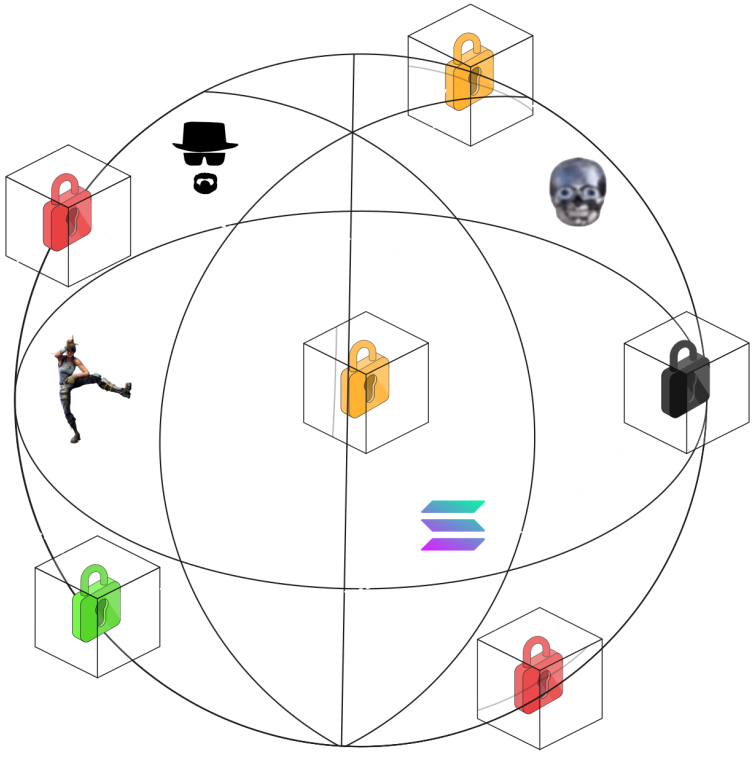
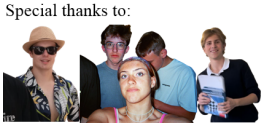
A deep dive into the technology behind cryptocurrency and creating my own

Christian Tognazza
6dG

Supervised by:
Michael Graf, Martin Wildhaber



Special thanks to:



Kantonsschule Rychenberg

06.12.2022

Table of contents

1 Introduction

In 2016 and again in 2021, financial records were broken, with the market capitalization[1] of the most popular cryptocurrencies reaching $828 billion and a whopping $2.8 trillion, respectively. Cryptocurrencies have dominated newspaper headlines; by now, they are a widely known concept, and most of us have already formed an opinion on them. Many refer to cryptocurrencies as the future of finance, while others demonize them, declaring them as scams and valueless. I am incredibly interested in cryptocurrencies and asked myself a bunch of questions regarding cryptocurrencies. What I am most interested in, however, is not their financial aspect but the technology powering this possible revolution. This essay focuses on how cryptocurrencies and their underlying blockchains work behind the curtains and tries to build one's knowledge about cryptocurrencies step by step. In addition to this theoretical report, I developed my own cryptocurrency and blockchain, which is available for the public to use and experiment with through the link https://matura.chrigeeel.dev/blochchain.

1.1 What is a cryptocurrency?

A cryptocurrency is a digital good designed to *eliminate trust* in our financial world. It is an asset created through cryptographic methods, making it impossible to forge. The main point of cryptocurrencies is *decentralization*. Unlike traditional currencies such as the US dollar, national governments or treasuries do not control cryptocurrencies. They can circulate without monetary authority, such as a bank, and no single entity can ever control them. As a result, if one owns any cryptocurrency, they truly own it, meaning that no bank and no government possess the ability to take the cryptocurrency away from them if they want to.

1.2 Use cases of cryptocurrencies

One might ask themselves why our society would need such a decentralized financial system if the current system with a central authority works just fine. While everything might seem great at the moment, countries can go into deep recessions because of bad economic decisions made by the government. *Inflation* is one of the biggest threats to the average person. People in Venezuela have to deal with an annual inflation rate[2] of 155% at the time of writing, and people in Russia with an annual rate[3] of around 13%. This might not seem like a lot but imagine someone saving up to buy a house or something similar and their money not even being worth half as much as the year before. Cryptocurrencies remove this trust one has to put into their national government by controlling inflation programmatically and having a fixed inflation rate, or one that everyone participating in the network agrees on.
Another problem some people face is being locked out of their personal financial accounts. An entity like PayPal has complete control over one's funds if one uses their service. This is extremely dangerous for the day-to-day person if PayPal cannot provide them with their money or decides to lock them out of their account for an indefinite time. Cryptocurrencies give power to the people. One truly owns their funds and has complete control over them, no matter what the government or any other powerful entity has to say.

[1] CoinMarketCap (2022) - https://coinmarketcap.com/charts/ (22.11.2022)
[2] Statista Inc. (2022) - Venezuela: Inflation rate from 1985 to 2023 - https://www.statista.com/statistics/371895 (07.11.2022)
[3] Statista Inc. (2022) - Year-over-year change in consumer prices in Russia from January 10 to October 31, 2022 - https://www.statista.com/statistics/1298843 (07.11.2022)

Moreover, cryptocurrencies allow for cheap, fast international transfers. This is a luxury for people living in third-world countries but also for people living in more modern environments. In 2022, a bank transfer in Switzerland can take up to three business days or longer, depending on the bank one decides to use.

2 Fundamentals

Let us look at some fundamentals and definitions of blockchain technologies to be then able to further understand the technology behind itself and cryptocurrencies.

2.1 Definition and objective of blockchain

A blockchain is a distributed, *iimmutable ledger* that is shared among the computers of a network. A blockchain stores information similarly to a database and can have many use cases in software that prioritizes security. However, they are mostly known for their crucial role in cryptocurrencies. The main objective of a blockchain is to maintain a secure, accessible, and decentralized *record of transactions* that cannot be edited.

2.2 What is a transaction?

"A transaction is a transfer of value on the blockchain. In very simple terms, a transaction is when one person gives a designated amount of cryptocurrency they own to another person."[4] A transaction[5] contains so-called *instructions*, which are pieces of information that tell us who transfers how much funds to whom. As a result, one particular transaction can contain multiple instructions, which in turn means that a single transaction can contain multiple transfers of value. One person could therefore send funds to two people in a single transaction. In most networks, the sender of a transaction has to pay a small fee as an incentive for their transaction to be added to the blockchain by those who participate in running it.

2.3 What is a hash?

A hash is produced by a *hash function*[6]. An output of a hash function is of fixed length, while the input can be of any size. A hash function can take any input and produce an *unrecognizable* hash from said input. Hashes are one of the cryptographic primitives used in blockchain technologies.

2.4 What is a block?

A blockchain is made up of subsequent blocks, which are chained together. Thus the name "blockchain". But what exactly is a block? Blocks are *containers for information* on the blockchain and contain different data, mainly transaction records and the last block's hash. The structure of a blockchain can be compared to the structure of a linked list in programming, where each block "points" to the last block with the previous block hash.

[4] Orenes-Lerma, Linda (2022): https://www.ledger.com/academy/how-does-a-blockchain-transaction-work (07.11.2022)

[5] Bitcoin Wiki contributors (2021) - Transaction - https://en.bitcoin.it/wiki/Transaction (07.11.2022)

[6] Wikipedia contributors (2022) - Cryptographic hash function - https://en.wikipedia.org/wiki/Cryptographic_hash_function (07.11.2022)

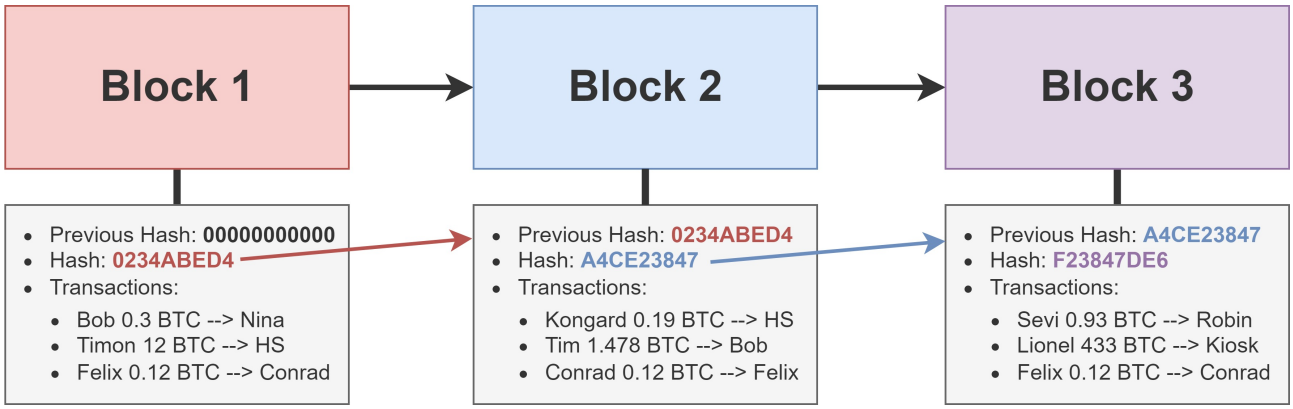


Figure 1: Linked Blocks (own work)

In detail, blocks consist of the following things[7]:

- **Magic number**: A number that indicates that this block is part of a blockchain network. This number was randomly chosen and had no special purpose other than declaring this piece of information - the block - as part of a blockchain network.
- **Blocksize**: Sets the size limit in bytes on the block so that only a specific amount of information can be written in it.
- **Block header**: Contains additional information about the block, such as the previous block hash and the hash of the current block header itself.
- **Transactions**: A list of all transactions within this block.
- **Transaction counter**: The number of transactions in this block.

The hash of the block is a hash of the block header. This block header contains the previous block hash. This chains all blocks together in an exquisite manner. Making one ever-so-slight change to the block header of any previous block would also affect all subsequent blocks. All blocks in a blockchain are chained together cryptographically.

2.5 Networking

Since a blockchain is a peer-to-peer network[8] of computers, any computer meeting its technological requirements can be a part of it. A computer that partakes in a blockchain is called a node. Being a peer-to-peer network means that these computers communicate with each other directly without using any middleman server through the internet. A node usually completes tasks within the network, such as validating information, and for completing those tasks, it is rewarded in cryptocurrency. A node's primary task is to confirm the validity of each batch of each subsequent batch of network transactions, known as a block.
I have asked myself how a new node can find its peers so it can participate in the network and how exactly nodes broadcast new blocks if there is no known central server to communicate with. When another peer is already known, it is simple to find other peers because every node keeps track of its peers in a local database. This means that if one node knows another node, it can simply ask it what other nodes there are and therefore build its local database of available

[7] Bitcoin Wiki contributors (2019) - Block - https://en.bitcoin.it/wiki/Block (07.11.2022)
[8] Bitcoin Wiki contributors (2018) - Network - https://en.bitcoin.it/wiki/Network (07.11.2022)

peers. Nevertheless, how do we find peer number one? DNS seeding is the solution to our problem. DNS seed servers are specifically configured to return a random list of known bitcoin nodes. This means that our bitcoin node asks a well-known DNS seed server about other nodes in the network. However, what if we cannot connect to a DNS seed server, or the people running them turn them off? Additionally, in almost every implementation of cryptocurrencies such as Bitcoin, a list of nodes believed to have a near-permanent uptime is hardcoded in the source code the nodes run. If DNS seeding fails, the node tries the IPs in that hard-coded list and goes through it until one of them works. Broadcasting a new block to the network is rather simple and is usually done using a gossiping protocol, which gets the name because of its similarities to real-life gossiping. Since every node has a list of some of its peers, one node can broadcast the new block to its known peers. Those peers then do the same with their list of other peers. A simple filter stops infinite recursive broadcasting. Using this gossiping protocol, any node can update the entire network in only a few seconds or less.

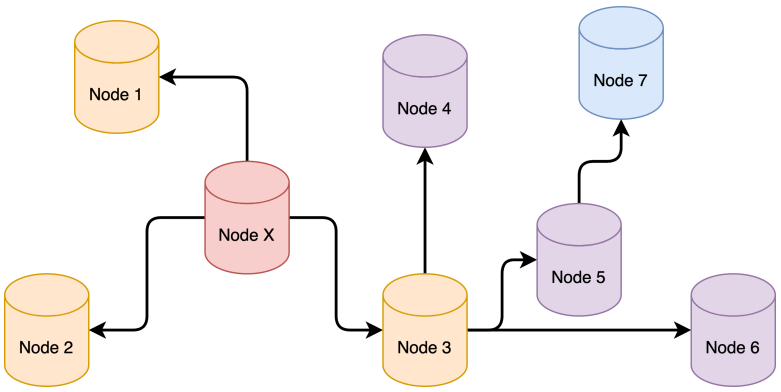


Figure 2: Gossip Protocol (own work)

Another interesting question one might ask themselves is how an outside computer can communicate with one of these nodes. A node has the ability to activate its RPC (Remote Procedure Call) abilities. Explained briefly, this means that in addition to trying to build new blocks, a node with this feature enabled also is the host of an HTTP server. This HTTP (HyperText Transfer Protocol) server can be protected using passwords or can be fully public for everyone to use. An HTTP server is a computer that allows other computers to communicate with itself to receive and send information. Users can communicate with this HTTP server of any node to receive information, such as someone's balance, or post data to them to initiate a transaction.

2.6 Mempool

The mempool, a short-form concatenation of the words "memory" and "pool", is where transactions are stored when they are waiting for a new block to be confirmed. Since blocks are added periodically, transactions must have a waiting room where they can be stored from when they are submitted to when the next block is added. This "waiting-room" is called the mempool. We often speak of "the mempool", but it is important to note that there is no single mempool shared between nodes. All nodes run their own mempool, but since a transaction is broadcast to the whole network when it is being submitted, all nodes usually have the same transactions in their mempool, just with a slight delay.

3 Identity, authentication and ownership

With the internet growing unbelievably large, one's *digital identity*[9] is becoming more critical every year. Digital identification and authentication are considerable concerns for blockchain and cryptocurrency technologies. If one wants to send someone funds, one cannot list their name and address as one might when making a traditional bank transaction. There is no centralized database able to keep track of one's funds with a username and password. When someone sends someone else funds, they should be the sole owner of those funds without anyone having the ability to take them away from them. Without someone checking for passwords, we still need to be able to tell with absolute certainty if a transaction is valid and if the sender, in fact, intended to execute said transaction. What if we could have a *digital signature* that could be signed by only one person but verified by everyone?

3.1 The solution to digital ownership

The solution to the problems discussed above exist and are*public key cryptography*, *keypairs*, and *digital signatures*. The ownership of cryptocurrency is established through the use of keypairs and their signatures. A keypair consists of a private and public key, and neither is stored on the blockchain but instead generated by end-users and stored offline on any computer drive or even a piece of paper. The reason for this offline storage is the fact that keypairs are sensitive information that must be kept secret to maintain full access to one's funds.

3.2 Keypairs

As already mentioned, a keypair consists of a private and public key. Their names already explain the main difference between the two. A private key should always be kept private, while a public key can be published safely. A private key is simply a big number and is the "password" to access one's funds. It is what grants permission to do things on the blockchain. Since the private key has to be kept private to keep one's funds safe, we use public keys as "addresses" on the blockchain. Public keys can be calculated using the private key, but it is virtually impossible to calculate the private key by only having the public key. One's public key can be given to others so they can send funds to oneself, while the private key is used to send funds to other people.

3.3 Signatures

A digital signature is comparable to a physical signature. One can use it to show that one owns the private key connected to a public key without having to reveal the private key. A private key can sign any piece of data to retrieve a signature. That data can be a transaction to show that we really intend on sending funds to someone else, or it can be a simple text message such as "Hello World!" that we want to send to a friend. One might ask themselves what it is that stops someone from simply taking one's signature and using it to send malicious transactions or messages. To make a signature, one uses not only one's private key but also the data that one wants to sign. If someone has a private key and signs the message "Hello World!" and then signs the message "Hello everyone!" using the same private key, the signatures would be

[9] Antonopoulos, Andreas (2017) - Chapter 4. Keys, Addresses - https://www.oreilly.com/library/view/mastering-bitcoin-2nd/9781491954379/ch04.html
(07.11.2022)

drastically different. Therefore, each digital signature is *strongly tied* to the data it stands for and, therefore, cannot be used to forge counterfeit messages or transactions.

3.4 Seed phrases

A seed phrase[10] is everyone's base for storing keypairs on the blockchain. This use of seed phrases, also called *mnemonic phrases*, was first introduced on 10 September 2013 with *BIP39*, or Bitcoin Improvement Proposal: 39. A seed phrase is a list of 12-24 words randomly chosen from the standard BIP39 wordlist, which contains a total of 2048 different words. In that list, the first four letters of each word are unique. That list of 12 to 24 words represents a number of bytes, which can be used with a few mathematical functions to derive one's private key, and, in turn, recover one's funds if one ever lost one's private key. A question one might ask themselves is why we should make use of seed phrases when we could remember the private key itself. The primary use of seed phrases is their property of being *human-readable*. It is very hard even to write down a private key correctly since it is a random, huge number. On the other hand, remembering or writing down 12-24 words is a lot easier for humans and therefore makes for a far better user experience. Using words also induces excellent error correction. Bad handwriting can still be read, and a word can often be deciphered even if a few letters are missing.

Figure 3: Example seed phrase

3.5 Privacy and anonymity

A common misconception about most cryptocurrencies is that one's activity is private, which could not be further from the truth. Every single action on-chain, whether a simple transfer of funds or placing a bet on an online casino, is *traceable* back to one's public key. Everyone can

[10] Narazanan, Arvind and Bonneau, Joseph and Felten, Edward and Miller, Andrew and Goldfeder, Steven (2016) - Bitcoin and cryptocurrency technologies: a comprehensive introduction - Princeton: Princeton University Press

look up exactly where one got their funds, how they spent them, or what other activities one has done on the blockchain. So-called "Explorer-Websites" exist for almost every cryptocurrency out there. They make it easy for anyone with an internet connection to enter someone's public key and look up that person's entire record of activities.
Anonymity, however, can be achieved on the blockchain. An average person usually cannot link one's public key to one's real-life identity, and vice-versa. What often is the case, though, is that one has to register to a cryptocurrency exchange with their full name and address to buy cryptocurrencies safely. In a legal case, for example, the government can force these over-the-counter exchanges to give them someone's information and deanonymize them.
If used correctly, anonymity and privacy can be guaranteed on cryptocurrency networks, such as Monero. Monero uses special, highly complex cryptography that allows secure transactions to be made without the sender knowing what public key they are sending funds to. To find out more about Monero, one might want to look up *ring signatures* and *stealth addresses*.

4 Consensus

Consensus is often defined as general agreement. Consensus needs to be achieved in blockchain technologies to agree on one particular state of the blockchain. Consensus in cryptocurrencies can be reached through different *consensus mechanisms*.

4.1 What is a consensus mechanism?

Imagine someone had the task of synchronizing a network of computers. At first, this would seem very easy to complete since one can send the information one needs to synchronize from one computer to another. However, why should computers assume that the information they receive is coming from a peer they can trust? How can they be sure that the peer they are receiving information from is not sending them erroneous information or leaving out pieces of information on purpose? After more consideration about trust, this task becomes significantly harder to solve.
In the modern world, computers should not trust each other. Humans are the ones programming computers and are able to tell computers precisely what to do. Humans are not always honest and often cannot be trusted, especially when they might receive some sort of personal gain due to not telling the truth. In an attempt to solve this problem of trust in the digital world, people have come up with consensus mechanisms[11] [12]. The goal of a consensus mechanism is to reach consensus - the same opinion - in a network of computers with close to zero trust between them and with as little effort as possible.

4.2 Proof of work

The Proof of work consensus mechanism has been in the news headlines lately due to some of its controversial attributes. In the proof of Work system, a big but doable amount of effort must be made in order to detect malicious uses of computer power, such as sending faulty transactions or spam emails. The concept was first used in securing a digital currency by Hal Finney in

[11] Wikipedia contributors (2022) - Consensus (computer science) - https://en.wikipedia.org/wiki/Consensus_(computer_science)
[12] Narazanan, Arvind and Bonneau, Joseph and Felten, Edward and Miller, Andrew and Goldfeder, Steven (2016) - Bitcoin and cryptocurrency technologies: a comprehensive introduction - Princeton: Princeton University Press (07.11.2022)

2004 through the idea of "reusable proof of work.". It is most known for its implementation in the Bitcoin[13] cryptocurrency in 2009. Hal Finney was highly involved in the coding of Bitcoin and was the first-ever recipient of a Bitcoin transaction. Today, proof of work is used widely in many cryptocurrencies for validating transactions and mining new funds. In proof-of-work networks, nodes are also called miners. This explanation illustrates how proof of work functions in the Bitcoin network but should hold generally true for most cryptocurrencies employing the proof of work mechanism.

4.2.1 How consensus is achieved

Since proof of work is a decentralized consensus mechanism, network members must agree on information without ever trusting each other. In proof of work, this consensus is achieved by solving an arbitrary mathematical puzzle[14] that requires a lot of computing power to be solved. The first computer to solve that puzzle can build the next block and is rewarded with the given cryptocurrency. Since the first person to solve the puzzle is chosen randomly and proportionally to the work completed, it incentivizes the computers in the said network to act truthfully and only build blocks based on valid transactions. One might ask themselves how a puzzle is chosen and what happens after the first computer solves it. This puzzle is based on hash functions and a target. A target is simply a number. Computers use nonces, numbers only used once, to generate these hashes and are looking for a hash lower than the target. As we discussed before, a block consists of a block header that contains a nonce. We hash the block header, including our random nonce, to calculate our output hash which has to be lower than the target. To reiterate, a hash is simply a random number computed by a hash function given a specific input. It is almost impossible to calculate the input used to calculate a hash just by knowing the hash. This means that finding a hash below the target requires pure brute-force computing power and that there is no "smart" way to do it. The more computing power one has, the more blocks one can build and the more rewards one can rake. In Bitcoin, the target hash is dynamic. It is adjusted every 2016 blocks so that the time between each block is approximately 10 minutes based on the total computing power available in the network.
Once someone finds the correct nonce so the hash of the block header, including the nonce, is lower than the target and builds the block, they broadcast their newly mined block to the network. The other computers in the network can verify the block they built, and if they believe it to be valid, they add it to their local copy of the blockchain and continue their task to find the next hash.

4.2.2 Rewards and incentives for partakers

Every block produced rewards the miner who generated that block with an amount of cryptocurrency, called the *block reward*. The block reward consists of the *cumulative fees* of all transactions included in the block and a *block subsidy*. In the case of Bitcoin and many other cryptocurrencies, this subsidy gets lower as more blocks get mined. The block subsidy is being sent in so-called coinbase transactions, which solely have an output and no input. It is the first transaction in every block and the subsidy amount gets cut in half every 210,000 blocks in the case of Bitcoin, which are approximately four years. In the Bitcoin network, the subsidy makes up most of the block reward and started at 50 BTC per block in 2009 and, at the time

[13] Nakamoto, Satoshi (2008) - Bitcoin: A Peer-to-Peer Electronic Cash System - https://bitcoin.org/bitcoin.pdf (07.11.2022)
[14] Bitcoin Wiki contributors (2021) - Block hashing algorithm - https://en.bitcoin.it/wiki/Block_hashing_algorithm (07.11.2022)

of writing, is at 6.25 BTC per block. Moreover, in Bitcoin, the block subsidy is locked for 100 blocks and cannot be spent during that time. The entire block reward incentivizes nodes to be honest, and confirm transactions truthfully. Furthermore, acting maliciously would result in far less financial gain than acting truthfully.

4.2.3 Advantages and disadvantages

The main advantages of the proof of work consensus mechanism are its extremely high levels of security in a well-developed environment. It is tough to monopolize computing power and take over a network. This also leads us to the first disadvantage and possible attack on the network: The "51% Attack". Suppose any single entity has ownership over more than 51% of the computing power in the network. In that case, it can create more than half of the blocks in the network and henceforth reach a consensus without any other entity agreeing. This allows for invalid transactions that do not require valid signatures and usually complete destruction of the network since the attacker can print funds out of thin air.
Another huge disadvantage is its energy consumption. Computing hashes en masse requires an unfathomable amount of energy and therefore damages the environment if non-renewable energy is used to participate in the network. This energy consumption also leads to higher transaction fees that have to be paid by users.
Proof of work is often regarded as a lousy consensus mechanism because of its energy consumption, but we should pay respect where it is due. Proof of work cryptocurrencies revolutionized the financial market and played a crucial role in the development of cryptocurrencies. Without proof of work, we would be nowhere near where we are today. The way proof of work is looked at as outdated beautifully illustrates the insane speeds the cryptocurrency world is moving at.

4.3 Proof of stake

Proof of stake has also been in headlines, but for the exact opposite reasons as proof of work. It is extremely energy efficient, and many dub cryptocurrencies using it the "future of finance". Proof of stake was first introduced in 2012 in the Peercoin network. It has since been developed further, and the most notable projects making use of it are Cardano, Avalanche, and Ethereum, which has recently made the switch from proof of work to proof of stake. In proof of stake, nodes are called validators. This explanation illustrates how proof of work functions in the Ethereum network but should hold generally true for most cryptocurrencies employing the proof of work mechanism.

4.3.1 How consensus is achieved

In proof of stake, consensus is achieved by validators "staking" their coins and validating transactions. Staking coins means locking them up and not being able to access them for a specific amount of time as collateral. Instead of using a competitive system for choosing who gets to build the next block, like proof of work, proof of stake selects the next builder, also called *forger*, randomly based on how many coins are staked. This means that the more coins a validator has staked, the more likely it is to be able to validate and build the next block. This requires much less computational effort than proof of work because no puzzle needs to be solved. A block is not built by a single validator, but rather multiple random validators are selected to vote on the block's validity for more security. When a specific number of validators deem the block accurate, it is finalized and added to the blockchain. The main thought behind this system is that someone with a lot of funds in the specific cryptocurrency is a lot less

likely to want to cause harm to the network and, therefore themselves. Another mechanism put in place to enforce this thought is so-called slashing. Slashing is a process that discourages miscreants and helps make validators more responsible. *Slashing* occurs when a validator is not behaving as expected, for example, when it tries to validate invalid transactions or blocks. If other validators notice bad behavior, they decide to slash the miscreant. If a validator is slashed, it loses a percentage of its staked funds, sometimes even all of them. Slashing is extremely important to keep a proof of stake network secure. Block times in proof of stake are fixed because no puzzle needs to be solved.

4.3.2 Rewards and incentives for partakers

Similar to proof of work, validators receive the transaction fees paid by users. Although, in proof of stake, this usually is only a small amount per transaction. Validators also receive rewards when they make votes consistent with the majority of other validators. This reward is usually proportional to the amount of cryptocurrency staked by a validator. This generally results in an annual percentage yield of three to ten percent, varying from coin depending on the size of the network, the number of other validators on the network, and many more factors.

4.3.3 Advantages and disadvantages

The main advantages of proof of stake are its energy efficiency and lower entry barriers for securing the network, therefore promoting further decentralization. Anyone can run a validator on most blockchains and does not need huge servers to solve puzzles like one would on a proof of work chain. A 51%-Attack is also less likely on a proof of stake network since it is much harder to monopolize a whole currency than computing power. To execute said attack, a miscreant would have to own more than 51% of all staked funds to, in turn, be able to forge more than half of the blocks on the blockchain.
Proof of stake, however, is not a saint and brings some disadvantages with it as well. For one, it is younger and significantly less battle-tested compared to proof of work. Moreover, it is more complex to implement, and it generally requires more technical knowledge to be a validator in a proof of stake network than a being a miner in a proof of work network. All in all, proof of stake does have its flaws, but according to many people and experts online, it seems to be a better option than proof of work for the time being, allowing for more transaction per second throughput, lower fees, and eco-friendly energy consumption.

4.4 Possible attacks

Some problems arise when bad actors make it onto the network. Since all nodes operate independently, there is no way to guarantee that they run the protocol as they should. A node could try to attack the network in favor of someone's personal gain or to wreak havoc. All of the attacks listed below are explained with the assumption that a miscreant called Alice is running a node. When discussing the following problems, we assume that the blockchain being talked about uses the proof of work consensus model for the sake of simplicity. The problems discussed are also valid in the proof of stake model.

4.4.1 Basic stealing

What if Alice, who found the solution to the next block, puts a transaction in the block that says to transfer funds from a random person to herself without the authorization of said random

person? This would not work because of digital signatures. When a node mines and broadcasts a new block, the nodes that receive that block verify it first. The miscreant, Alice, would not be able to produce the correct signature for that transaction since she does not own the private key to the account she wants to steal funds from. As a result, other nodes that receive that block would verify all transactions within it and conclude that the forged transaction is invalid, therefore rejecting the block and not adding it to their copy of the blockchain.

4.4.2 Denial-of-service attack

Let us consider another attack where no one tries to steal funds, but Alice dislikes someone else, whom we will call Bob, very much. Therefore, Alice does not include any transactions originating or going to Bob. One might think that this would cause Bob to be unable to participate in the network. This is not a fantastic attack either since a transaction purposely left out by a node will stay in the mempool and then be included in the next block mined by an honest node. It might pose a slight annoyance to Bob that he must sometimes wait longer for his transactions to be included on the blockchain, but he will still be able to participate in the network, and his funds remain safe and spendable.

4.4.3 Double-spend attack

The double-spend attack is the most infamous problem in digital currency and remains one of the most challenging problems to solve in a decentralized financial system. Let us assume that Bob is the owner of an e-commerce website where he sells a download link to software he made. Now let us assume that Alice wants to purchase this download link from Bob but does not want to pay for it. This is where Alice could try to launch a double-spend attack. First, Alice submits a transaction to the network, sending funds to Bob. She then waits for this transaction to be included in the next block, created by an honest node. Bob sees that he received funds on the blockchain and provides Alice with her download link. Next, assume that Alice can find the solution for the next block and can successfully mine it. She can propose ignoring the last block, where the funds were sent from Alice to Bob. Instead, she includes a transaction where she sends the same funds that were sent to Bob in the previous block, which Alice proposes to ignore, to herself. She would generate this new block, completely ignoring the last block, and therefore end up with a chain equally as long as the chain before. To reiterate, she is building on the version of the blockchain that existed without the previous block where she sends funds to Bob. Nodes decide which chain to build on based on how long it is; the longer chain is valid. But in this case, both chains are of equal length and, from a computer's perspective, equally valid. We, as humans, can say that the chain where Alice sends the funds to Bob is more morally valid since she has received the product she bought by sending that transaction. A computer, however, does not see a difference in validity between the block where Alice sends funds to Bob and the block where Alice sends the funds to herself. A computer cannot decide which chain it should build on and, therefore, would usually choose the one it detected first in the peer-to-peer network. However, this could be any of the two chains because of network latency and other factors.
How could Bob protect himself against this attack? Cryptography has nothing to say about this since all blocks and transactions mentioned are equally cryptographically valid. This is a purely consensus-based decision made by the network with no right or wrong. An exquisite and straightforward solution to this problem is for Bob to wait until the transaction where he receives funds has multiple confirmations. The certainty that a transaction ends up on the consensus blockchain, the longest chain that most nodes agree to be "the one", grows exponentially

in proportion to the number of confirmations a transaction has. After six confirmations, Bob knows there is virtually no chance he will be deceived by Alice and that the transaction where he receives his funds is not on the consensus blockchain. To recap, Bob waits until the transaction has six or more confirmations and only then provides Alice with her download link.

5 Public key cryptography

We have discussed the fundamentals of public key cryptography already. However, now it is time to look at how deriving public keys from private keys works and why it is virtually impossible to calculate the private key by just knowing the public key. A function like this, where it is easy to calculate the output but hard to calculate the input from the output, is called a *trapdoor function.*

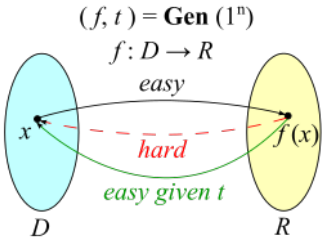


Figure 4: Trapdoor function

The first successful attempt at public key cryptography was in 1977 when the RSA algorithm and the Diffie-Hellman key exchange algorithm were introduced. These algorithms were revolutionary for their time and were based on the fact that it is elementary to multiply numbers but much harder to factorize them. This is especially true for large numbers. Multiplying the prime numbers 41 and 61 is trivial, especially for a computer $41 * 61 = 2501$. However, if one is given the task to factorize 2501, one might run into trouble. This fact is what powered the first public key cryptographic practices. In our modern day, we have an even better, more efficient type of cryptography: *Elliptic curve cryptography* [15] [16].

5.1 Elliptic curve cryptography

Elliptic curve cryptography, in short ECC, is one of the most powerful but least understood types of cryptography widely used today. ECC is a type of public key cryptography used in nearly all cryptocurrencies to derive public keys from private keys, create and validate digital signatures, and much more. Nevertheless, what is an elliptic curve, and how do its trapdoor capabilities work? Everyone that went to the first few years of school is familiar with multiplying and factoring. Sadly, that is not the case for elliptic curves, and to make things worse, the math around elliptic curves is not simple, and nor is explaining it. An elliptic curve is a plane curve over a finite field, rather than the real numbers as most of us know it from traditional functions. It consists of the points satisfying the following standard elliptic curve equation: $y^2 = x^3 + ax + b$.

[15] Sullivan, Nick (2013) - A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography - https://arstechnica.com/information-technology/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/ (22.11.2022)

[16] Wikipedia contributors (2022) - Elliptic-curve cryptography - https://en.wikipedia.org/wiki/Elliptic-curve_cryptography

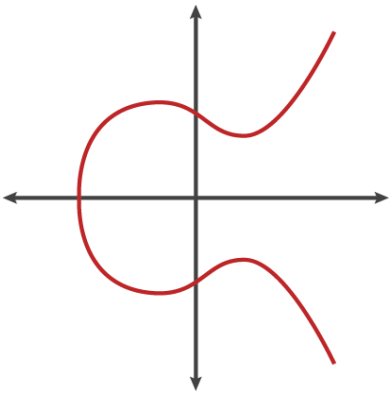


Figure 5: Simplified elliptic curve

The graphed-out elliptic curve in Figure 5 has some interesting properties. For one, it is horizontally symmetric. This means that any point on the curve can be mirrored on the x-axis and still be on the curve, on the opposite side. Another less obvious property is that any non-vertical line will cross the curve in a maximum of three points. Let us explore how these two properties can be used to wind up with an excellent trapdoor function.

5.1.1 Basic trapdoor mechanism

Let us use our simplified elliptic curve c above to illustrate how we can develop a viable trapdoor mechanism. Let us start with two points, A and B, and draw a straight line $\overline{AB}$ through them. We will end up with a third point that is neither A nor B where $\overline{AB}$ intersects with our curve c. Then, mirror this intersection at the x-axis to end with point C. Next, draw another straight line $\overline{AC}$. $\overline{AC}$ will intersect our curve c at another point which is neither A or C. Mirror this point again at the x-axis and end with point D. Next, repeat the process and draw another straight line $\overline{AD}$. $\overline{AD}$ will intersect again without curve c at another point which is neither A or D. Mirror this point at the x-axis and end with our final point E.

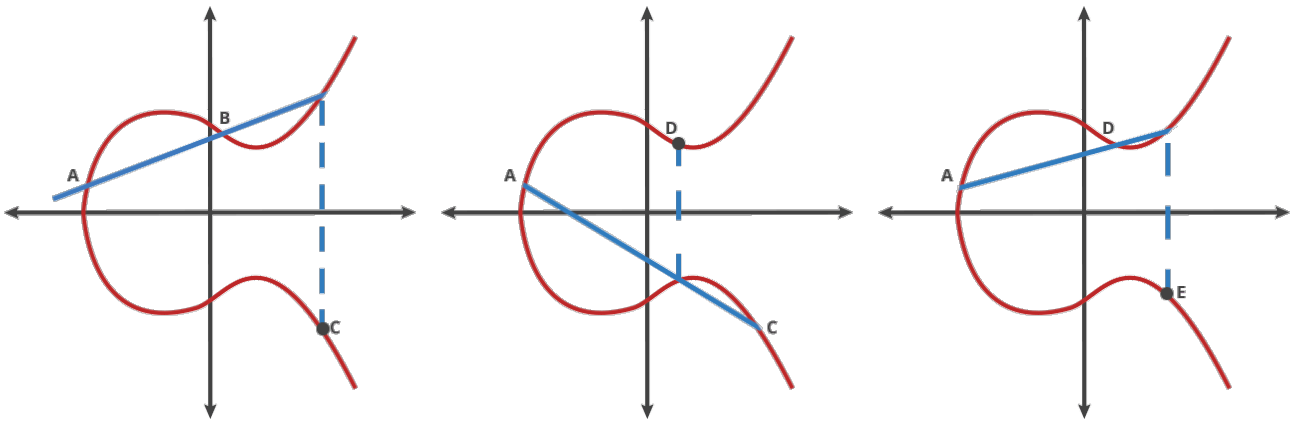


Figure 6: ECC trapdoor process

Let us assume that one does this process for a point A "dotted" with itself n times to arrive at a final point Z. It turns out that calculating n when provided with A and Z is extremely difficult. Similarly, imagine someone playing a game of billiards and hitting the ball based on a set of rules. After a few hits, someone walks in and sees the ball in its final position. It is very hard for them to determine the number of hits the player made to the ball, even if

the spectator knew all the rules, the starting position, and the end position. They can only determine the number of hits after running through the whole game themselves until the ball gets to the same point. More specifically, to derive point Z from point A when knowing n does not require someone to repeat this dotting process n times. This can be done more efficiently using elliptic curve scalar multiplication and exponentiation by squaring. So, in conclusion, ECC is a trapdoor mechanism because calculating Z from A is easy given n. However, one must calculate the output for every integer $1 - n$ until the output matches Z to find n. This requires an inscrutable amount of computing power if n is large enough. As one might guess, n is our private key, while A and Z together are our public key.

5.1.2 Public key size

As mentioned before, elliptic curves span over a finite field. The size of this finite field dictates the size of our public keys. As one might be able to imagine, repeating the dotting process along an elliptic curve might produce some points that have an unbelievably huge x value. This is why we employ a concept called the maximum value m. m is simply a maximum for Z that we define before calculating Z. We can treat this number like the numbers on an analogue clock. Any point with an x value higher than m gets "wrapped around" to a point with an x value below m. m serves as the key size for our public key since no public key can be larger than m.

5.1.3 Cryptographic curves

The security of an elliptic curve depends on its a and b parameters. Cryptographers have come up with parameters that make elliptic curves especially secure. The most common m maximum value is 2^{256}, meaning that public keys generally are 256 bits long. Let us look at some of the curves used by Bitcoin, called Secp256k1 and the Edwards25519 curve, used in cryptocurrencies like Solana and many others. Secp256k1 is a standard elliptic curve and employs the parameters $a = 0$ and $b = 7$. Its formula therefore is $y^2 = x^3 + 7$. The Edwards25519 curve however is not a standard elliptic curve, but a *montgomery elliptic curve*. Its formula is $y^2 = x^3 + 486662x^2 + x$.

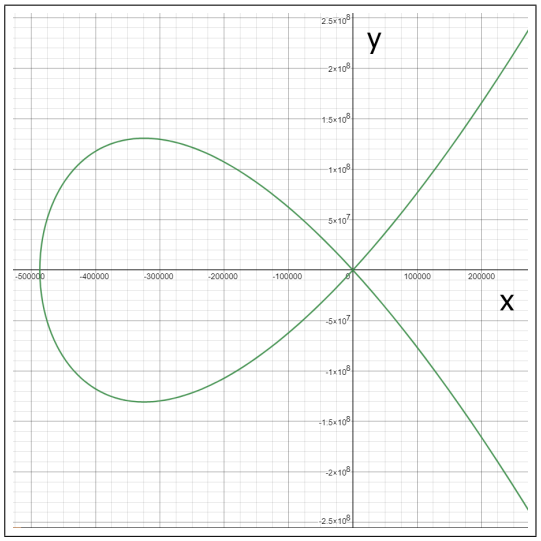


Figure 7: Edwards25519 (own work)

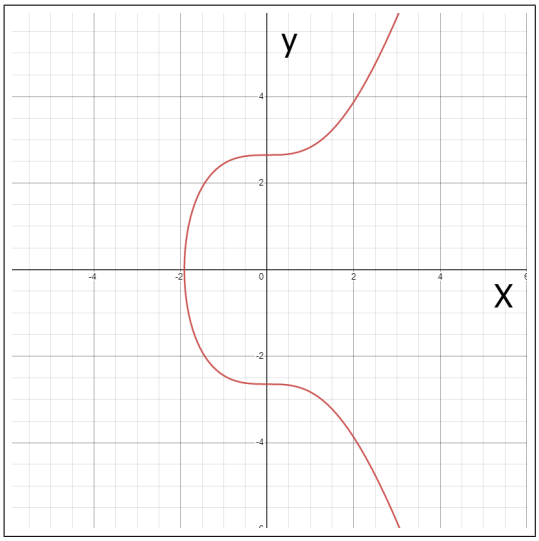


Figure 8: Secp256k1 (own work)

6 My own cryptocurrency

As mentioned in the introduction, my goal was also to develop my own cryptocurrency and blockchain as proof of concept. I am a software developer with around two years of real-world experience in networking, algorithms, and backend infrastructure and about ten months of experience working with blockchain technologies and building applications on top of other blockchains. Building a blockchain seemed like a reasonably difficult task to me.

6.1 Design decisions

Since I am not a senior developer, I decided to make some design decisions, trading off efficiency and energy consumption for simplicity. First off, I decided to write this blockchain in the programming language Golang. Golang is a compiled backend language made by Google to handle high workloads but is also simple to write in. I decided to go with Golang since it is the language I am most familiar with, and I deemed it good enough, especially for the networking part of a blockchain.
As for the consensus mechanism, I decided to go with the proof of work model. It is secure and easy to implement but not very energy efficient. Since my currency is not meant to be run in production by thousands of servers but rather is a proof of concept, I decided that energy efficiency and high throughput are not too important. As for the hash function, I went with SHA-256, a well-known hash function deemed the gold standard. It outputs 256 bits from an input of any length, perfect for our job. It is also used by Bitcoin and virtually every other cryptocurrency on the market.
To derive public keys and sign messages, I decided to use elliptic curve cryptography and, more specifically, the Edwards25519 curve. This allows for a 256 bit public keys and so much security that it is almost impossible for any computer or amount of computers to crack someone's wallet.
As for block structure, I went with the absolute minimum.

```
You, 2 days ago | 1 author (You)
type Block struct {
    MagicNumber        uint64        `json:"magicNumber"`
    TransactionCounter uint64        `json:"transactionCounter"`
    Transactions       []Transaction `json:"transactions"`
    Header             BlockHeader   `json:"header"`
}

You, 2 days ago | 1 author (You)
type BlockHeader struct {
    Version      string    `json:"version"`
    PreviousHash Hash      `json:"previousHash"`
    Miner        PublicKey `json:"publicKey"`
    Solution     []byte    `json:"solution"`
    Hash         Hash      `json:"hash"`
    Time         int64     `json:"time"`
}
```

Figure 9: Block structure of my cryptocurrency (own work)

The blockchain I built is an array of the Block structure defined above.

I made a similar approach for transactions.

```
You, 2 days ago | 1 author (You)
type Transaction struct {
    IsCoinbase bool            `json:"isCoinbase"`
    Data       TransactionData `json:"data"`
    Signature  Signature       `json:"signature"`
}

You, 2 days ago | 1 author (You)
type TransactionData struct {
    Sender   PublicKey `json:"publicKey"`
    Receiver PublicKey `json:"receiver"`
    Amount   uint64    `json:"amount"`
}
```

Figure 10: Transaction structure of my cryptocurrency (own work)

Public keys, private keys and signatures are just byte arrays with predefined lengths.

```
const (
    PublicKeyLength = 32
    SignatureLength = 64
)

type PublicKey [PublicKeyLength]byte
type PrivateKey []byte
type Signature [SignatureLength]byte
```

Figure 11: Transaction structure of my cryptocurrency (own work)

In general, I held myself pretty close to the standards of Bitcoin, with a lot less attention to detail. To make a comparison, the current version of my blockchain is around 800 lines of code, while the Bitcoin node code is over 2M lines of code.

6.2 Obstacles and coding process

The coding process was callous, especially the networking part. After a lot of trial and error and a few hours invested into the project, however, I managed to get everything working in my test cases. Fortunately, I was able to use third-party packages made by other developers to manage highly cryptographic parts such as the signing of transactions, the derivation of public keys, and the implementation of the SHA-256 hash function. What packages I used precisely can be looked up in the "go.mod" file in my source code. There were no specific significant obstacles I had to overcome since I decided to write my blockchain when I was done with writing the rest of this paper, and I had acquired quite some knowledge about blockchains until then.

6.3 Security

Of course, security is the main concern one might have when making a cryptocurrency. Security also concerned me, but I was not scared about losing any real funds since no value is pegged to my cryptocurrency. The network is currently running on three miner nodes, each with two CPU cores, which I set up myself. Gaining 51% of the computing power in the network should not be hard for any modern computer, and I highly encourage technically interested readers to try to break my network. A 51% attack is currently the biggest threat to the network, but

I think there might be some security flaws in the code that I wrote that could be exploited. Again, I encourage any interested reader versed in Golang or other programming languages to look at my code and try to find any exploits that might exist.

6.4 Result

I am very content with how this little blockchain turned out. It might not be the most efficient blockchain by any means, but it works, and I was able to verify the knowledge I acquired by writing this essay. I can recommend to anyone interested in blockchain and software development to try writing their own. It is a very fun project and something interesting to have on your coding resume. The code is publicly available under https://github.com/chrigeeel/matura-blockchain and one can interact with the blockchain on https://matura.chrigeeel.dev/blochchain.

7 Conclusion

There are many ways a cryptocurrency can function and plentiful decisions a cryptocurrency developer can make to improve a blockchain. We are still in the early stages of development, and with time, more incredible things will be achieved, and we may arrive at an almost perfect cryptocurrency. There are still many obstacles to overcome, and the future can look either grim or bright, depending on how one looks at it. A big obstacle today is mass adoption. How do we get as many people as possible to trust cryptocurrencies, and how do we design blockchain networks so they can eventually handle more than 8B daily users and millions of events per second? Even if all those obstacles are overcome, what stops a government from making cryptocurrencies illegal? Sure, the networks will stay online because of their decentralized manner, but if one goes to prison for using them, I do not think that many people would enjoy using cryptocurrencies. With the speed at which things are moving, current technologies might be outdated within just a few decades or even years. However, we are laying a huge stepping stone for our future generation of developers with our work. Cryptocurrency and blockchain technologies are new, fun, and exciting but have yet to be involved in our day-to-day lives. For some, they might instantiate a feeling of a forthcoming revolution, while for others, they are just another bubble that will burst within a few years. No one can tell what truly is upon us and what the next revolution, if any, of our financial system will be. A multitude of people, including myself, are eager to find out if cryptocurrencies really are the future, but only time will tell.

8 Bibliography

8.1 Sources

- Antonopoulos, Andreas (2017) - Chapter 4. Keys, Addresses - https://www.oreilly.com/library/view/mastering-bitcoin-2nd/9781491954379/ch04.html (07.11.2022)
- Bitcoin Wiki contributors (2019) - Block - https://en.bitcoin.it/wiki/Block (07.11.2022)
- Bitcoin Wiki contributors (2021) - Block hashing algorithm - https://en.bitcoin.it/wiki/Block_hashing_algorithm (07.11.2022)

- Bitcoin Wiki contributors (2018) - Network - https://en.bitcoin.it/wiki/Network (07.11.2022)
- Bitcoin Wiki contributors (2021) - Transaction - https://en.bitcoin.it/wiki/Transaction (07.11.2022)
- Bitcoin Wiki contributors (2022) - Weaknesses - https://en.bitcoin.it/wiki/Weaknesses (07.11.2022)
- Nakamoto, Satoshi (2008) - Bitcoin: A Peer-to-Peer Electronic Cash System - https://bitcoin.org/bitcoin.pdf (07.11.2022)
- Narazanan, Arvind and Bonneau, Joseph and Felten, Edward and Miller, Andrew and Goldfeder, Steven (2016) - Bitcoin and cryptocurrency technologies: a comprehensive introduction - Princeton: Princeton University Press
- Orenes-Lerma, Linda (2022) - How does a blockchain transaction work - https://www.ledger.com/academy/how-does-a-blockchain-transaction-work (07.11.2022)
- Statista Inc. (2022) - Venezuela: Inflation rate from 1985 to 2023 - https://www.statista.com/statistics/371895 (07.11.2022)
- Statista Inc. (2022) - Year-over-year change in consumer prices in Russia from January 10 to October 31, 2022 - https://www.statista.com/statistics/1298843 (07.11.2022)
- Sullivan, Nick (2013) - A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography - https://arstechnica.com/information-technology/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/ (22.11.2022)
- Wikipedia contributors (2022) - Consensus (computer science) - https://en.wikipedia.org/wiki/Consensus_(computer_science) (07.11.2022)
- Wikipedia contributors (2022) - Cryptographic hash function - https://en.wikipedia.org/wiki/Cryptographic_hash_function (07.11.2022)
- Wikipedia contributors (2022) - Elliptic-curve cryptography - https://en.wikipedia.org/wiki/Elliptic-curve_cryptography
- Wikipedia contributors (2022) - Hash function - https://en.wikipedia.org/wiki/Hash_function (07.11.2022)

8.2 Graphic sources

- Figure 3: Hall, Stephen (2021) - What is a bitcoin seed phrase and how does it work? - https://unchained.com/blog/what-is-a-bitcoin-seed-phrase/ (22.11.2022)
- Figure 4: IkamusumeFan (2015) - The idea of trapdoor permutation https://commons.wikimedia.org/wiki/File:Trapdoor_permutation.svg (22.11.2022)
- Figure 5: Sullivan, Nick (2013) - A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography - https://arstechnica.com/information-technology/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/ (22.11.2022)

- Figure 6: Sullivan, Nick (2013) - A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography - https://arstechnica.com/information-technology/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/ (22.11.2022)