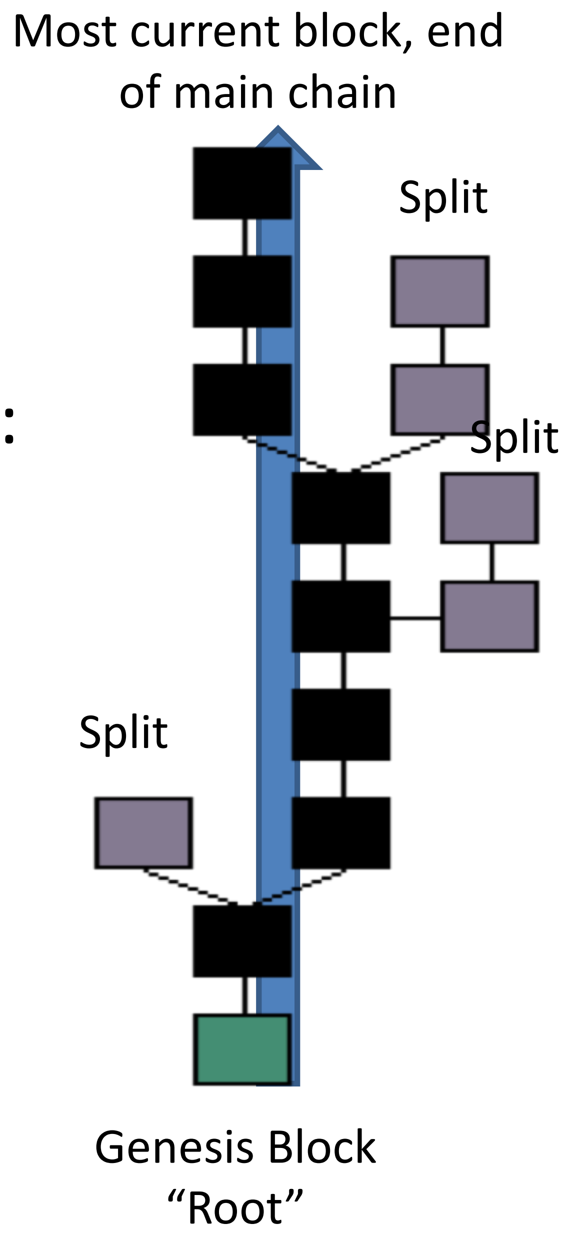
Blockchain: Concept I

- Blockchain: distributed ledger (=log)
  - time-stamped, non-repudiable (un-disputable) database
  - contains the entire history of transactions
  - Tamper proof via cryptographic primitives
- Each transaction processor
  - maintains own local copy of the blockchain
  - consensus algorithm:
    - every copy to stay in sync



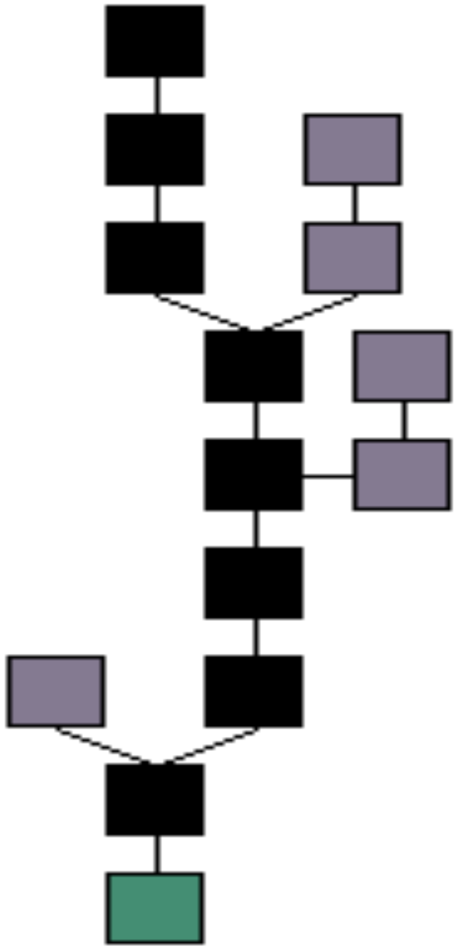
Blockchain: Concept II

- Each block
  - connected to previous one
- Each block has a hash of previous block:
  - -> blockchain
- Can/will get large
- Main chain (black)
  - Blocks from genesis block (green)
  - to the current block
- Can have splits
  - Dark grey blocks
  - Orphaned



Blockchain: Concept III

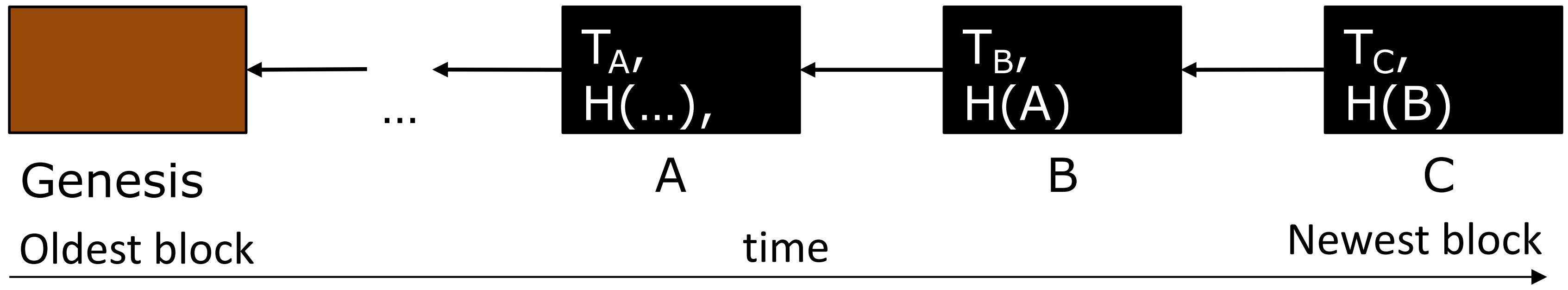
- Ensures complete transparency
  - Each transaction is visible to all
    - Participants use anonymous identifiers
- Prohibits double spending
  - No coin can be spend twice
    - As all transactions are known
    - Double spending can be detected
- Large data structure
  - Replicated on nodes in the network
  - Scaling: open research challenge



Block in the Blockchain

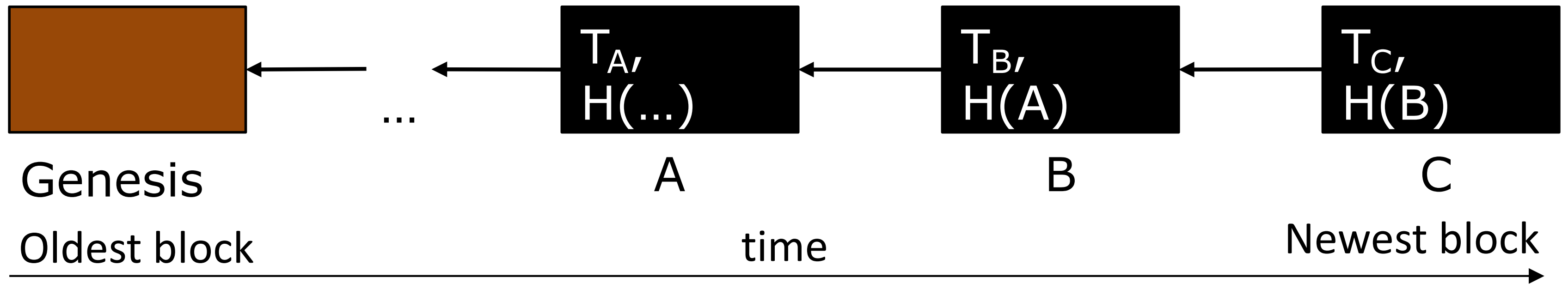
- A block is a container for a set of transactions
  - “A pays to B 10 crypto coins”$_{\text{signed by A}}$
  - “C pays to D 10 crypto coins”$_{\text{signed by C}}$
  - ….

Blockchain Details I



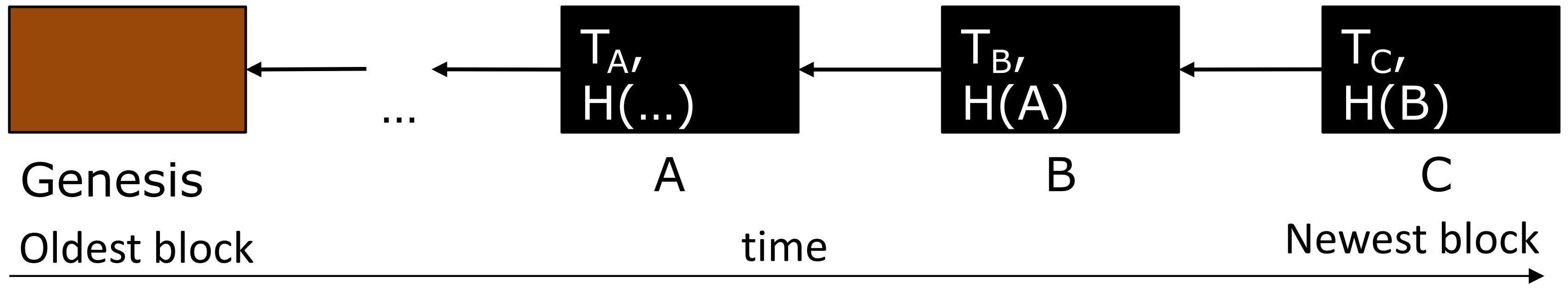
- Each block X contains
  - A set of transactions: T_X
  - Hash H_P of parent P (previous block => previous transactions)
  - And some meta data
- The genesis block has no parent
  - Bitcoin: It does have 50 bitcoins (initial money supply)
- Block identified by its hash or its height
  - Genesis has height 0

Blockchain Details II



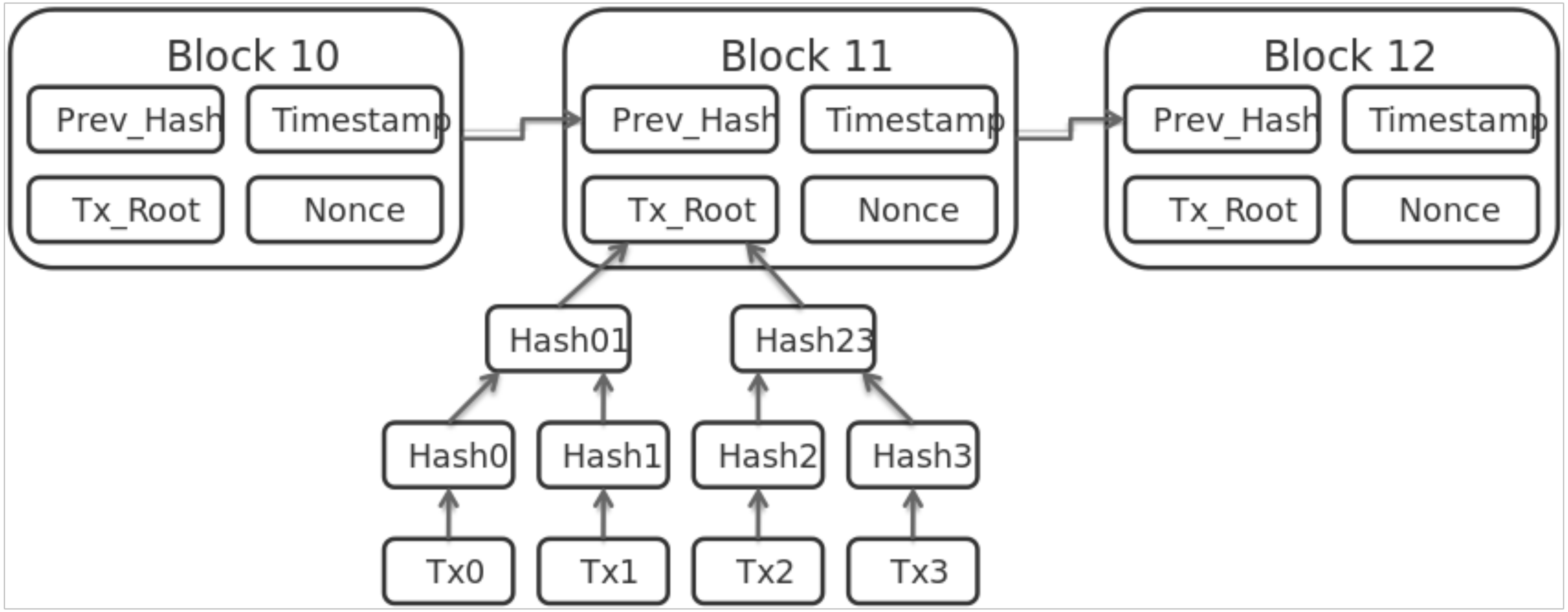
- Each block refers to its parent block:
  - Contains a hash H of the block’s parent
  - The parent block is the previous block added to the chain
  - Example: B’s parent is A
- Why is this tamper proof?
  - Change A forces a change to B and C
  - Proof of work needs to be recomputed too (later slides)
  - This makes later changes to transactions very hard

Blockchain Details III



- Transactions
  - A Merkle root summary of ~ 500 transactions

Blockchain Details IV



From Wikipedia

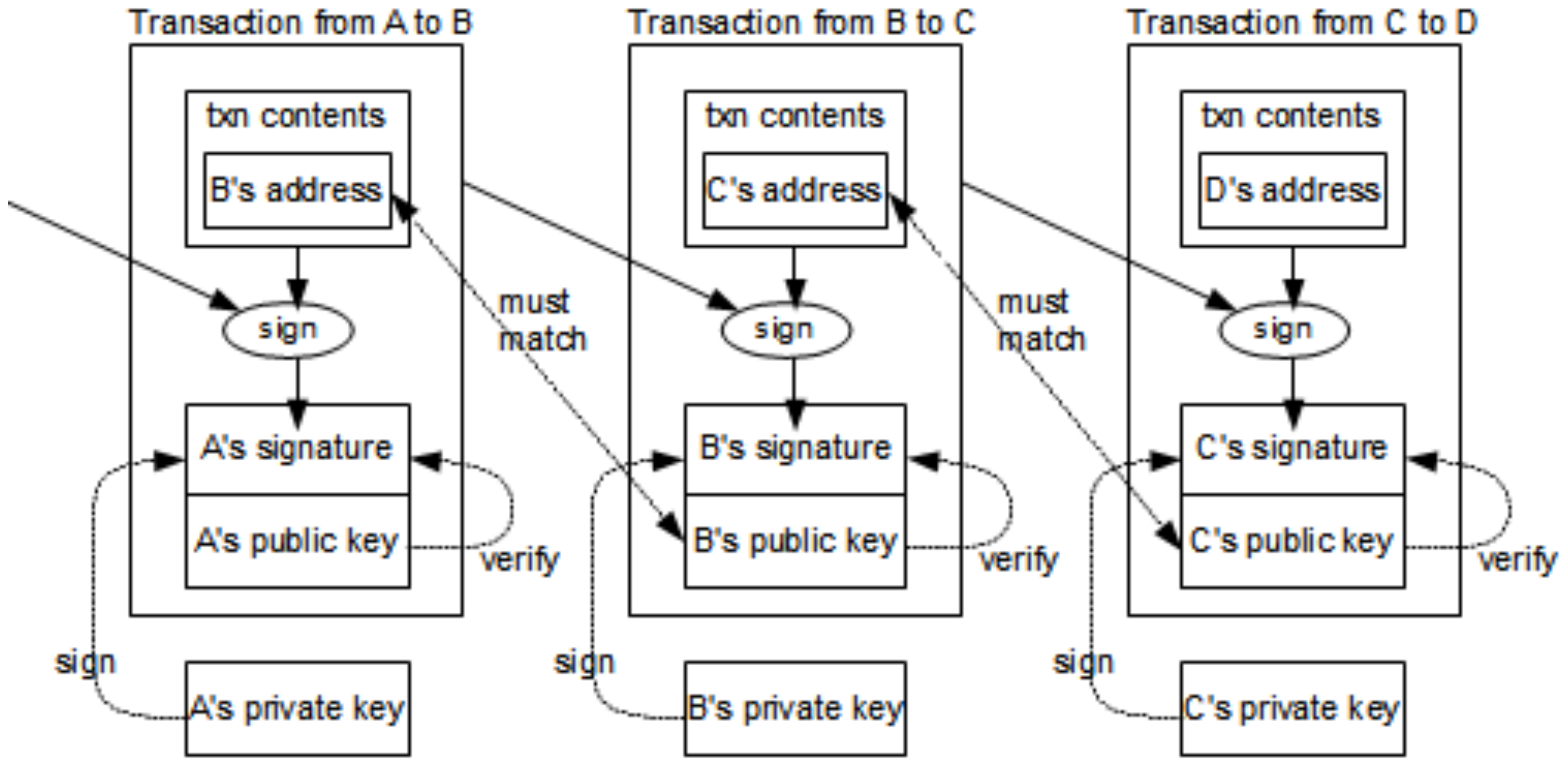
Transactions

- Many transactions live in the block’s Merkle tree.
  - You do not spend from an account. You spend from previous transactions.
- Transaction
  - Input address
    - points to the output of an earlier transaction
  - Output address
  - Signed by owner of funds

Transactions

- Transactions are between pseudonyms
  - hash of public keys
- Pseudonyms
  - Not anonymous
  - But also not connected to an identidy
  - You can reveal your public key to others
    - So that they can send money to you

Transactions



Until now

- Motivation
- Crypto Primitives
- Block chain
- Transactions

- Next
  - Consensus
  - Proof of Work

CONSENSUS

Transactions Version 1

- Alice signs a message and sends it to Bob
  - `A -> B      {A gives B 1 BC}`K_{APriv}
  - Private key of Alice: K_{APriv}
- Positive:
  - Alice shows intent
  - Alice cannot send a payment signed by Carol. Alice does not know Carol's private key K_{CPriv}
- Problem:
  - Not safe against replays from Alice: Does Bob now have several coins or 1?
  - Bob could make 5 copies. How many coins does he have now?

Transactions Version 2

- Alice signs a message and a unique serial number
  - Alice sends it to Bob
  - `A -> B     {A gives B 1 BC, 123990245}`K_{APriv}
- Positive:
  - Alice shows intent
  - Replay not of concern with Bob
- Problem:
  - How does he know she has the money to spend?
  - She can double spend with Charlie
    - `A -> C     {A gives C 1 BC, 123990245}`K_{APriv}
  - Where do we get all these unique serial numbers?

Transactions V2 With a Bank

- Alice gets a unique serial number from the bank
  - Alice signs a message and the serial number and sends it to Bob
  - `A -> B     {A gives B 1 BC, 123990245}`K_{APriv}
- The bank maintains a ledger of who owns what
  - Bob checks with the bank before providing service to Alice
  - Does Alice have this to spend?
- Positive:
  - No replays – everyone checks with the bank before accepting
  - No double spending: Transaction to C would fail verification with bank
    - `A -> C     {A gives C 1 BC, 123990245}`K_{APriv}
- Problem:
  - Centralized model

Transaction Version 3 No bank – the community keeps track

- Alice signs a message with a unique serial number
  - and sends it to Bob
  - `A -> B     {A gives B 1 BC, 123990245}`K_{APriv}
- Bob verifies – he checks his copy of the ledger
  - Does Alice own the bitcoin?
  - Bob accepts and Bob tells everyone about the transaction
  - Everyone updates their own copy of the ledger
- Positive
  - Decentralized model
- Problem
  - Alice may double spend with Charlie
  - How do the others fix and keep up-to-date their ledgers?

Version 4 Everyone Verifies

- Alice signs a message with a unique serial number
  - and sends it to Bob
  - `A -> B     {A gives B 1 BC, 123990245}`K_{APriv}
- Bob asks everyone to verify
  - If enough verifications
    - Bob provides service
    - and asks everyone to update their copy of the ledger
- Positive
  - Double spending will normally be detected
  - Decentralized model
- Problem
  - Sybil Attack possible and allows the double spend.
  - How: Alice starts many entities, each can cast one vote
    - Alice votes: sure, money has not been spend
      - But already spent it

Detour Sybil attacks

- Scenario
  - Collect votes from all participants
  - As in Bitcoin
- Sybil attacks
  - What if one entities pretends to be thousands of entities
    - And thereby gets thousands of votes?
- Problem
  - Sybil Attack possible and allows the double spend.
  - How: Alice starts many entities, each can cast one vote
    - Alice votes: sure, money has not been spend
      - But already spent it

Version 4 Everyone Verifies

- Goal: (Practically) nobody can rewrite history
  - Example:
    - In blockchain: the longest chain wins
    - Attack: to secretly maintain a second chain that is longer than the first one and use this for double spending
- Just a signed transactions is not enough
  - Signed with the private key
  - I can still create multiple ones
    - And nobody knowns who I am, so they cannot “punish me”
    - Or stop accepting transaction by me -> just create another entitiy

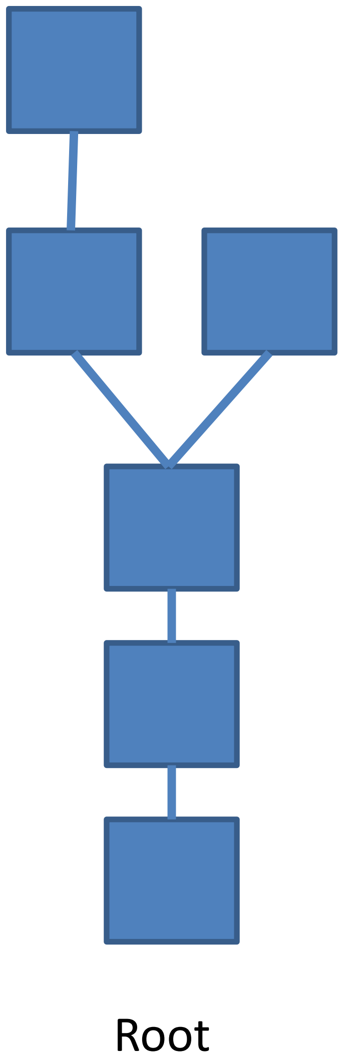
PROOF OF WORK

Proof of Work

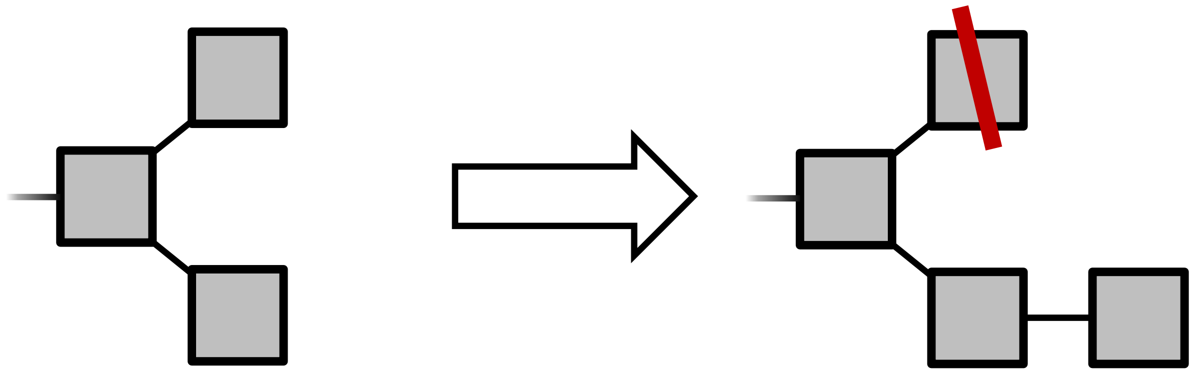
- Goal:
  - Avoid Sybil attacks
  - Verify transactions of others
  - Identify double spending
  - Make changes to blockchain (practically) infeasible

Consensus

- Do some transactions
  - Make a block, link to previous block
  - Share in the network
- Others do the same
  - Also link to previous block
- -> “Competing” blocks
  - Can only continue on one chain
    - Majority wins (and the larger one wins)

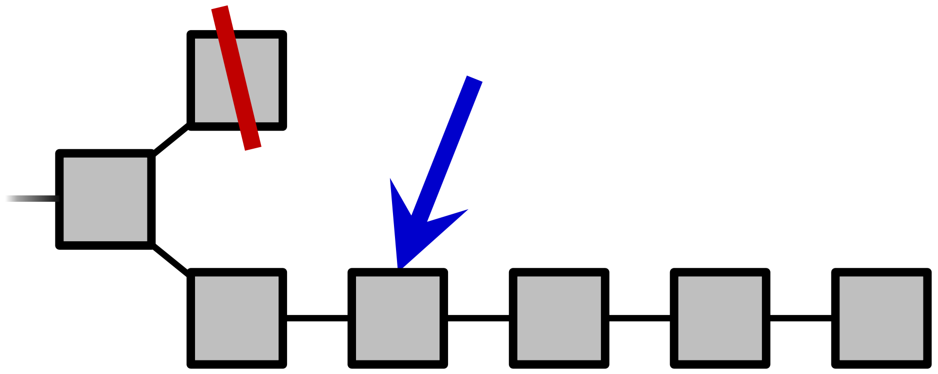


Fork Resolution



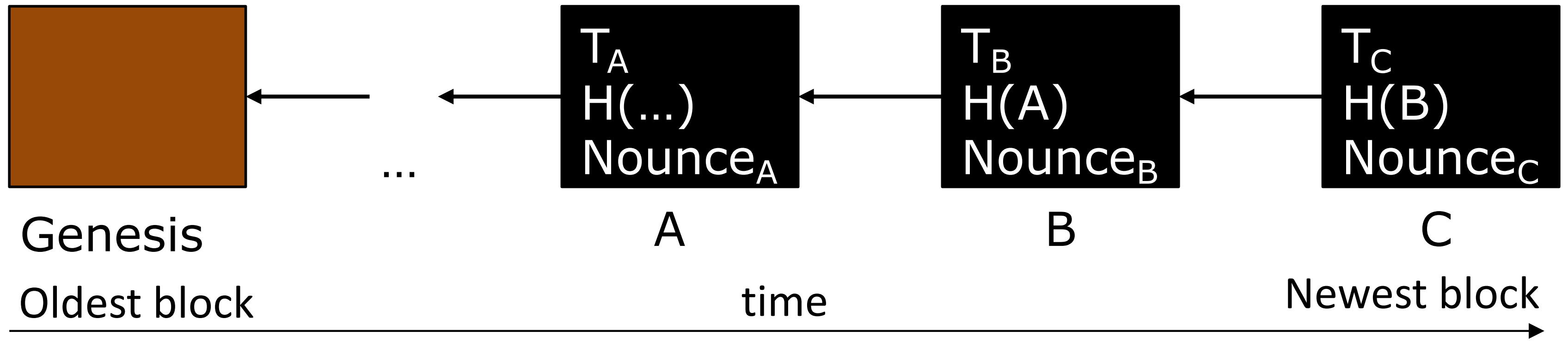
- **Longest** chain wins
- Transactions are reverted
- Double-spending a threat

Fork Resolution



A transaction is **confirmed** when it is **buried** deep enough

Proof of Work



- Select Random Nounces until
  - Hash of block is less than target
  - Target: value known to all participants
  - Example:
    - Find Hash less 9999999 (easier)
    - Find Hash less 99 (harder)

Proof Of Work: Idea

- Consider a difficulty of 2:

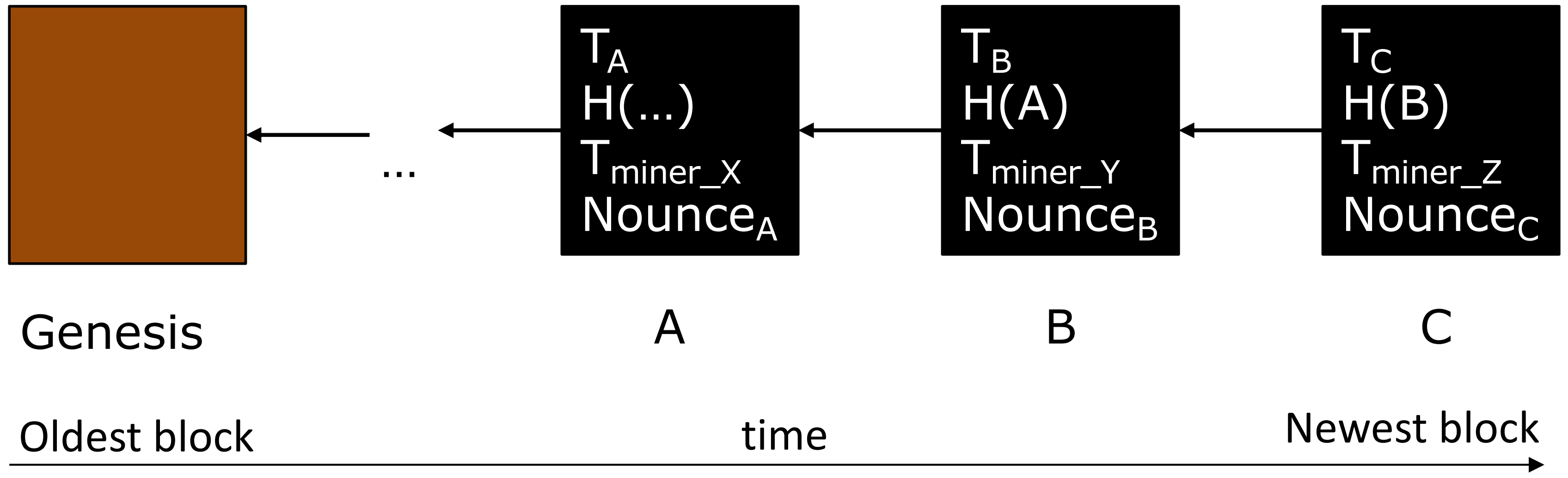
```
int i = 0;
h = SHA-256(someData + i);
while(two leftmost hex digits of h are not 00) {
    i = i + 1;
    h = SHA-256(someData + i);
}
```

- How any times will this loop execute?
  - Prob(”00”) = 1/16 * 1/16 = 1/256. 256 trials expected.
  - Hard to compute prooof of work but easy to verify.
- The difficulty can change. We can force this to take 2 weeks on average.
  - If computers become faster we can increase the difficulty

Version 5 Everyone Verifies with POW

- Introduce verifiers
- Alice signs a message and a serial number and sends it to everyone
  - `A -> All     {A gives B 1 BC, 123990245}`K_{APriv}
  - The transactions are collected by each verifier
  - Verifiers maintain a queue of pending transactions.
- To verify:
  - (1) Check ledger for double spend
  - (2) Perform POW (solve a puzzle)
  - (3) Announce block is verified
- The first to do so, gets a share in the transaction
  - It is easy for others to check this result against the blockchain and check the POW.
  - Alice will likely not be the first to verify.
  - POW makes a Sybil attack very expensive.

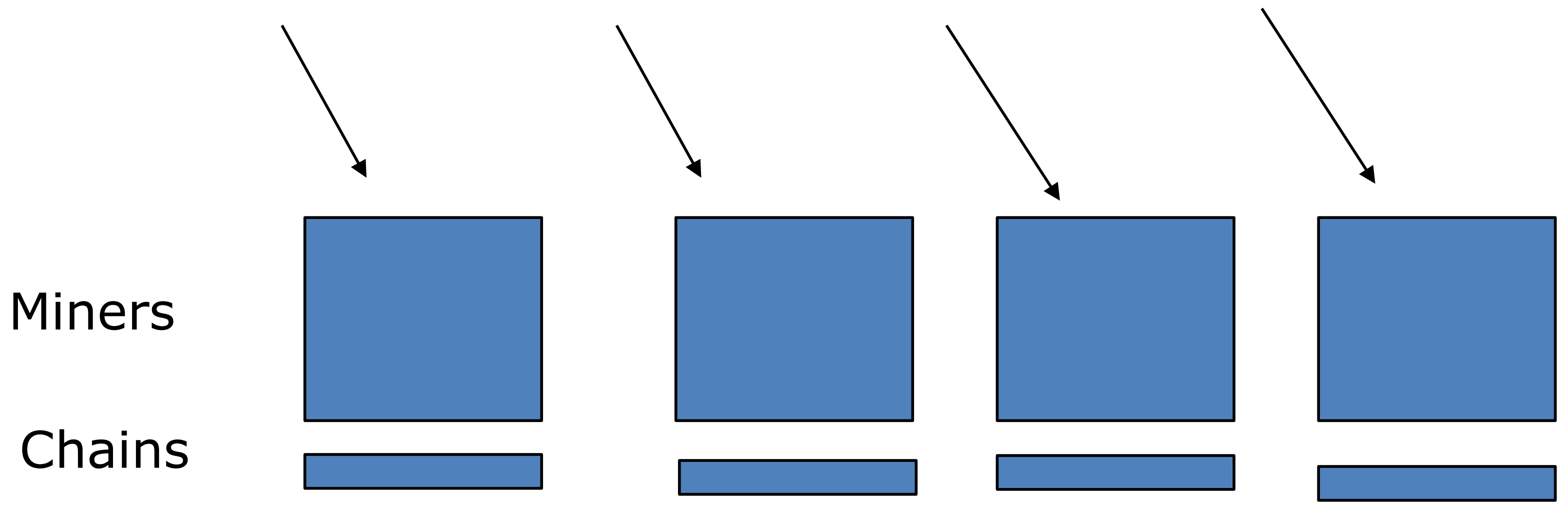
How are the miners paid?



- Transaction fee
- Stored in the block
  - Transaction to the miner itself

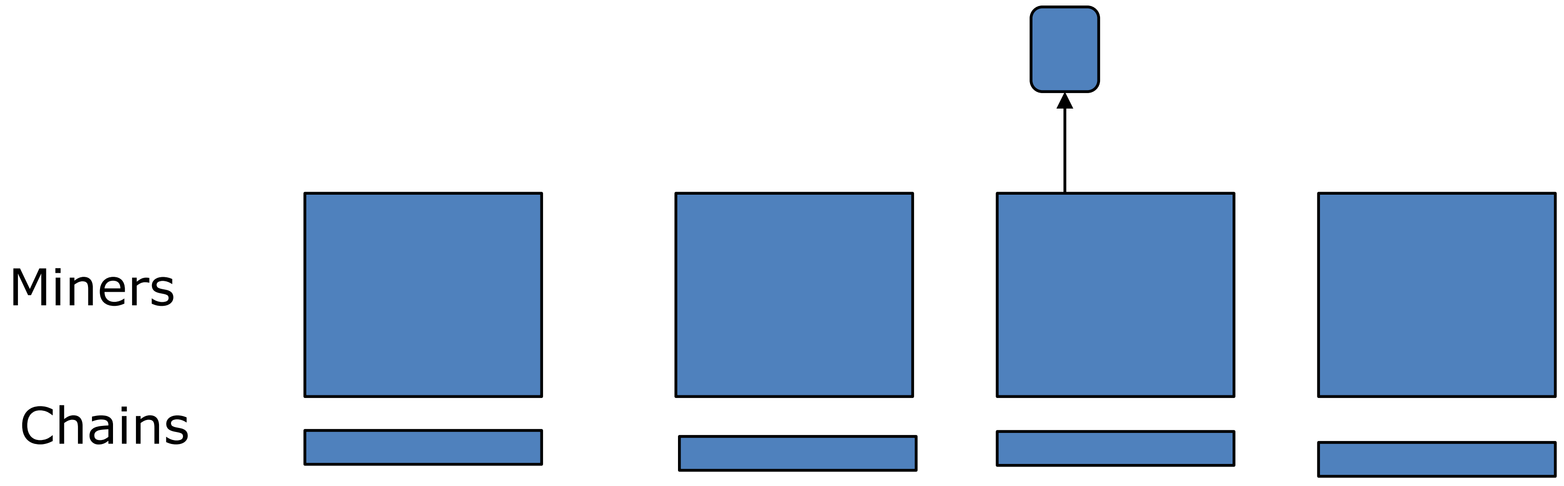
Try to double spend

- A -> All {A gives B 1 BC, 123990245}K_{APriv}
- A -> All {A gives C 1 BC, 123990245}K_{APriv}



Suppose both transactions are placed on the peer to peer network.
Miners verify each transaction and then collect the pending transactions into blocks.
An honest miner will not include inconsistent payments in their block.
-> One of these double spend transactions will be rejected.

Try to double spend



After about 10 minutes, some miner produces a block. This miner won the race.
Other miners receive the new block and verify transactions and the proof of work.
It is easy to verify the proof of work and all transactions in the block.
If the block passes all checks, it is added to the blockchain.
Unless she controls a lot of nodes, Alice cannot predict the winner
of the race to solve POW.

Fraud prevention

- Users can trust the block chain that was most difficult to produce
  - longest chain wins
- If there was a "fake" blockchain competing with the real ones the fraudster would have to do as much work as the rest of the network to make their block chain look as trustworthy
  - intense work that goes into finding blocks through hashing secures the network against fraud

Proof of Work

- Problems
  - Energy intensive
  - Sensitive threshold
    - One participant with more than 50% of the computer power
  - Only probabilistic protection
    - With sufficient resources a participant could buy sufficient computer power to cross 50% threshold

Bitcoin Security

- An attacker with > 50% of hash power can
  - Double spend: Reverse transactions that he sends while he's in control
  - Prevent some or all transactions from gaining any confirmations
  - Prevent some or all other generators from getting any generations

Task of Bitcoin Miners(1)

- Listen for transactions
- Validate transactions – signature is correct and no double spending
- Maintain blockchain and listen for new blocks
- Validate each block that you receive by validating each transaction in the block and checking that the block has a valid nonce.

Task of Bitcoin Miners(2)

- Assemble a candidate block from validated transactions.
- The candidate block will extend the longest known valid chain.
- Find a nonce for your block that makes it valid.
- Publish your block and hope it gets accepted
- Profit if accepted

Task of Bitcoin Miners(3)

- If two different blocks are mined and announced at around the same time, it results in a two block fork.
- So, which to build on?
- The default is to build on the block you heard about first.

New blocks

- How difficult should we set the proof of work
  - To ensure a proper creation rate of blocks
  - What is a proper creation rate?

Block creation rate

- Too low
  - People have to wait too ling for the transactions to be included into the blockchain
- Too high
  - Too many blocks created in parallel
    - Unsure which miner won
- Bitcoin
  - Adjust difficulty to have one block roughly every 10 minutes

Block Chain

- Bitcoin makes sure there is only one block chain by making blocks really hard to produce.
- miners have to compute a cryptographic hash of the block that meets certain criteria
  - difficulty of the criteria for the hash is adjusted based on how frequently blocks are appearing
  - also carefully validate all the transactions that go into their blocks
- Successful miners are rewarded some bitcoins according to a preset schedule

Bitcoin

- Governance - an open source community of developers backed by the Bitcoin Foundation.
- Democratic - if you don't like one of the changes, you are more than welcome to fork the chain and implement your own rules
- Money Creation - is given to the people, not to the central bankers.
- Deflationary by design - money supply cannot be manipulated and is fixed at 21 million coins, each divisible up to 8 decimal

Bitcoin Mining

- Dedicated HW: ASICS etc.
- Serious energy consumption

Bitcoin Mining

Bitcoin Mining

Where do people Bitcoin Mining

- Where energy is cheap
  - Mining one Bitcoin:
    - $3,000 in China,
    - $500 in Venezuela,
    - $4,700 in USA (2018)
    - https://www.trustnodes.com/2018/04/26/bitcoin-mining-costs-just-3000-china-500-venezuela-4700-usa
- But cheap energy is not always clean energy
- Ethical considerations?

Your turn!

- Where do bitcoins come from?
- How often, on average, can we expect a new block be found by miners?
- Why is the difficulty of the proof of work in Bitcoin set to X minutes? What would go wrong if it was changed to Y seconds?
- What is Bitcoin Pizza Day, May 22nd?

https://olafland.survey.fm/blockchain-ii

Next Time

- Smart Contracts

- Questions?

- Inspired from / based on slides from
  – Jason Madden, Mehmet H. Gunes
  – Johannes Kofler
  – Terence Spies
  – And many others