Block Chain, Bitcoin & Security

…But truly Bitcoin, Proof of Work and Smart Contracts Development

EPITA - 2021

Part 1: course organisation

EPITA - 2021

About me

Thanh-Quy Nguyen

- Entrepreneur, freelance
- Double Master’s degree in fundamental physics
- École 42
- JavaScript developer
- Blockchain enthusiast (trading, smart contract development)

About me

Thanh-Quy Nguyen

- Not a security expert
- Not a DevOps

About me

Thanh-Quy Nguyen

- LinkedIn: https://www.linkedin.com/in/thanhquy/
- GitHub: https://github.com/tnguyen42
- Email: thanh-quy.nguyen@epita.fr

How about you?

A few questions

- What’s your name?
- What’s your background? (developer, DevOps, other)
- What do you want to do later?
- What experience do you have with blockchain? (development, investments/trading)
- What do you expect from this course?

Objectives of this course

15 hours class - ~6 hours homework

- Understand the blockchain technology (3 hours)
- Get a practical understanding of smart contract developments on Ethereum (10 hours)
- Get glimpse of the state-of-the-art in blockchain technology (2 hours on your presentations)
- Turn you into blockchain enthusiasts

What won’t be covered

- Development using Vyper
- Development using Remix
- Security audit process
- Blockchain development from scratch
- Trading

Grading

What to focus on

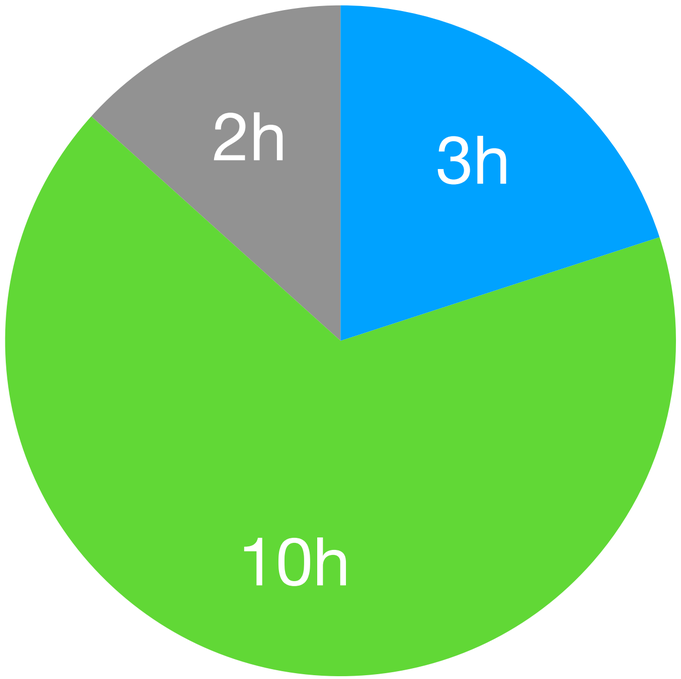
- Quiz: 20% (2 hours of homework)

- 1 (or 2) of these group project (80%):
  - Code project: 80% (~ 4 hours of homework), 3 students per group
  - Group Research and Presentation: 80% (~ 4 hours of homework), 2 students per group
- If you do the 2 group projects: the group project grade will equal BestGrade + 1/2 * OtherGrade
  - Example: if you get 15 and 12, the group project grade should equal 15 + 6 = 21 which will be capped at 20

Grading

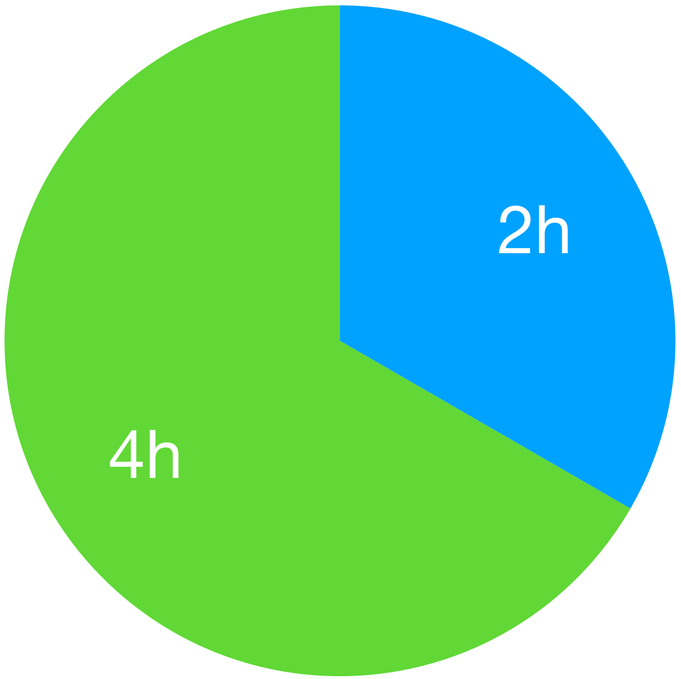
Sum up

Classes



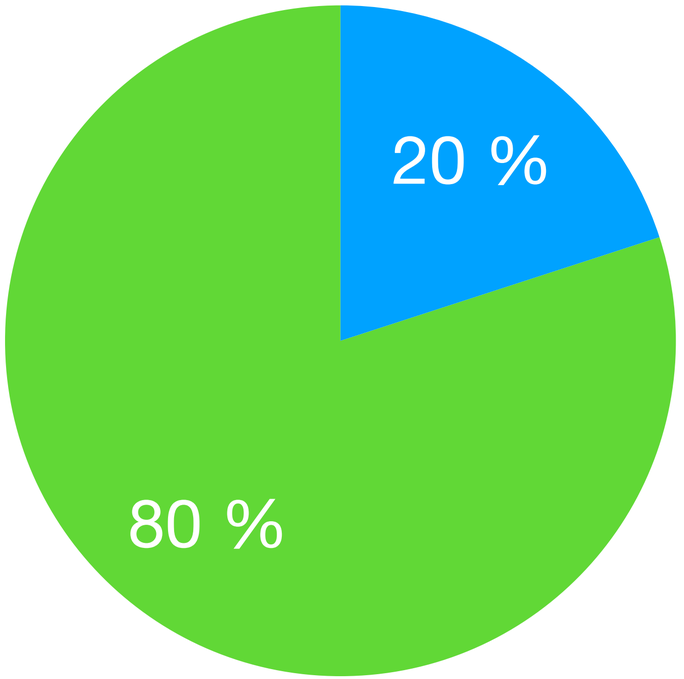
- Blockchain technology
- Smart contract coding
- Presentations

Homework



- Blockchain technology
- Group Project

Grading



- Blockchain technology
- Smart contract coding

Grading

Quiz

- Date: 03/05/2022
- Duration: 30 min
- Topics: Blockchain, Bitcoin, Ethereum
- Goal: not to be hard, just to give solid foundations for the following classes
- Ratio on the total grade: 20%

Grading: Project 1

Code project

- Due date: 02/06/2022, 23h42
- Topic: Develop a mini-Twitter on Ethereum
- Goal: to be able to develop an entire (smart contract) project on Ethereum
- Group size: 3 students (mixed profiles recommended for software development)
- Ratio on the total grade: 80%
- Grading: on the number of features and tests achieved + bonus points on the front-end
- Notes: you can start whenever you want if you want a head start - you can work with other groups

Grading: Project 2

Research and Oral Presentation + Peer Review

- 3 parts: 3 page Report + Presentation
- Goal: to (superficially) discover the state of the art; to learn how to synthesise complex topics; to learn from others' presentations
- Topic: of you choosing among a list
- Group size: 2 students
- Ratio on the total grade: 80%

Grading: Project 2

Report

- Report due date: 17/05/2022
- Presentation date: 19/05/2022
- Grading: 10 points on the report, 10 points on the presentation

Grading: Project 2

Oral Presentation

- Presentation date: 19/05/2022
- Duration: 20 min Presentation + up to 10 min of questions
- Note: synthesising is important
- Notes: camera and slides are optional, but think about what would be best for your audience

Block Chain, Bitcoin & Security

…But truly Bitcoin, Proof of Work and Smart Contracts Development

EPITA - 2021

Part 2: Bitcoin

1. Introduction
2. Block Hash
3. Proof of Work
4. P2P Network

Part 2: Bitcoin

1. Introduction
2. Block Hash
3. Proof of Work
4. P2P Network

Bitcoin origins

https://bitcoin.org/bitcoin.pdf

- Followed the financial crisis of 2008
- August 2008: bitcoin.org was registered
- October 2008: initial white paper released by “Satoshi Nakamoto”
- 3 January 2009: Genesis block
- 12 January 2009: first Bitcoin transaction

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must

Issues Bitcoin addresses

Issues

- Current finance is very opaque (high transaction fees, “There is no magic money”)
- Current finance relies on centralised authorities (governments, banks)
- Financial data belongs to those centralised authorities
- Those centralised authorities are often related, resulting in a domino effect

Solutions

- Bitcoin provides a public ledger and value is (mostly) defined by supply and demand
- Bitcoin relies on a peer-to-peer network
- Bitcoin is **pseudonymous**
- Bitcoin only relies on its network

What is Bitcoin?

A peer-to-peer network AND a currency

- It is literally a **block chain database**: each block contains **transaction data** resulting in hashes, and those hashes identify each block individually - a reference of the previous block's hash is contained in the following block.
- The block chain provides Bitcoin's public ledger: an ordered and timestamped record of transactions. The system is used to protect against double spending and modification of previous transaction records, making it an ideal **currency**.
- It is a **distributed database**: anyone can open a new node (a server) and have a (partial or total) copy of the blockchain.
- It is a **network**: each transaction is verified and propagated through the network through a set of rules, and when completely validated, the transaction is confirmed.

A few resources

- A video explaining the superficial concepts of blockchain: https://www.youtube.com/watch?v=SSo_EIwHSd4
- The official white paper: https://bitcoin.org/bitcoin.pdf
- The official technical documentation of Bitcoin: https://developer.bitcoin.org/devguide/index.html
- Bitcoin wiki: https://en.bitcoin.it/

Vocabulary

- Ledger: an ordered and timestamped record of transactions
- A node: a server maintained by a peer
- Consensus: when several nodes all have the same blocks on their block chain
- Consensus rules: the validation rules the nodes follow to maintain consensus
- Mining: the process of working to build new blocks. You provide computation power, you help secure the network, and you get rewarded in Bitcoins + transaction fees
- A satoshi: the smallest unit that is recorded on the Bitcoin blockchain.
  1 sat = 1.0 * 10 ^ (-8) btc
- A wallet: a public key + a private key. The public key is an “address” at which you can send bitcoin, the private key is a “password” allowing you to sign (and therefore send) transactions

Financial value

A totally optional slide

- One of bitcoin goals according to its community is to make it a safe haven.
- The safe haven status will only be achieved if it becomes popular enough.
- Contracts can be used on the Bitcoin blockchain, automating and securing some transactions (escrow, arbitration, micropayment channels…).
- Nowadays, Bitcoin value is mostly defined by supply vs demand on public exchanges: Coinbase, Binance being the most popular centralised exchanges. There are also “Over the Counter” exchanges for huge private transactions.
- Other cryptocurrencies can be traded on exchanges as well. Notably, Ethereum (another blockchain) enables decentralised exchanges to exist.

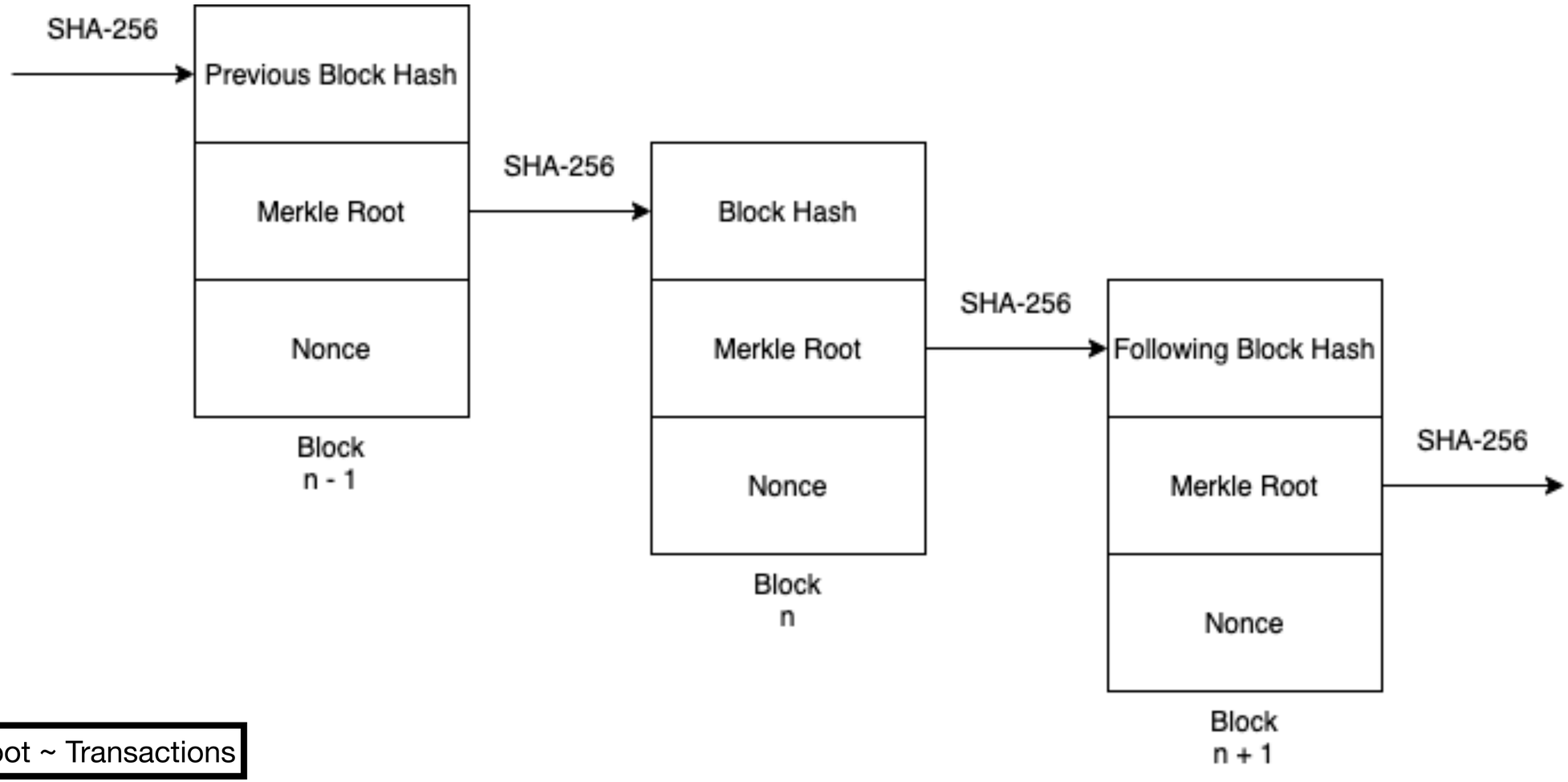
Part 2: Bitcoin

1. Introduction
2. **Block Hash**
3. Proof of Work
4. P2P Network

Block Hash

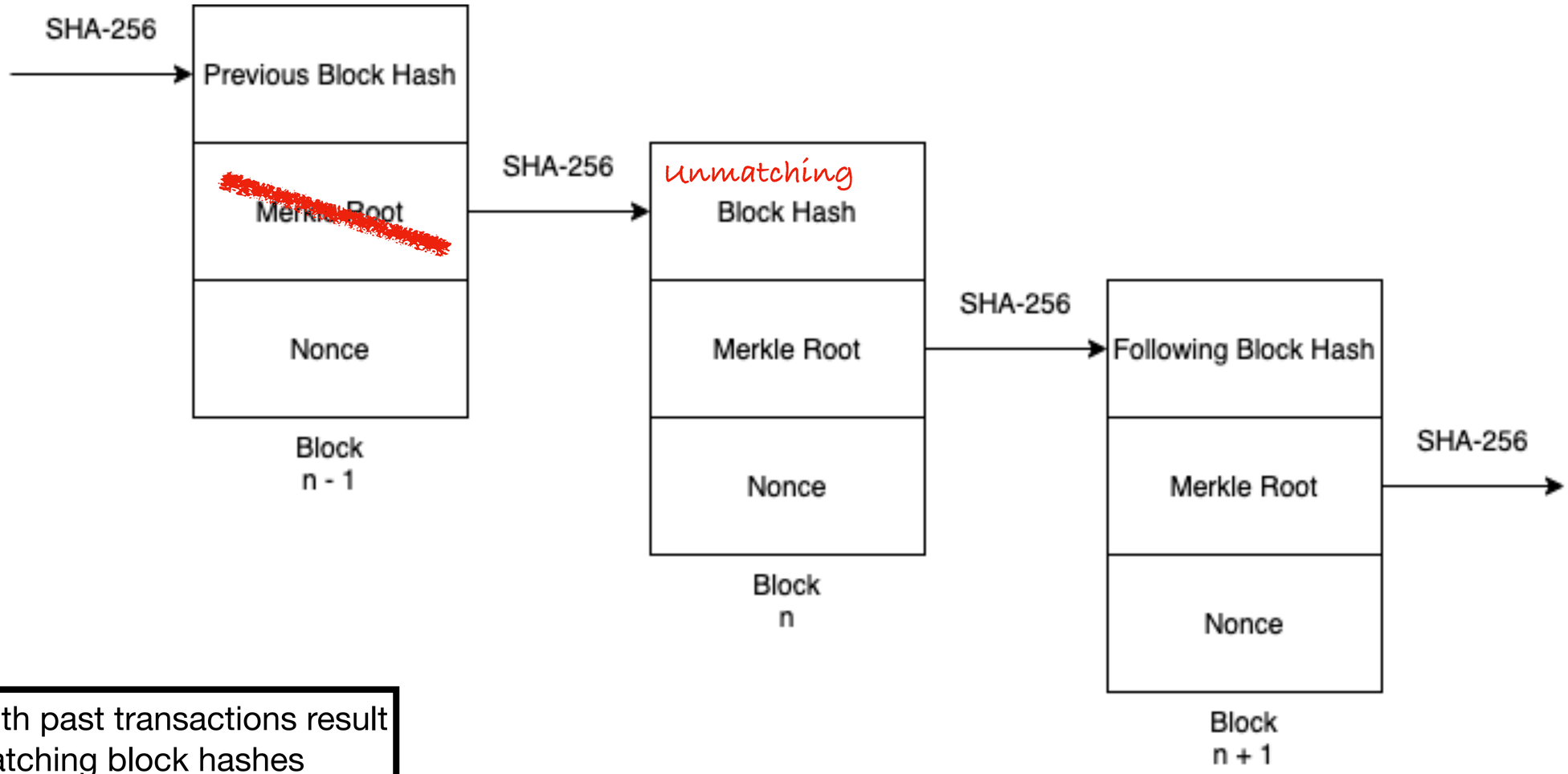
- Hash algorithm: creates an apparently random output from an input
- Hash algorithm used by Bitcoin: SHA-256
- Try it out: https://emn178.github.io/online-tools/sha256.html

Block Hash



Merkle Root ~ Transactions

Block Hash



Tampering with past transactions result in unmatching block hashes

Block Hash

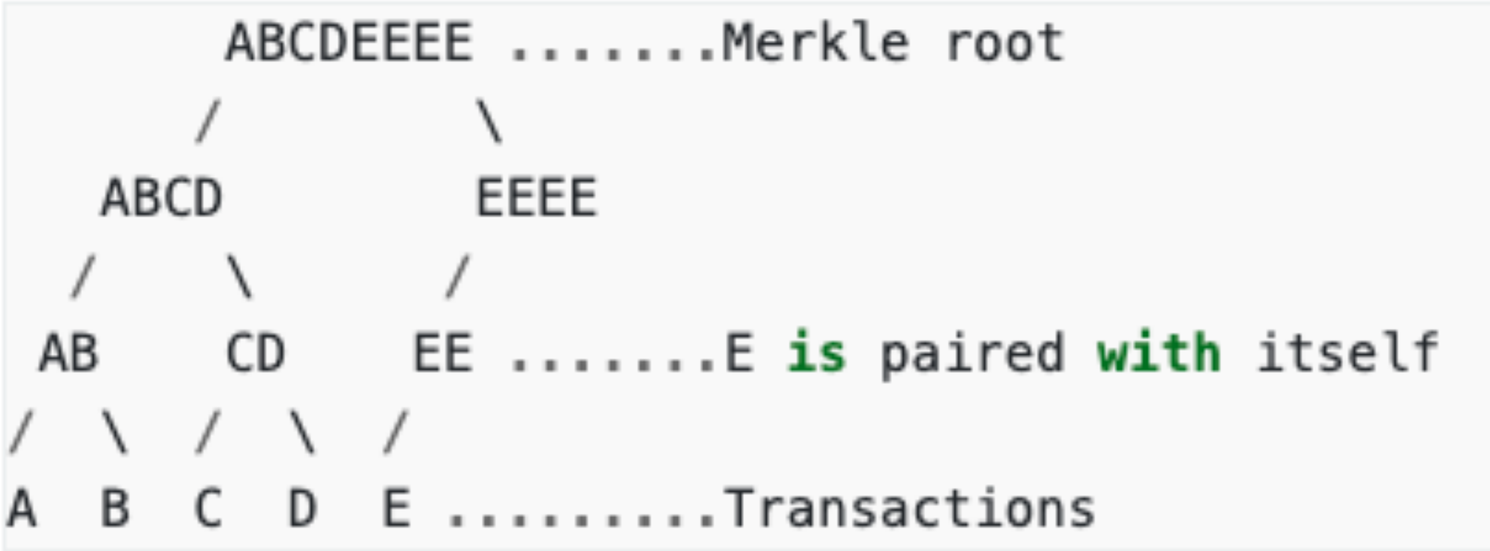
The Merkle Root

- Every block must contain one or more transactions
- The first transaction must be a “coinbase transaction”: it should collect and transfer the block reward (= block subsidy + transaction fees)
- All other transactions are optional, but are usually included to increase the reward
- The block reward cannot be spent before at least 100 blocks have been mined (~ 1000 minutes) - in case the block is determined as stale later on

Block Hash

The Merkle Root

- All transactions are encoded into blocks in binary raw transaction format
- They are paired with each other, hashed, then the hashes are paired again, hashed, and so on, until it only leaves on final hash: the Merkle root
- Any hash without a partner is hashed with itself

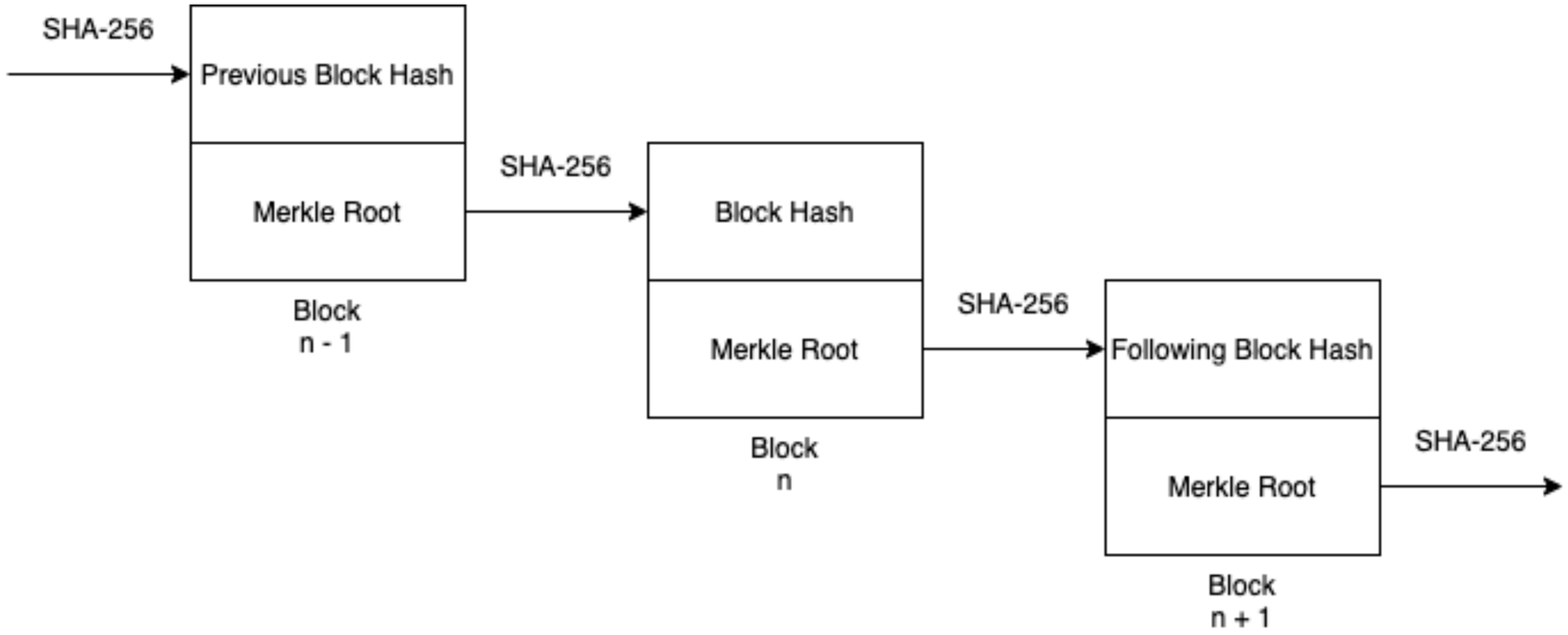


Part 2: Bitcoin

1. Introduction
2. Block Hash
3. **Proof of Work**
4. P2P Network

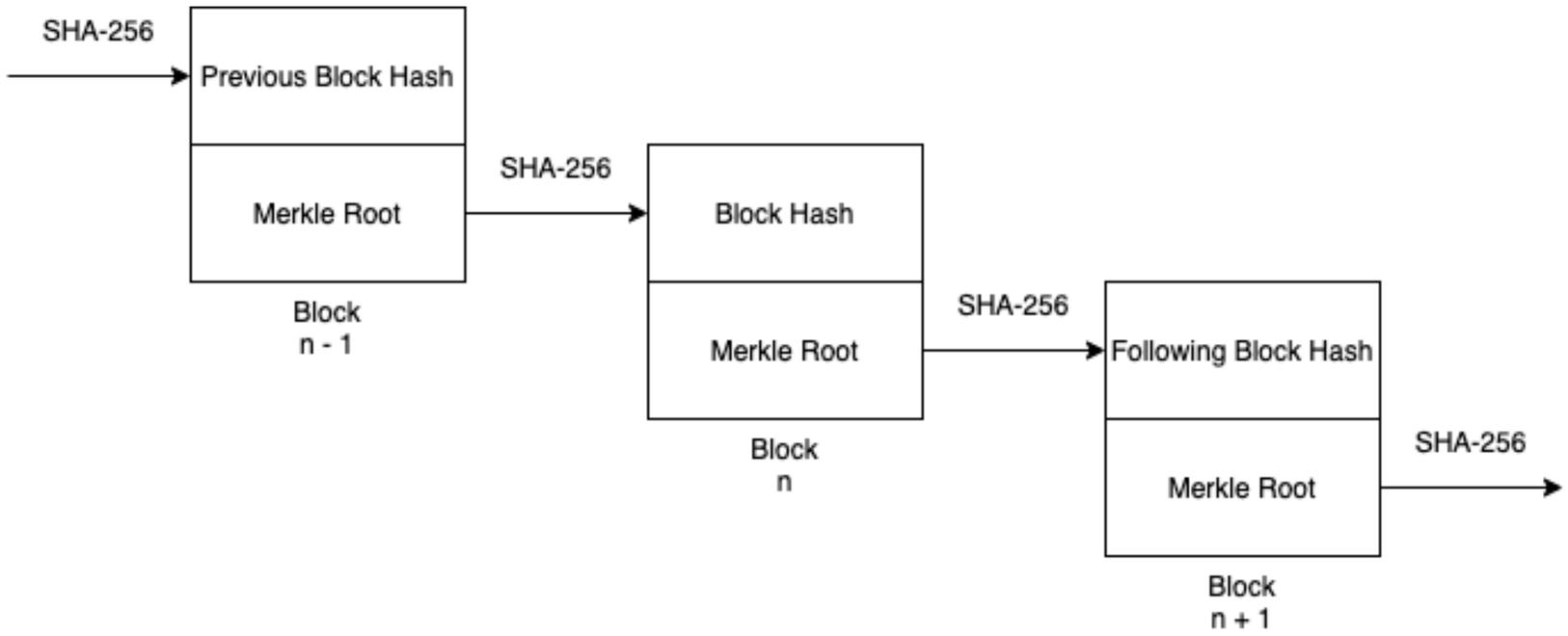
Proof of Work

What is the problem with the following diagram?



Proof of Work

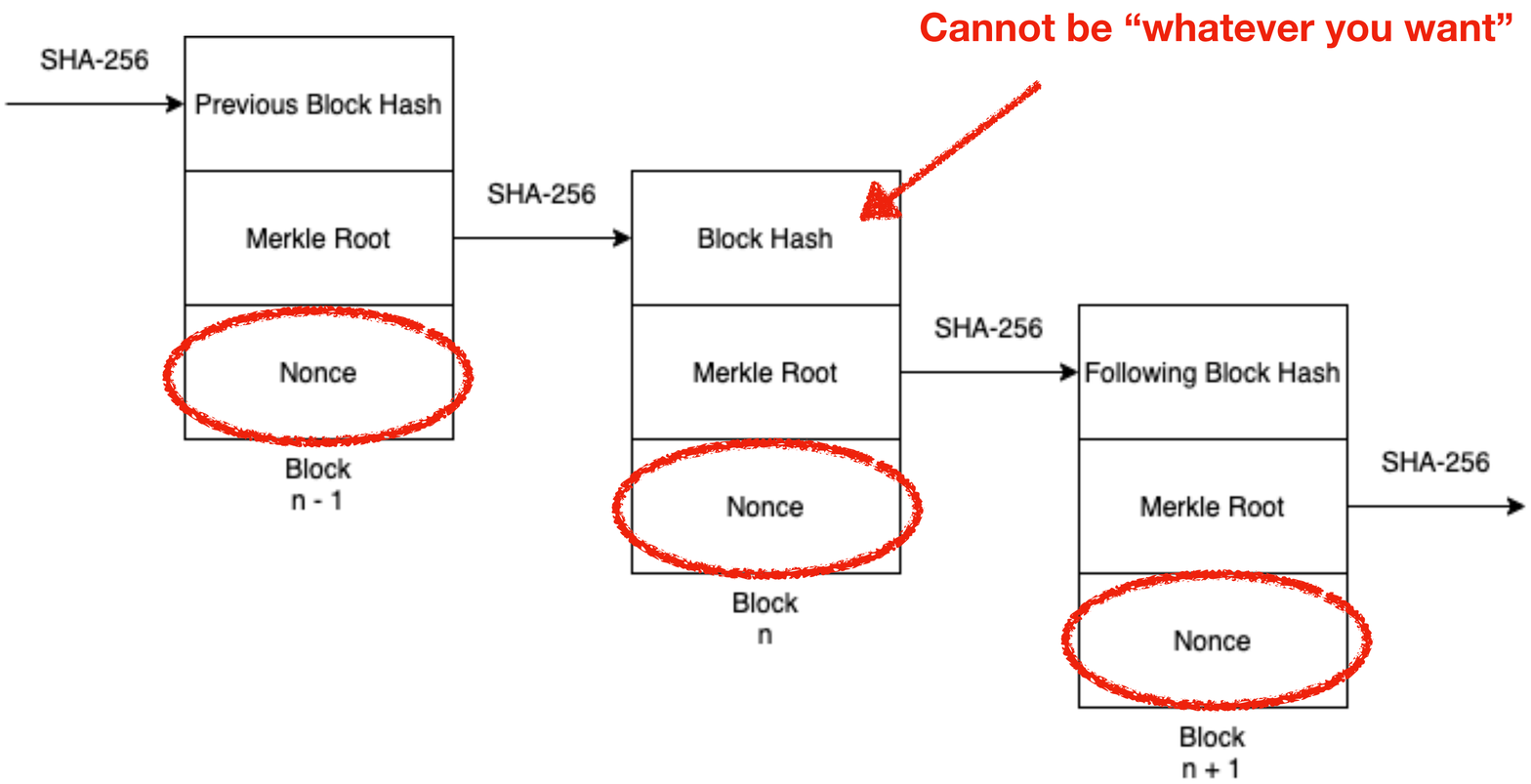
What is the problem with the following diagram?



- It would still be secure against past block tamper…
- …For maybe 2 minutes: brute force can recreate a block chain really fast

Proof of Work

This is why were added 2 things…



- A constraint on the resulting Block Hash
- A nonce field to satisfy that constraint

Proof of Work

A step back…

- Proof of Work has not been invented by Satoshi Nakamoto
- It was first described and implemented in 1993 to deter Denial of Service attacks, and was also used the same way against SPAM
- The idea is to use an asymmetric problem so that given a problem to solve “N”, you have to find a proof “P” so that (for example) the hash of “N-P” satisfies a given condition, for example to start with 40 zeros
- Finding such a proof would require a huge processing effort, while verifying SHA-256(“N-P”) starts with 40 zeros would require only one step

Proof of Work

Bitcoin case

- Problem: Merkle Root + Previous Block Hash
- Proof: the Nonce field
- Condition: the hash of the block header must not exceed a certain value (<=> start with a certain number of 0)

- => The lower the value, the harder it is to find a valid Proof
- The difficulty is reevaluated every 2016 blocks so in average, a new block is mined every 10 minutes

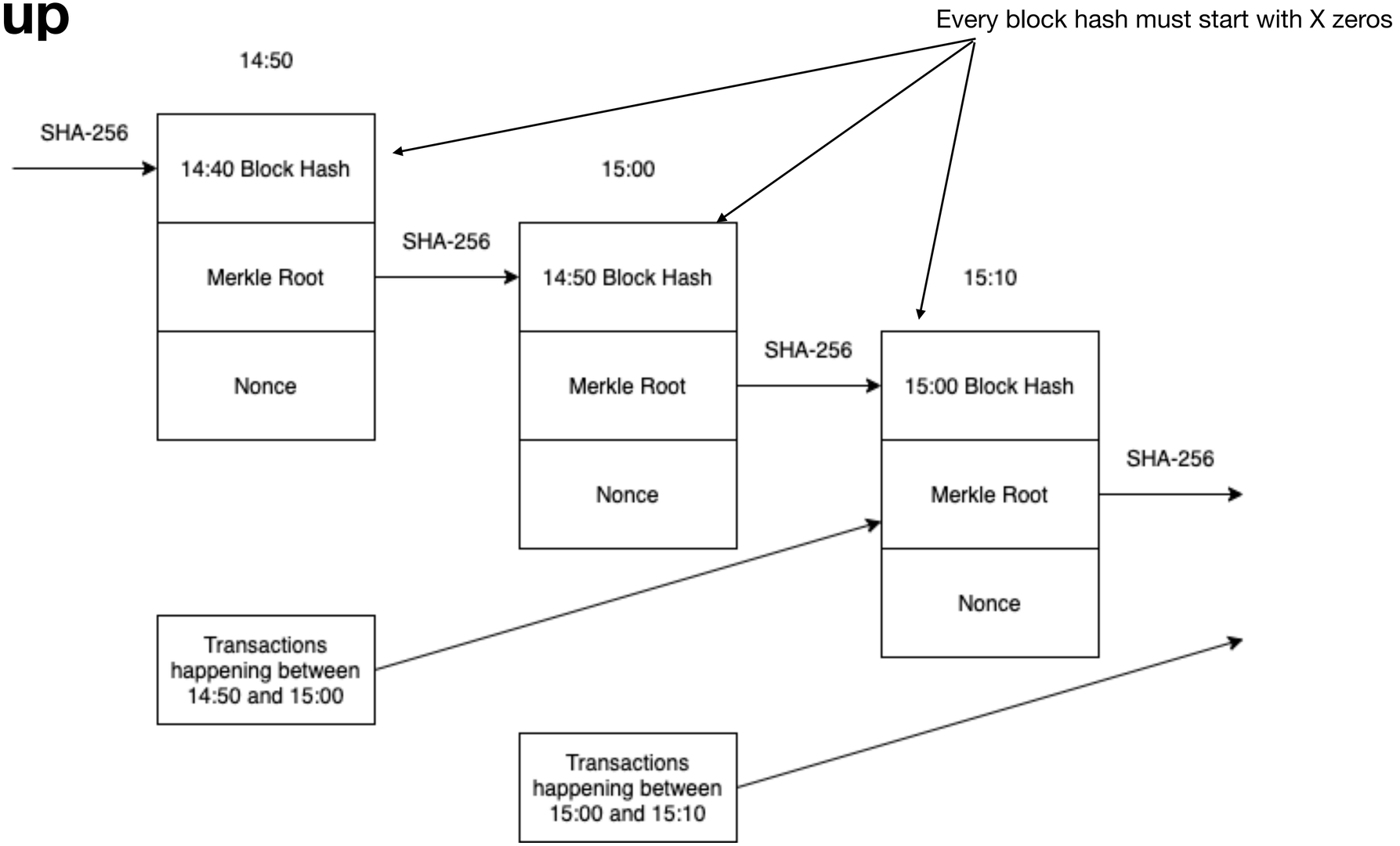
Proof of Work

Bitcoin case

- Consequence of this Proof of Work with a “10 minutes difficulty”: tampering with an older block entails that you also need to recalculate the proof of work of all the following blocks to have a coherent block chain

Proof of Work

Summing up



Part 2: Bitcoin

1. Introduction
2. Block Hash
3. Proof of Work
4. **P2P Network**

P2P Network

Centralised vs Decentralised

Centralised	Decentralised
• One server and one copy of the data to be provided to the entire world: single point of failure • Only one owner of this copy who can do whatever he wants with the data • Faster and cheaper	• Data provided through a consensus of nodes: no single point of failure • Data is copied to all Bitcoin miners • Data is immutable • Anyone is allowed to join the network • Slower and more expensive to run

P2P Network

Network process

- According to the Bitcoin white paper:

5. Network

The steps to run the network are as follows:

1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not already spent.
6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

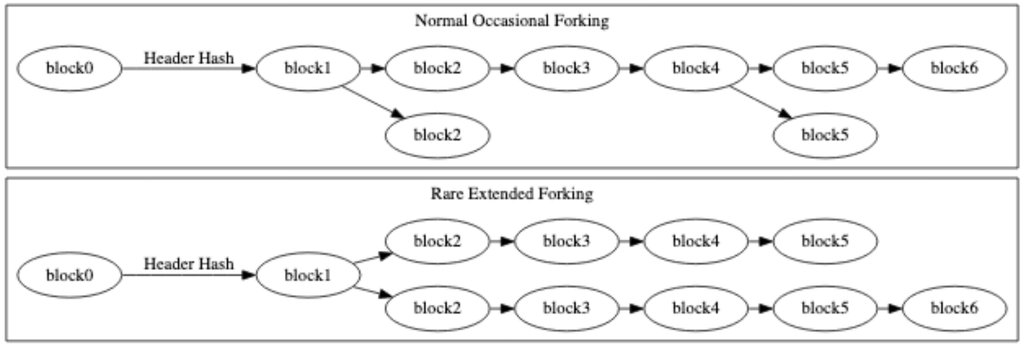
New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.

Source: bitcoin.org/bitcoin.pdf

P2P Network

Forks

- In different few conditions, 2 blocks can compete with each other, resulting in a chain fork:



- Because of those forks, the block height (the number of blocks between the Genesis block and the current block) cannot address a block. We usually use its hash to address it.
- Examples of reasons: malicious attacks, 2 blocks found at the same time, consensus rule change

P2P Network

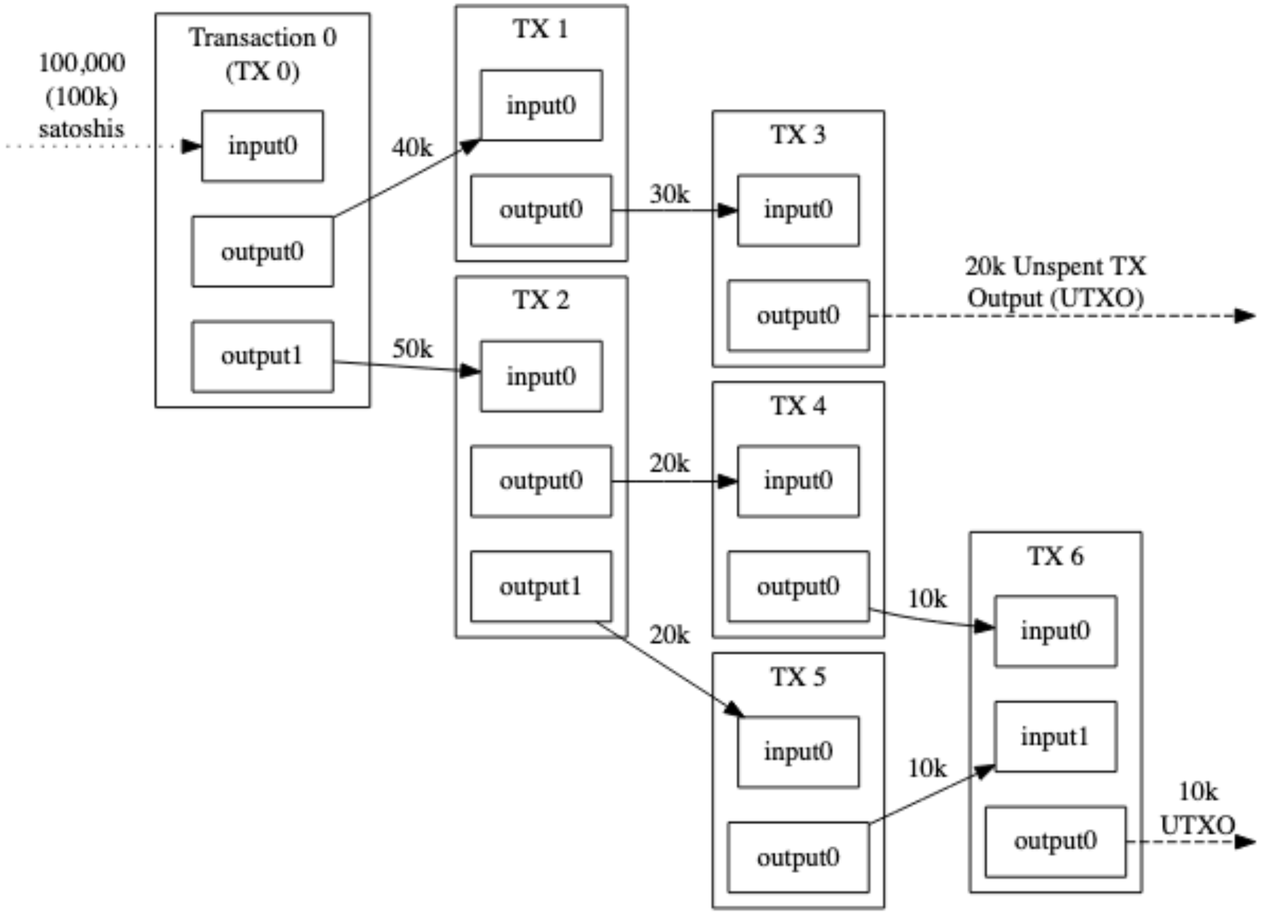
51% Attack

- A 51% Attack: when someone (or a group) who has the majority of the network hash rate revise transaction history and/or prevent new transactions from confirming
- Note: this can happen because it’s a race between which block gets propagated the fastest; also, it means than you don’t necessarily need 51% of the hash power
- Those never happened on Bitcoin

P2P Network

Transactions

- Just like block hashes which are chained together, transactions are also chained together
- Satoshis and bitcoin are not sent from wallets to wallets, but move from transactions to transactions
- A single transaction can create multiple outputs, but each output of a particular transaction can only be used as an input once in the block chain (forbids double spend)
- Transactions mainly contain the sender's address, the recipient's address, the amount of satoshis to send
- Transactions also have a script field to create "contract transactions"



Triple-Entry Bookkeeping (Transaction-To-Transaction Payments) As Used By Bitcoin

P2P Network

An example of Security issue handled

- On 6 August 2010, a major vulnerability in the bitcoin protocol was spotted
- Transactions weren’t properly verified before they were included in the transaction log, which allowed users to create an indefinite numbers of bitcoins
- On 15 August, over 184 billions bitcoins were generated in a transaction and sent to 2 different addresses
- Within hours, the transaction was spotted and erased from the transaction log after the bug was fixed and the network forked to an updated version of bitcoin protocol

P2P Network

Consensus rules

- What allowed the transaction to be validated during that attack were weak consensus rules
- The consensus rules are what say which block should be valid or not at the block validation step, during the propagation of the said block among the nodes
- This allows all nodes to reach a consensus
- Consensus rules can include the size of a block, the kinds of transactions allowed, the mining reward, the transaction fee amounts…

Final resources

Totally optional

- The bitcoin developer guide: developer.bitcoin.org
  You'll get to know more about transactions, contracts, wallets and P2P Network

- Entire Bitcoin history: http://historyofbitcoin.org/

Conclusion

…To the introduction

- **Block hashes** are chained together, making it a block chain, with transaction data in each block. Tampering with a block would require to change all the following blocks
- Changing all the following blocks would take time because a **Proof of Work** is required for each block
- Even if you're able to provide an alternative chain with tampered transaction data, you would need to propagate it to the majority of the **network**
- All these make Bitcoin an excellent and secure decentralised **currency** based on a P2P **network**, but… What if you want to make more than a currency?

Block Chain, Bitcoin & Security

…But truly Bitcoin, Proof of Work and Smart Contracts Development

EPITA - 2021

Part 3: Ethereum

1. Introduction
2. How does it work?
3. Ethereum ecosystem

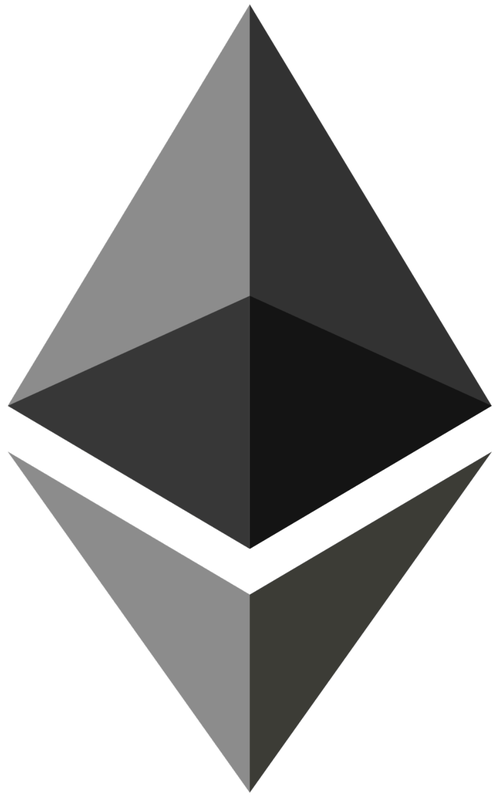
Part 3: Ethereum

1. Introduction
2. How does it work?
3. Ethereum ecosystem

Introduction

Why does Ethereum exist?

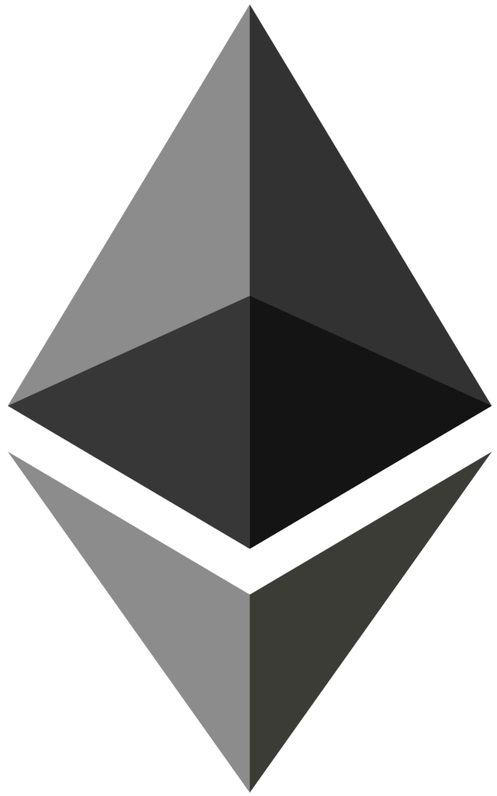
- Bitcoin was “just” a decentralised currency
- There was the need to decentralise more things (real estate ownership, artwork ownership, games, social networks…)
- Having to build a new network with enough peers and hash power for each of these was a solution, but a tough work



Introduction

What does it do?

- Ethereum is a platform to upload Decentralised Apps (Dapps), based on what are called smart contracts
- Just like Bitcoin, Ethereum takes profit of a decentralised network, but anyone can code programs called “smart contracts” on it
- It uses “ether” as its currency to reward miners and to pay transaction fees



Ethereum

Goals

- The goal of Ethereum is to decentralise the Internet
- It would do it by providing a secure, decentralised internet
- It truly enables “cutting the middle-man”: Uber, Amazon, eBay, YouTube (music producing)… And fund raising (Kickstarter => ICOs)

Part 3: Ethereum

1. Introduction
2. **How does it work?**
3. Ethereum ecosystem

How does it work?

A few reminders

- Ethereum is still based on a Blockchain, so most things which are true for Bitcoin are true for Ethereum: Block hashes, Proof of Work (for now), P2P Network
- You also still have wallets (private key, public key), transactions, mining rewards…

How does it work?

A few things about smart contracts

- The real innovation here are the smart contracts: these are simple byte codes that are ***immutable*** and ***won't be owned or controlled by any user*** (unless…)
- Those smart contracts will have a public address and their byte code will be made public, so that the community can audit its features and security
- Smart contracts functions can be called by anyone, but only accept external calls…

How does it work?

What “external calls only” entail

- You NEVER need to log in to use an Ethereum smart contract
- CRON jobs cannot be done in Ethereum (unless…)
- It may be vulnerable to DDoS attacks
- To counteract that, Ethereum includes a “transaction fee” for (almost) every function call
- In consequence, intensive computing tasks can cost a lot… And are forbidden by Ethereum
- It is very hard for an Ethereum smart contract to use data from “the outside world”

How does it work?

The “unless” answers

Constraint	Solution
Smart contracts are immutable	You can reference another smart contract and use its functions
Smart contracts are not owned by anyone	You can code an “owner” in the smart contract constructor and allow him to call functions that only he can call
CRON jobs are impossible	You can code your own server to make CRON job calls from your server; services called “Oracles” also do that
Impossible to use data from the outside world	Oracles can help with that

How does it work?

Side note on “immutable”

- Immutable means it cannot change… But what cannot change?
- The hard code of any smart contract will never change. However, your smart contract may depend on state variables that you may change if you write a function dedicated to it.
- **Truly, what is immutable is the history of what has ever happened on the blockchain.**

How does it work?

Cons

- Calls to functions are much slower than by using a simple server-based API: because each function call is a transaction, the transaction must be validated and propagated through the network.
- Ethereum is a very poor database: since storing anything on Ethereum would make a copy of it on every node in the world, it is both inefficient and expensive.

Part 3: Ethereum

1. Introduction
2. How does it work?
3. **Ethereum ecosystem**

Ethereum ecosystem

Notable library and smart contracts

- ERC20: the “standard” token developed by the Open Zeppelin team. It allows the easy creation of “tokens”, that you can use to create your own currency in minutes.
- Crowd sale smart contracts: from Open Zeppelin as well, those smart contracts allowed the ICO craze of 2016-2017.
- ERC721: the “standard” non-fungible token. That means that each token may be different, cannot be divided. Ideal for any collectible.
- ERC1400: a standard for “security tokens”: these should be technically considered as ERC20, but should have additional features to allow them to be legally valuable as securities.

Ethereum ecosystem

2 languages

- Solidity: the most popular language for smart contract development. It has more library, a bigger community, open source codes are more easily audited. The language looks like JavaScript or C++.
- Vyper: another language with an emphasis on security and math functions. Some security and math features are native to the language itself.
- It is important to note that both languages will compile to give the following outputs:
  - an *abi* file (looks like a json file, has the same function as a .h in C)
  - A bytecode which is assembly for the Ethereum VM

Ethereum ecosystem

Solidity compilers and migration managers

- Truffle: develop in your favorite local IDE, unit test more easily: https://www.trufflesuite.com/docs/truffle/overview
- Hardhat: just like Truffle, easier to use and to debug: https://hardhat.org/
- Remix: develop and deploy more easily and faster through their online IDE: https://remix.ethereum.org/

Ethereum ecosystem

What will be used for this course

- Solidity: most popular programming language for Ethereum smart contracts
- OpenZeppelin-contracts: most popular library for Solidity
- Hardhat: in order to self-host smart contracts code on GitHub
- Alchemy: a hosted Ethereum node cluster to deploy smart contracts
- Metamask: a Chrome extension to sign the transactions (for deployment and function calls)
- Ropsten: a “testnet”, as opposed to “main net”, to avoid spending real ethers for tests
- MyEtherWallet: a platform that will help us ease interactions with smart contract