



# Blockchain Technology

## Advanced

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Foreword</b>	<b>4</b>
<b>What is a Blockchain?</b>	<b>5</b>
Blockchain as a Data Structure	5
The Blockchain is a Data Structure	5
Arrays	5
Linked Lists	6
Blockchains	7
Summary	9
A Protocol to Transfer Value	10
Network Protocols	11
What Are the Rules?	13
The Great Innovation Introduced with Bitcoin	14
Summary	15
Guaranteed Execution with Smart Contracts	16
The Promise of Smart Contracts	16
dApps	17
Smart Contract Platform	18
Are They Really Trustless?	19
Summary	21
<b>How Does a Blockchain Work?</b>	<b>23</b>
The Elements of a Blockchain	23
Hash Functions	23
Public-Key Cryptography	25
A Peer-to-Peer Network	25
Consensus Mechanism	26
Mining	27
Demo	28
Hash Functions	29
Summary	31
Public-Key Cryptography	32
Elliptic Curve Cryptography	33
Private Key	38
Public Key	38

Address	40
Digital Signature	41
Summary	41
The Peer-to-Peer (P2P) Network	42
Variations	43
Incentive	44
Secure Nodes and Super Nodes	45
Summary	45
Consensus Mechanisms	46
Proof of Work - PoW	47
The Longest Chain Rule	49
Proof of Stake - PoS	50
Comparing POW and POS	51
Summary	53
Mining	54
What Does the Miner Do?	54
Finding a Nonce	55
Block Time	59
Hardware	59
How Does This Protect the Network?	59
Summary	60
<b>Wallets</b>	<b>61</b>
Your Mnemonic Phrase	61
The Different Types of Wallets	62
Hosted Web Wallets	62
Non-Hosted Web Wallets	64
Desktop and Mobile Wallets	65
Paper Wallets	66
Hardware Wallets	66
How Does a Hardware Wallet Work?	67
Summary	69
<b>Transactions</b>	<b>70</b>
The UTXO Model	70
Bob Receives His First Transaction	72
Bob Sends His First Transaction	72
Another UTXO	73
Spending Two UTXOs at Once	73
Summary	74

<b>Block Explorer Continued</b>	<b>75</b>
What You Will Find in a Block Explorer	75
The Genesis Block	76
The First Bitcoin Transaction	78
Summary	79
<b>Atomic Swaps</b>	<b>80</b>
The Technology Behind Atomic Swaps	80
The Process	81
Atomic Swaps Today	83
Summary	84
<b>Privacy on the Blockchain</b>	<b>85</b>
Why Privacy?	85
Change Addresses	85
Coin Mixing	86
Ring Signatures	87
Zero-Knowledge Proofs	88
Summary	89
<b>Attacks</b>	<b>91</b>
The Byzantine Generals Problem	91
DDOS Attack	93
Sybil Attack	95
51% Attack	96
Summary	97
<b>Summary Advanced Level</b>	<b>99</b>
What is a Blockchain?	99
How Does a Blockchain Work?	101
Wallets	105
Transactions	106
Privacy on the Blockchain	108
Attacks on Blockchain	109
<b>Final Remarks</b>	<b>111</b>

# Foreword

We designed the Zen Academy to be the go-to place for education on blockchain, cryptocurrency, and online privacy. It doesn't matter if you are interested in learning about the topics out of curiosity or need to learn about them because your company is exploring blockchain technology. No matter your comfort level or interest: this project is for you.

In the first chapter of this section, we talk about how blockchains store transactions, why this allows you to transfer value, and what smart contracts are.

Next, we will bring you up to speed on the different elements that make blockchains work.

In order to receive and send cryptocurrencies, you will use a wallet. In the third chapter, we will show you the different types of wallets.

After we covered wallets, the interfaces you use to create transactions, we will take a closer look at what transactions are and how they work.

While most cryptocurrencies are not private by default, you have the option to create truly private transactions. This chapter introduces the different methods that achieve privacy on the blockchain. Blockchains are a secure way to store data but there are ways to attack a blockchain. We show you the most common attacks and how their risk is mitigated.

Lastly, we summarize the entire level.

**We hope you have a good time reading!**

# What is a Blockchain?

We would like to look at blockchain from three different perspectives in the first section of our advanced technology section.

First, from a computer scientists perspective looking at blockchain as a data structure.

Second, from a more philosophical perspective; reviewing the implications of blockchain technology used for value transfer.

Lastly, we discuss smart contracts, a technology that blockchain technology enables. You can summarize Smart contracts as digital contracts with guaranteed execution. You might have heard the term programmable money in the context of cryptocurrencies before. Smart contracts are the technology that enables this programmability.

## Blockchain as a Data Structure

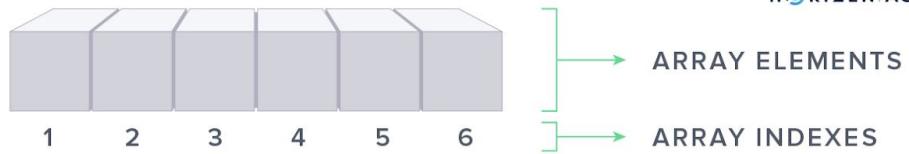
One can explain Blockchain technology in many different ways. Looking at blockchain through the lens of cryptocurrencies has been the dominant narrative until recently. Bitcoin is the first thing most people will associate with blockchain technology but storing cryptocurrency transactions is only one use case out of many. We would like to take a step back and look at blockchain in more general terms from the perspective of a computer scientist in this article.

## The Blockchain is a Data Structure

A data structure is a way to store, organize, and manage data. A data structure enables you to access, add, modify and search the data contained within it. Some of the most common and basic data structures include *arrays* and *linked lists*.

### Arrays

An array is a number of enumerated elements. These elements can be numbers, letters, words or even entire files. The indices allow you to access any element individually, so if you want to modify an entry in an array and you know its location, you have *instant access*.



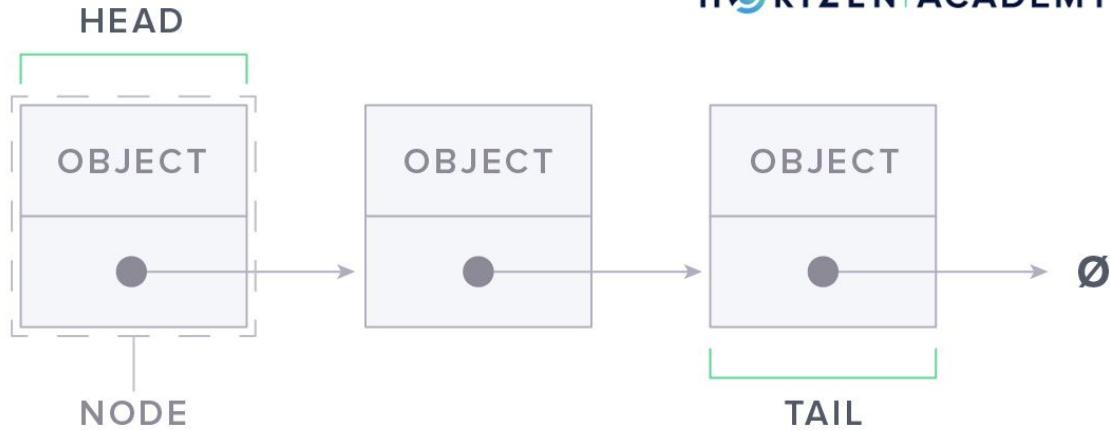
## One-dimensional array with six elements

### Linked Lists

*Nodes* are the data elements in a linked list. A node comprises at least one data object and a pointer to the next element. This pointer's function is to tell your computer where to find the next element of the list.

If you look at the first piece of data on the list and wish to access the second one, you will look at the pointer that directs you to the next node. It is easier to add data to a linked list through expanding it by an extra node than it is to add data to an array by increasing the number of elements. What you don't have with a linked list is *instant access*.

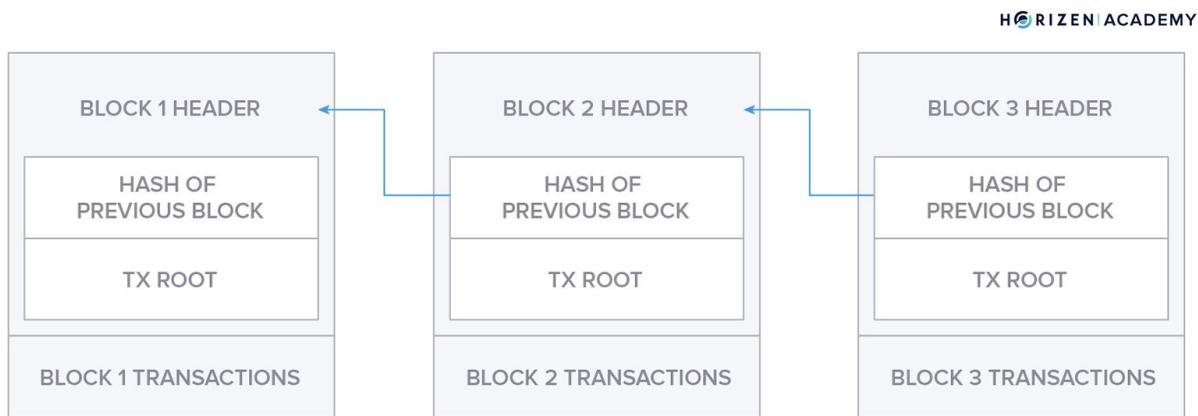
If you are searching for a specific piece of data in your linked list you will look at the first *node*, the *head* of the linked list. If it is not the element you were looking for, you follow the pointer, that will lead you to the next node. If this node does not contain the data you were looking for either, you continue by following the links throughout all nodes until you find the desired data.



## Linked List with three nodes

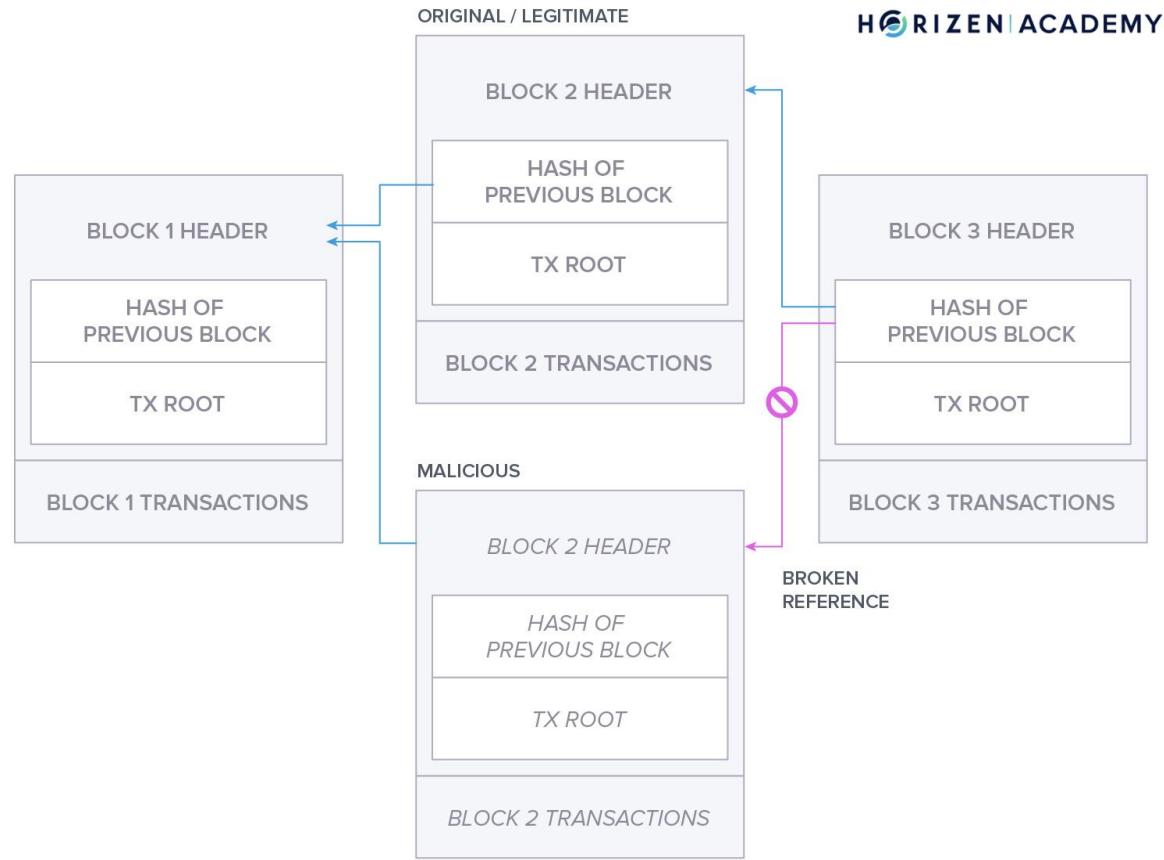
### Blockchains

The blockchain is like a linked list in the context of data structures. The blockchain also separates the data into containers - the *blocks*. The *blocks* are quite analogous to *nodes* in a linked list. Each *block* contains a *reference*, which is the hash of the preceding block. This serves as a link to the preceding block and establishes the order throughout the chain of blocks.



The key difference between a blockchain and a linked list is that each reference in a blockchain is cryptographically secured. You may hear the term *append-only data structure* describing

blockchains. This means you can only add data to a blockchain by appending it to the front. The secured links are constantly checked for validity. If you were to insert a malicious block in the middle of a blockchain, e.g. between Block 1 and 3 in the graphic below, you could include a reference to its predecessor (Block 1), but you cannot make the next block (3) reference your maliciously inserted block.



Each new block built on top of an existing block is commonly known as a *confirmation*. The older a block gets, the more confirmations it will have. Each confirmation makes tampering with the data in a block more difficult. Block 2 in the graphic below has one confirmation. To tamper with its data, you would have to recreate one valid block including a new valid reference. With each confirmation, you would have to recreate an additional block. This means the older the block, the more certain you can be that no changes to the block will occur.

The references between blocks don't just depend on the order of blocks, but also on the data contained within each block. It is not possible to add or delete data from a block in the blockchain. This property is the foundation of the trust people put in the data stored in a blockchain. The data

structure of a blockchain is naturally *tamper-evident*. Any change of data breaks the references of all subsequent blocks.

While it is not possible to change or delete data, it is easy to add data in a new block to the chain. For example, you could add a new transaction on a cryptocurrency blockchain. The transaction is easy to verify because all the preceding transactions recorded on the network are immutable - this makes the verification of transactions simple. When Address Y wants to spend amount X, it must have a balance of at least amount X.

Cryptocurrencies are only one specific use case of blockchain technology. The blockchain is rapidly becoming a viable option to track supply chains, do fleet management, and more.

## Summary

The blockchain is a method to store data in the context of computer science. The elements of a blockchain - its blocks - are cryptographically linked. It is not feasible to change data after it has been recorded to the block. This is why there is value in blockchain. It is an immutable ledger that stores data reliably in a trustless environment.

We would like to end this section with a tweet that distills the implications of blockchain technology in the context of data structures.

Jameson Lopp @lopp

"Blockchains don't guarantee truth; they preserve truth & lies from later alteration, allowing one to securely analyze them and be more confident in uncovering the lies. Typical computers are computational etch-a-sketch, while blockchains are computational amber." - [@NickSzabo4](#)

1,240 1:34 AM - Nov 5, 2018

443 people are talking about this

[Link](#)

## A Protocol to Transfer Value

Money is one of the oldest "technologies" of humankind - a technology to communicate value. There have been many forms of money throughout several centuries and even millennia - from barter to bones, feathers to precious metals, gold-backed currencies to today's fiat currencies. Blockchain might be the technological advance that was needed to move to the next chapter in the history of monies: cryptocurrencies.

In this article, we will look at blockchain as a settlement network for financial transactions and the rules and conventions that make blockchain work.

*"[The] Bitcoin protocol and network today is [a] value transfer network. Beyond that, it is a core, backbone security service securing contracts, physical and digital property, equities, bonds, robot AI and an enormous wave of applications which have not yet been conceived."* - Jeff Garzik, Bitcoin Core Developer

The protocol is the set of rules and standards that governs the overall workings of the blockchain.

[Techopedia's definition](#) of a network protocol goes as follows:

*"Network protocols are formal standards and policies comprised of rules, procedures and formats that define communication between two or more devices over a network. Network protocols govern the end-to-end processes of timely, secure and managed data or network communication."*

The inception of the blockchain defines this set of rules, procedures, and formats. These rules can only be altered if a majority of network participants decide to do so. Public blockchains require consensus building, which in turn requires careful consideration of each change proposed. This desirable feature helps to create a robust foundation of trust for the users.

## Network Protocols

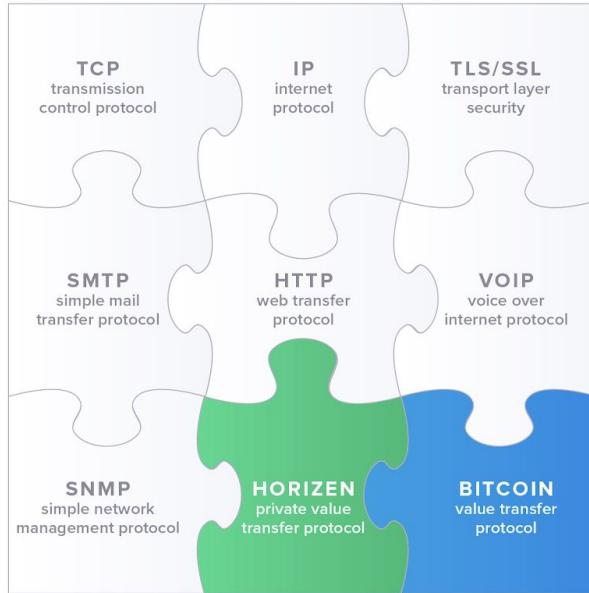
You can distinguish between several broad types of networking protocols. There are:

- Network communication protocols like TCP/IP (transmission control protocol/internet protocol)
- Network security protocols like HTTPS and SSL (hypertext transfer protocol secure/secure sockets layer)
- Network management protocols such as SNMP and ICMP (simple network management protocol/internet control message protocol) and
- Value transfer protocols such as Horizen or Bitcoin

Below you see a collection of protocols, many of which you use every day. For example, The TCP/IP Protocol specifies how to break data down into packets, address the packets, and route them through the network to their final destination. You are using the TCP/IP standard any time you go online to communicate with different servers to receive the information you want to access.

SSL (Secure Sockets Layer) is a standard protocol used to establish encrypted links between a web server and a client (like your computer) in online communication. For example, when sending your payment details to an online shop you use SSL to establish an encrypted connection before you actually send your credit card information across the internet.

The Simple Network Management Protocol (SNMP) is a set of protocols supported by network devices such as routers, servers or printers. SNMP takes care of the different devices on a network interoperating seamlessly.



Throughout the history of the internet, people have defined standards for the exchange of various types of data. It's important to note that it was not always the best protocol that gained mass adoption. People are unlikely to switch to a different protocol once they begin using it and building upon it. Developers become attracted to the increasing number of libraries and tools available for these protocols. This cycle repeats until a protocol eventually becomes the standard for a given use case.

Bitcoin is in a good position to become the agreed-upon standard for the storage and transfer of value across the internet. Bitcoin is a convention that specifies rules, procedures, and formats for the transfer of money online without intermediaries. Horizen aims to become the standard for private value and data transfer in the decentralized web 3.0.

People like to point out that the throughput (mostly referring to the number of transactions per second) blockchains can handle is not sufficient for mass adoption (yet). We believe that using [sidechains](#) is a viable way to overcome those growing pains.

Scalability is a discussion in and of its own. We are confident in saying that a distributed ledger of some sort will become the agreed-upon standard to transfer value in the future. Bitcoin has the greatest adoption of all cryptocurrencies today. Many projects and companies have started building upon the Bitcoin protocol. In the past, this kind of adoption was the first indicator of the success of a

protocol. We at Horizen are building a blockchain protocol that offers privacy, is scalable and usable by developers, especially with the release of our sidechain implementation and the [HDE - the Horizen Developer Environment](#).

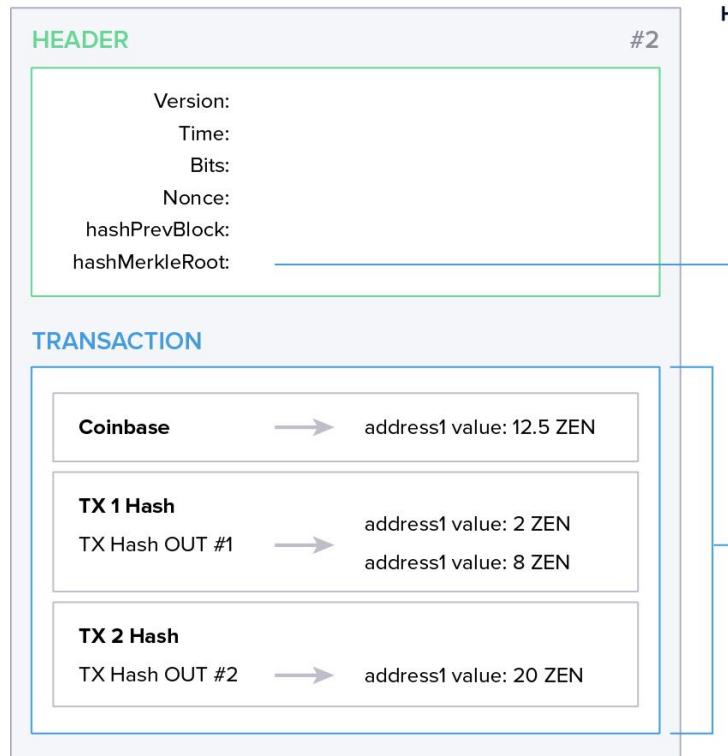
## What Are the Rules?

A protocol is a set of rules and conventions used for a given purpose. Let us take a look at what some of the conventions and rules for a blockchain look like.

One subset of the rules regards maintaining the ledger. Each node keeps a copy of the blockchain and verifies every transaction it receives. Once a transaction becomes verified it is then saved in the *mempool* (short for memory pool) with all the other transactions that are not yet included in a block. When a node receives a new block from its peers, it checks the validity of the block. If it is valid, then it is added to the local copy of the blockchain and all transactions included in the block are removed from the mempool. The mempool will only ever contain unconfirmed transactions.

Another subset of rules is concerned with the structure of a valid block. Remember that a block is just a container for data as we explained in our last article. A block has a block header that contains information about the version of the client it was created with, a reference to its preceding block (its hash), a summary of all transactions that are contained in the block (the *Merkle root*), a timestamp and some other metadata.

Following the block header, the block contains all the transactions that were included.



Now there needs to be a set of rules describing what an individual transaction must look like. We will talk about transactions in detail in a dedicated article but for now, it should suffice to know that a transaction needs to include the sender, the recipient, the amount to transfer, and a digital signature. The digital signature is a way to authorize the spending of your funds. The private key of a user creates this digital signature. We explain this process further in our article on public-key cryptography.

## The Great Innovation Introduced with Bitcoin

*Game theory* is the study of mathematical models of strategic interaction between rational decision-makers. Miners, nodes, and users of cryptocurrencies are the (hopefully ;) rational decision-makers. Here rational means each person is trying to maximize their personal benefit. When Satoshi Nakamoto published the bitcoin whitepaper, he introduced a system in which each participant is economically incentivized to follow the rules of the protocol. In other words, it is more profitable to play according to the rules, than to break them - we say Bitcoin is *incentive compatible*. This seems to hold true, even 10 years after the inception of the protocol. One cannot overestimate the foresight and thought that went into building such a system.

It can be difficult to see why we consider all this to be a groundbreaking innovation when we look at the different parts of a blockchain. It is true that most concepts that comprise Bitcoin have been around for years. Game theory is arguably the most important part of a public blockchain protocol. The design of the incentive-structure is the reason why Bitcoin was the first cryptocurrency that survived - few know that there have been many attempts to build a similar system before. They were either highly centralized or didn't provide the right incentives for their participants.

## Summary

The protocol of a blockchain is a set of rules that all participants must follow. It governs how to communicate data across the distributed network, what blocks and transactions have to look like and how participants are incentivized so that abiding the protocol is the most profitable strategy.

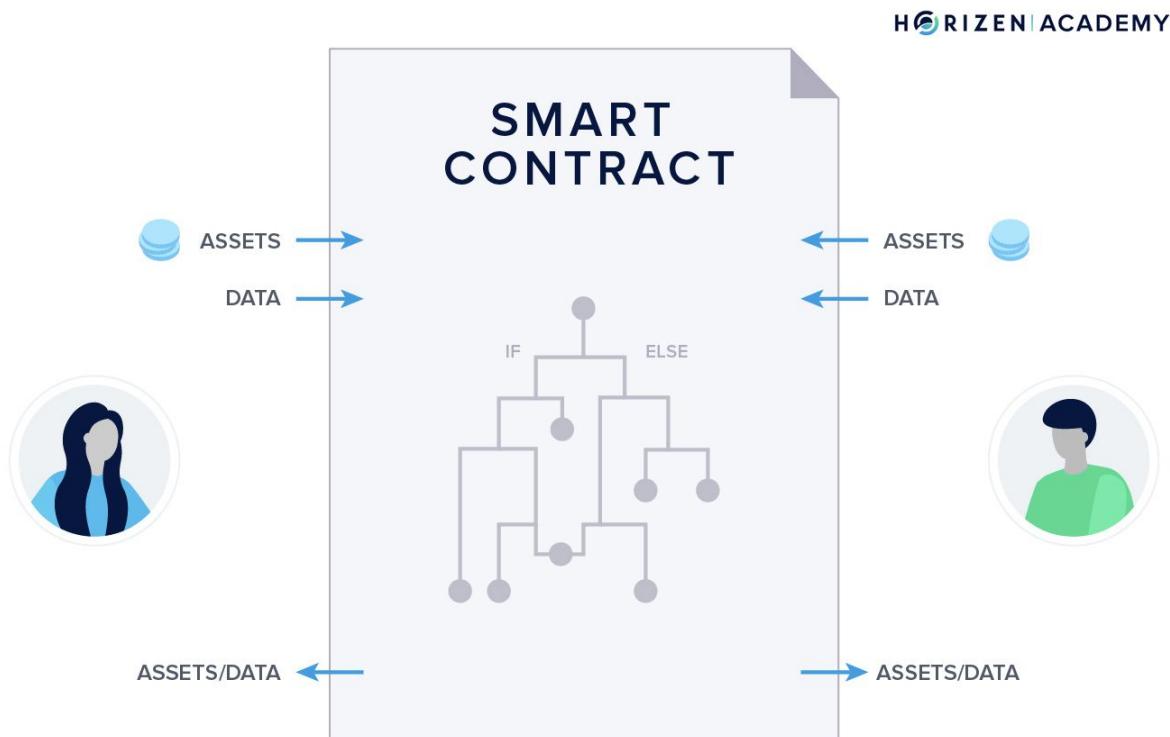
I would like to end this article the same way it began, with a quote from Jeff Garzik, one of Bitcoins better-known developers.

"Do not try to stuff every feature into the Bitcoin protocol. Let it do what it does best. Build systems on top of Bitcoin which use its strengths... Putting all the world's coffee transactions, and all the world's stock trades, and all the world's Internet of Things device samplings, on the Bitcoin blockchain seems misguided" - Jeff Garzik

## Guaranteed Execution with Smart Contracts

We have looked at blockchain in two different ways. First as a data structure and second as a protocol to transfer value. For the last article of this chapter about what a blockchain is we want to talk about *smart contracts*. Besides AI, IoT, and blockchain, smart contracts have been one of the hottest topics over the last two years in the tech world. They are software on the blockchain.

A blockchain can not only host simple data like transactions, but also small programs. We call these programs smart contracts. A contract, in general, is an agreement between parties that binds them to something happening in the future. The "smart" comes from the automatic execution of these digital contracts. Simply speaking they consist of many "if, then" statements that are written in and enforced by code. The contract executes automatically if the contract conditions are met.



## The Promise of Smart Contracts

Smart contracts promise to reduce the need for middlemen, such as lawyers or notaries, and thereby reduce the cost of transacting. Disposing of middlemen can also save a lot of time. You don't

have to wait for opening hours or through banking holidays. Smart contracts can not only be used to govern the transfer of digital assets such as cryptocurrencies, but they could govern many other types of value transfer, such as trading of shares and other traditional financial instruments or even transactions involving physical property (like real estate).

A landlord and a tenant could use a smart contract to govern a rental agreement. The smart contract could automatically lock the tenant out of the apartment if the tenant fails to pay rent. The if-then statement, in this case, could look like this:

*If the contract address receives amount X by the 3rd of each month from address Y, then grant Y access to the apartment. If payment fails for 2 consecutive months, then revoke the right of Y to unlock the apartment.*

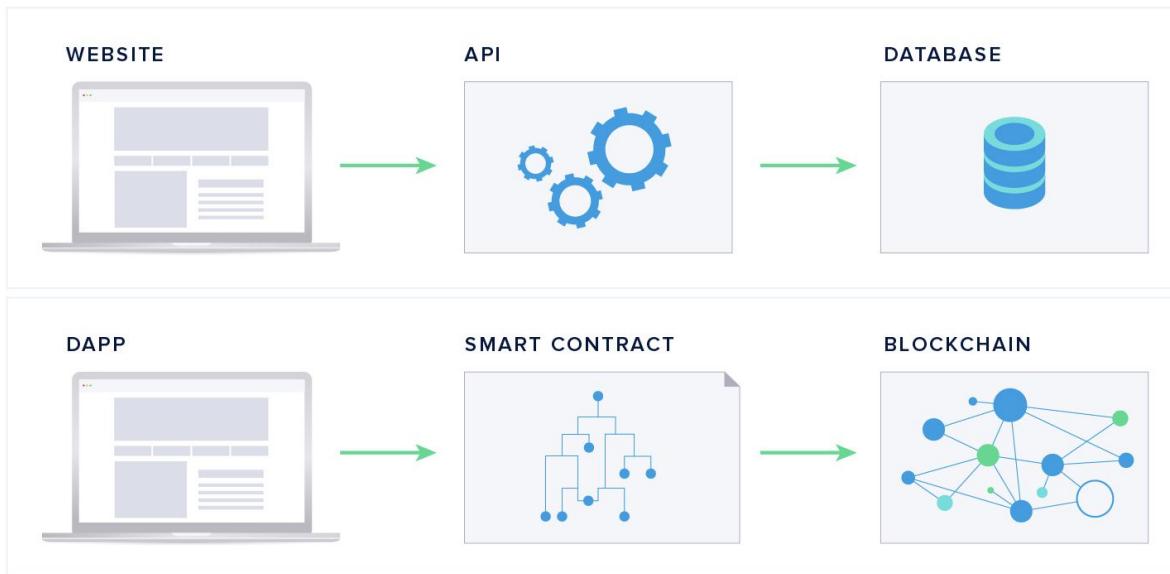
Another potential use case for smart contracts would be a decentralized eBay-like platform. One can construct a smart contract with an expiry date defining the time period of the auction and a starting bid. The highest bidder would receive the item at the end of the auction period in exchange for their money. The losing bidders would automatically be refunded by the smart contract. Such a decentralized version of eBay would be considered a *dApp* or decentralized App.

## dApps

A decentralized application or dApp is a more sophisticated use case for smart contracts. Apps and websites usually use APIs (application programming interfaces) to communicate with their underlying database and other services. A well-written API makes it easier for developers to offer a service by defining the communication between the various components of a system.

Dapps use smart contracts to communicate with their underlying blockchain. Imagine a future with smart contract libraries that hold a large number of template contracts that can be used for various purposes. We are already seeing such developments with smart contract platforms like Ethereum.

## TRADITIONAL WEB V. DECENTRALIZED APP



### Smart Contract Platform

There are many smart contract platforms out there besides Ethereum, which is the most popular one today and has the most developer activity around it. Other platforms include Lisk, NEM and Hyperledger, a modified version of Ethereum designed for corporate use.

Few people know that Bitcoin-based blockchains also allow for the deployment of smart contracts. Bitcoin has a built-in scripting language called *Script*. It is a rudimentary language compared to *Solidity*, which is used to write smart contracts on Ethereum. Solidity is a *Turing-complete* programming language, that enables more complex contracts compared to Bitcoins Script. The cost of being more complex comes at being harder to write, analyze and secure.

Security in the context of smart contracts means considering every possible way in which a contract could execute and to account for each of these scenarios. Smart contracts on Bitcoin-based blockchains (such as Horizen) written in Script allow for less complexity compared to ones written in Solidity on Ethereum. This limits their potential use cases but makes the possible states of the contract (or program) easier to enumerate, examine, and account for, resulting in easier to secure contracts.

It is interesting to note, that the most commonly used smart contract templates on Ethereum, namely the *ERC-20* standard used to issue tokens, does not require Turing completeness. Almost all tokens deployed on the Ethereum blockchain are using this standard, as it allows for easy integration with different wallets and exchanges.

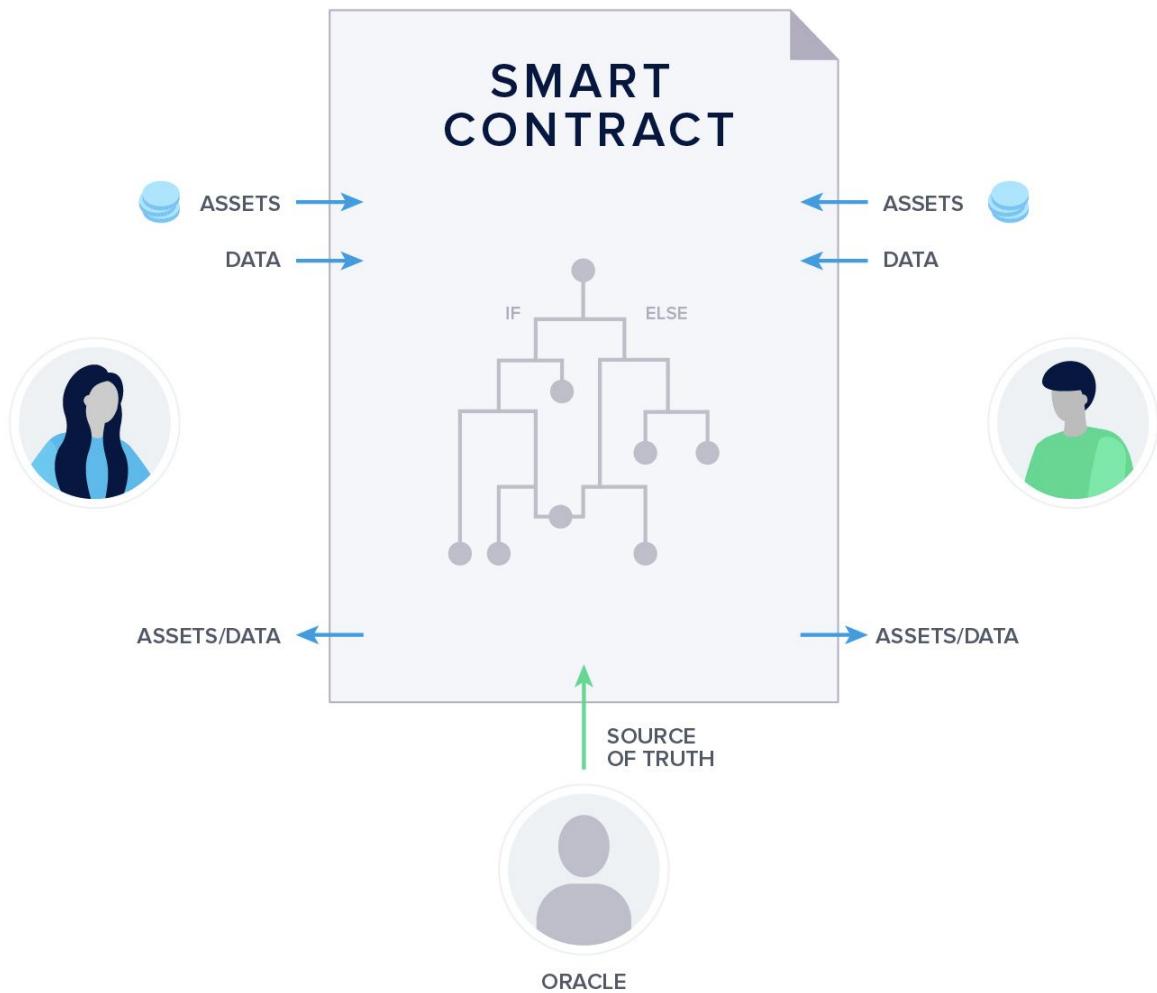
## Are They Really Trustless?

The promise of smart contracts is to allow trustless execution with automatically enforced rules without the need for middlemen. But can they actually live up to this promise?

All types of assets are subject to the local jurisdiction you are in. This means a contract, no matter if smart or not, requires additional trust in the jurisdiction besides the trust put in the contract itself. Possession in a smart contract does not equal possession in the real world. Just as with regular contracts, the terms can be subject to changing circumstances and interpretation thereof. An illegal contract is not legally binding.

One must consider that writing traditional contracts takes years of studying the legal framework regulating the different areas of contractual law. Writing smart contracts is even more difficult, as one also needs to understand the technical dimension behind them. We will need to see an entirely new generation of tech-savvy lawyers emerge to enable a meaningful adoption of legally binding smart contracts.

There is another major challenge to overcome. The digital world needs to learn about real-world events in order for a smart contract to function and execute. An *oracle* is an entity submitting data to a smart contract. The trust problem that comes with this role is referred to as the *oracle problem*. Imagine a smart contract running a betting platform in a trustless environment. The oracle needs to submit the result of a game in order to have the smart contract distribute funds to the winners. Because an oracle determines what a smart contract sees, it also controls what it does.



Centralized oracles are not considered a solution to the oracle problem. No matter what the actual implementation looks like, the incentives to untruthfully submit data might outweigh the benefits of acting honestly in some decisions. And what is the point of having trustless execution on the basis of information provided by a trusted third party?

No matter if centralized or decentralized, an oracle will always come at a cost. Acting honestly must always be the most profitable strategy and therefore strong incentives must be in place. This is another problem that needs game-theoretic evaluation and incentive design, just like the incentives for miners that we talked about in our last article.

Projects such as Augur and Gnosis are working on decentralized solutions to the oracle problem via *prediction markets*. Before these prediction markets gain serious adoption they will remain easy to game. And since prediction markets have the potential to influence the outcome of an event - as Dan Finley suggests - it remains to be seen if they can become a solution.

A screenshot of a Twitter post from Dan Finlay (@danfinlay). The post contains the following text:

Maybe/Theory: Since decentralized prediction markets can influence the incentives around an event's probability, they are never just prediction tools, and should always be treated as incentivization tools.

The post has 33 likes and was made at 5:48 PM - Jan 27, 2019. There is a link to see more of Dan Finlay's tweets.

[Link](#)

Before solving the problem of creating legally binding smart contracts and the oracle problem, the use of smart contracts will be limited to small experimental fields with low risk involved. An example of a potential use case would be in-game payments triggered by certain achievements that are objectively verifiable by code.

As Jimmy Song puts it: "A smart contract that trusts a third party removes the killer feature of trustlessness." There is a long way to go before we will see the broad use of trustless smart contracts across different domains, but they are most definitely a concept worth exploring.

## Summary

By now you should have a good idea of what a blockchain is. We looked at it as a data structure and as a set of rules, a protocol. A blockchain can store data reliably due to the distributed nature of the ledger. Anybody can participate in the network without any permission or registration as long as they follow the rules of the protocol. This makes public blockchains censorship-resistant, permissionless, trustless, and valuable. It can not only store pieces of data but also programs running in a distributed fashion.

Smart Contracts can ensure objective execution on the basis of mutually agreed-upon terms enforced by code. They have the potential to reduce middlemen and thereby reduce cost and save time. They will most likely foster a closer connection between software developers and the judicial system.

Before we can see widespread adoption of smart contracts we have to overcome some hurdles: Regulators will have to create a framework to allow the deployment of legally binding smart contracts and decentralized and trustless oracles need to be developed.

This was the last article in the chapter on what a blockchain is and what it can do. In the next chapter, we will take a look at how blockchains work.

# How Does a Blockchain Work?

In this chapter of our Advanced Level about blockchain technology, we will introduce you to the most important ideas, elements, and entities of a public blockchain.

In the first article of this chapter, we give you an overview of the different elements that make a blockchain work.

Hash Functions are an important part of every blockchain because they are used to verify the integrity of data.

Another major element is public-key cryptography. It is used to verify ownership and gave cryptocurrencies their name.

Here we talk about the "physical" infrastructure that blockchains are running on - the Peer-to-Peer Network.

There are many different parties in a blockchain and all of them need to agree on the transaction history. Here we show you how this is achieved.

In this last article of the chapter, we show you how miners secure the blockchain with their computational power.

## The Elements of a Blockchain

When Satoshi Nakamoto released the Bitcoin whitepaper he presented an innovation comprising several well-known components in a new composition. This chapter explains how a blockchain works and what its individual parts do. We want to put the individual pieces into context before we move on to explaining each concept separately.

### Hash Functions

A *hash function* is a mathematical function with a few special properties but like any other function, it does one job. The hash function takes an *input* and produces an *output* (also called *hash value*, *hash digest* or simply *hash*).

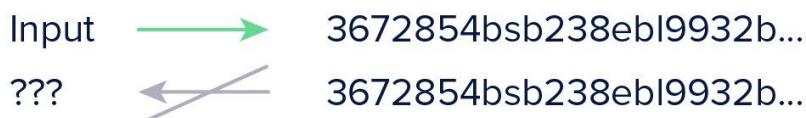
The input doesn't have to be a number, it can be any sort of data from a single character up to a large file like a video. The output of a given hash function has a constant length no matter the input.

There are many hash functions out there and most of them carry the length of the output they produce in their name.

One of the most commonly used hash functions is SHA256 (Secure Hash Algorithm 256 bit). The number indicates the output of the hash function has 256 bit. A hash value can serve as a fingerprint of data. One can verify the integrity of files and detect changes by comparing their hashes.



## 1. ONE-WAY FUNCTION



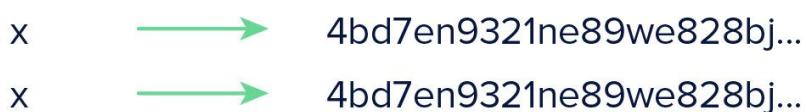
## 2. PSEUDO RANDOM



## 3. COLLISION RESISTANT



## 4. DETERMINISTIC



## Public-Key Cryptography

*Public-key cryptography* is also known as *asymmetric cryptography*. The term asymmetric stems from the property of keys always coming in pairs and them being used complementary. If you encrypt something with one of the keys, you need the other one to decrypt it and vice versa. The keys are called *public key* and *private key* (also *spending key* or *secret key*). Your keys translate to your identity on the blockchain. You receive funds with your public key and send funds with your private key.

HORIZEN ACADEMY

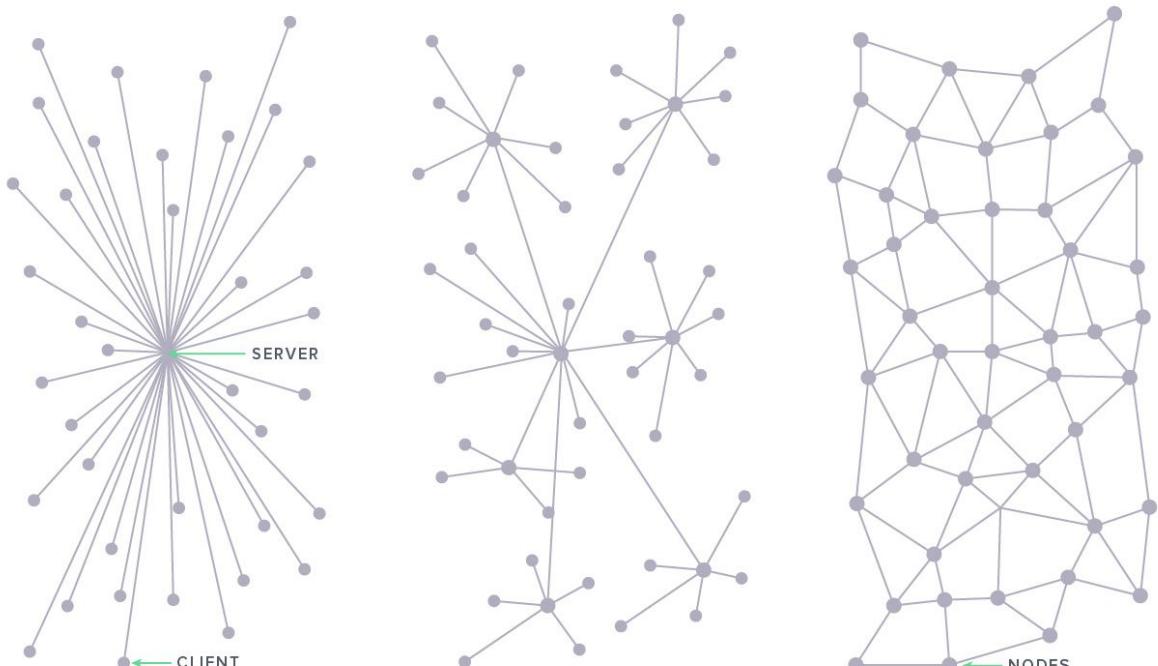
### ASYMMETRIC CRYPTOGRAPHY



## A Peer-to-Peer Network

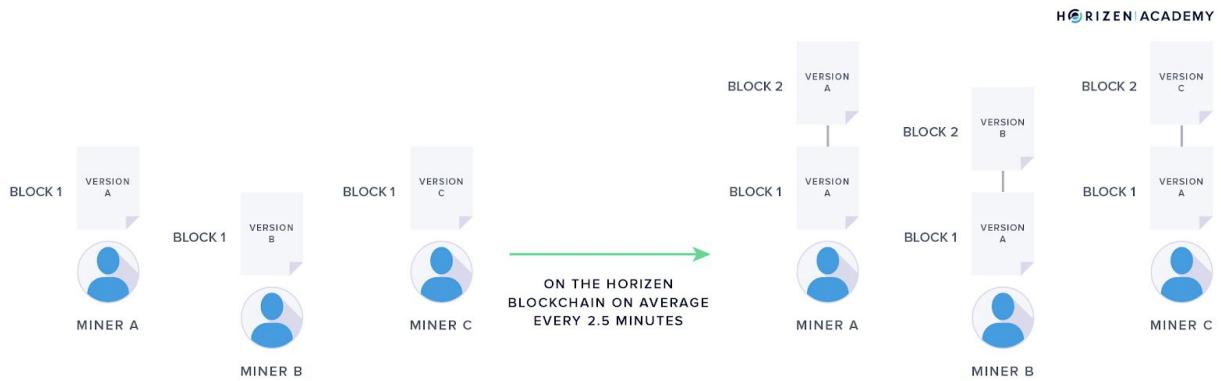
You have probably come across the term Peer-to-Peer (P2P) network before, most likely in the context of file-sharing services like BitTorrent. In a distributed P2P network the users don't connect to a central server to access a service but to many peers. The peers are other network participants and all of them provide the service to each other. P2P networks are very resilient, as there is no single point of failure.

Blockchains make use of this concept and it is one reason why they are so robust. You will often hear the attributes *permissionless* and *censorship-resistant* when reading about the value propositions of public blockchains. The Peer-to-Peer network plays a significant part in giving blockchains these properties.

CENTRALIZED  
(A)DECENTRALIZED  
(B)DISTRIBUTED  
(C)

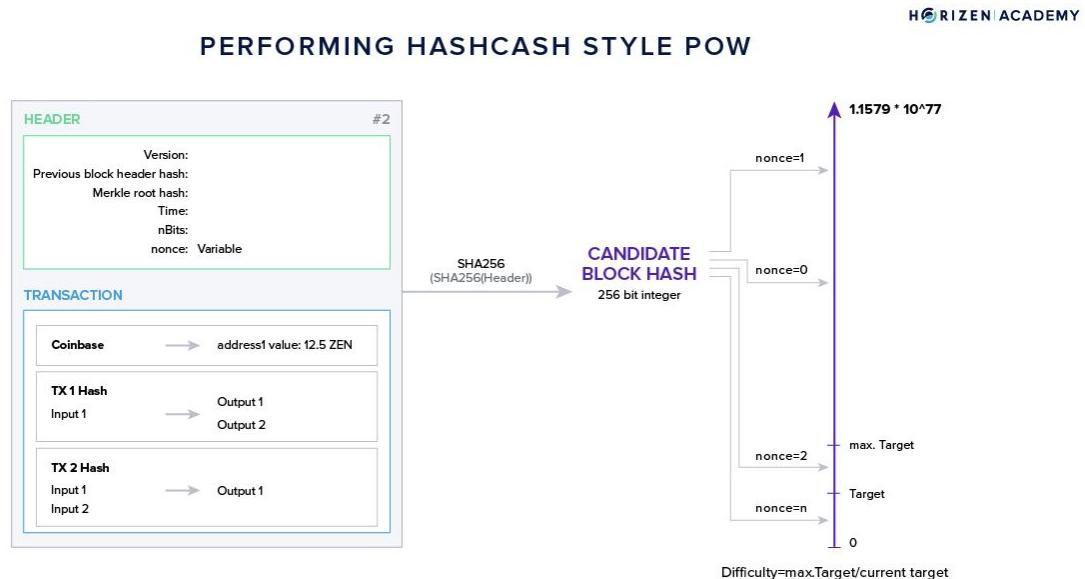
## Consensus Mechanism

If you want to build a form of digital money on a P2P network with many unknown participants, you will need to build *consensus* on the order of transactions. If a user has one ZEN and creates two different transactions that spend that same coin simultaneously, some peers will receive version A first, while others might receive version B first. The network needs to come to an agreement, or consensus, on which of the two transactions happened first and is valid. The *consensus mechanism* is what enables a large number of different entities that neither know, nor trust each other to agree on a single version of the blockchain.



## Mining

There are many consensus mechanisms. The "original" introduced with Bitcoin is the Proof of Work (PoW) mechanism. You have probably heard about cryptocurrency mining before and about the miners having to solve a computationally expensive puzzle. We want to explain what this puzzle is, and how solving the puzzle makes the network agree on a given order of transactions.



## Demo

This is a [great blockchain demo](#). We invite you to go play around with it, either now, or after you have finished this chapter. The demonstration is hands-on and really helps to understand how a blockchain is built. The demo provides a guide that will lead you through the processes running on a blockchain step-by-step.

## Hash Functions

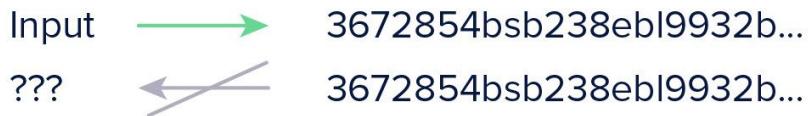
Blockchains are a way to record data in an immutable way. For this to be a valuable feature, the data written to the blockchain better be correct. Verifying what data gets added makes a lot of sense, especially as storage on the blockchain is limited. One tool in the toolbox used to verify data is the *hash function*.



Hash functions are an integral part of blockchain technology and serve many purposes. A hash function is a mathematical function that takes an input of any length and produces an output of fixed length. The output is often called a *hash value*, *(hash) digest*, or simply *hash*.

There are [many hash functions](#) and just as many online "calculators" for hash functions such as [this one](#). The tool allows you to hash any input with different hash functions at the same time. The size of the input can range from a single digit to entire files, but the size of the output will always be the same.

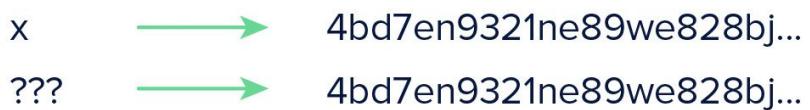
## 1. ONE-WAY FUNCTION



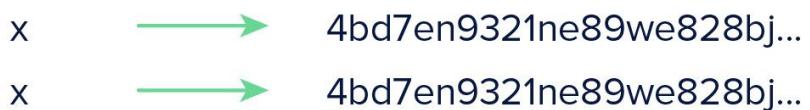
## 2. PSEUDO RANDOM



## 3. COLLISION RESISTANT



## 4. DETERMINISTIC



A cryptographic hash function must fulfill the following set of criteria to be viable for use in a blockchain:

- *One-wayness* - It has to be easy to calculate an output from a given input but impossible to calculate the input from a given output.
- *Pseudorandom* - A change in the input must result in an unforeseeable change of the output. If the hash value of the input 2 was 4, the hash of 3 better not be 6.
- *Collision resistant* - It should be hard (read impossible) to find two inputs to a hash function producing the same output.

- *Deterministic* - A given input always needs to produce the same output

The most frequently used hash function today is *SHA-256*. SHA is an acronym for *Secure Hash Algorithm*. The number indicates the length of the output in bits, e.g. there are four different lengths of outputs in the SHA family: 224, 256, 384 or 512 bits. Another type of hash functions relevant in blockchain tech is the RIPEMD family. *RIPEMD160* is used in many cryptocurrencies and as you might have guessed produces an output of 160 bits.

Hash values are used for many purposes in cryptocurrencies and blockchains. The most notable use case is the process of chaining together the blocks, thus creating the blockchain. We call a hash value a fingerprint of data for its property of being collision-resistant.

Each set of data used as an input can be easily identified by the unique hash that it generates. It is nearly impossible to find two inputs to the hash function, that result in the same output (cause a collision). It would take all the supercomputers on earth combined several thousands of years to create a collision. This is because there is no way to "calculate" a collision. The only way to find one is through a brute-force approach, where you try different inputs until you get a collision by chance.

This works because the *output space* of a 256 hash function is incredibly huge. The output space describes the set of all possible hash values from 0 to  $1.1579 \times 10^{77}$  which can be written out as

115792089237316195423570985008687907853269984665640564039457584007913129639936

## Summary

Hash Functions are the first cornerstone of blockchain technology. A hash of a file is like a fingerprint. It is easy to detect if two files are identical by comparing their hashes. Blocks are "chained" together by including the hash of the last block in each new block. You cannot change data from previous blocks without breaking the chain of references.

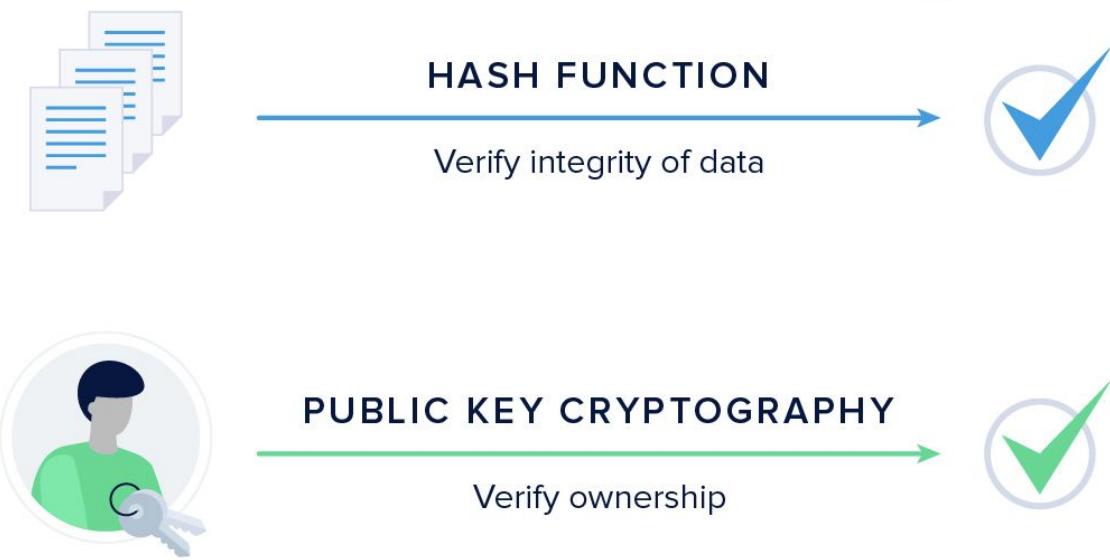
In our next article, we will look at public-key cryptography - the second cornerstone of blockchain technology and the concept that makes ownership on the blockchain possible.

## Public-Key Cryptography

Since the goal of the original blockchain design was to enable a new and fair form of money, it would be great to be able to actually own money in this system, wouldn't it? There must be a concept of identity to have ownership on the blockchain - you cannot have ownership if there is no representation of the owner.

*Public-key cryptography* makes it possible to represent identity on the blockchain. It is the second cornerstone of blockchain technology besides *hash functions* that we talked about in our last article. While hash functions are used to verify the authenticity and integrity of data, public-key cryptography is used to verify ownership on the blockchain.

HORIZEN ACADEMY

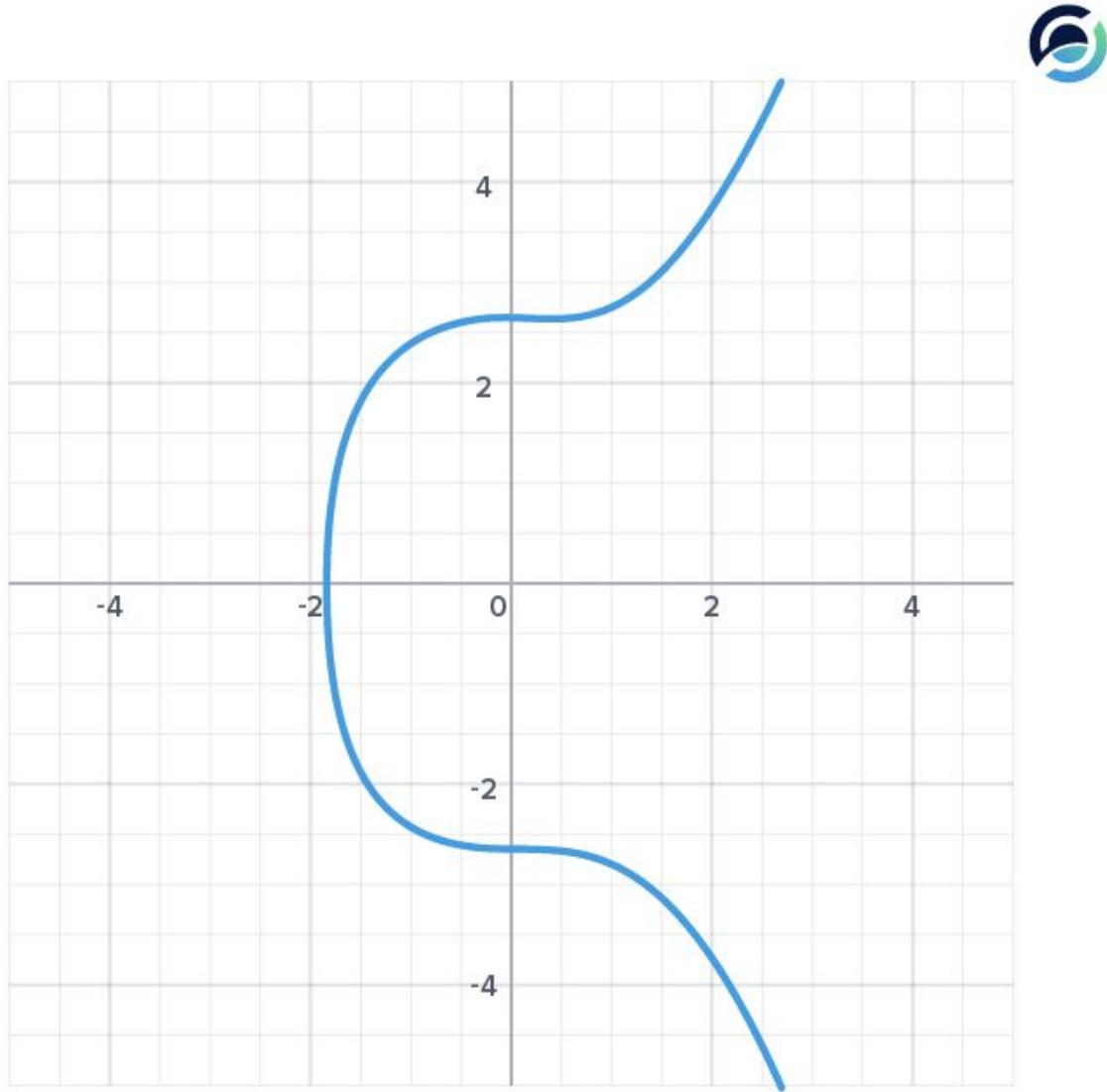


Let's take a step back and start from the beginning.

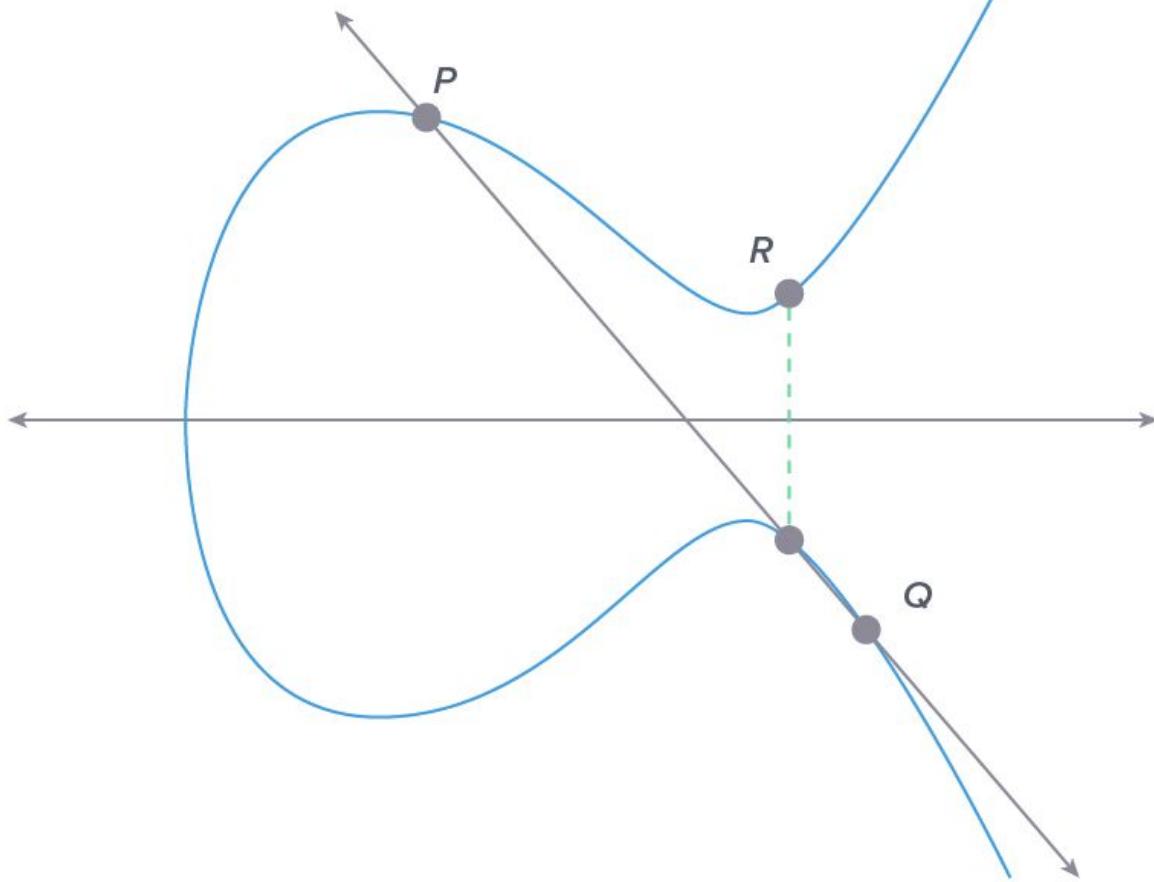
The basis of public-key cryptography is private keys, public keys, addresses, and digital signatures. When you own some cryptocurrency, the blockchain contains a record that there are some coins associated with your public key. You must provide a digital signature to authorize the spending of those coins. You can only provide this digital signature if you are in the possession of the private key that corresponds to the public key.

## Elliptic Curve Cryptography

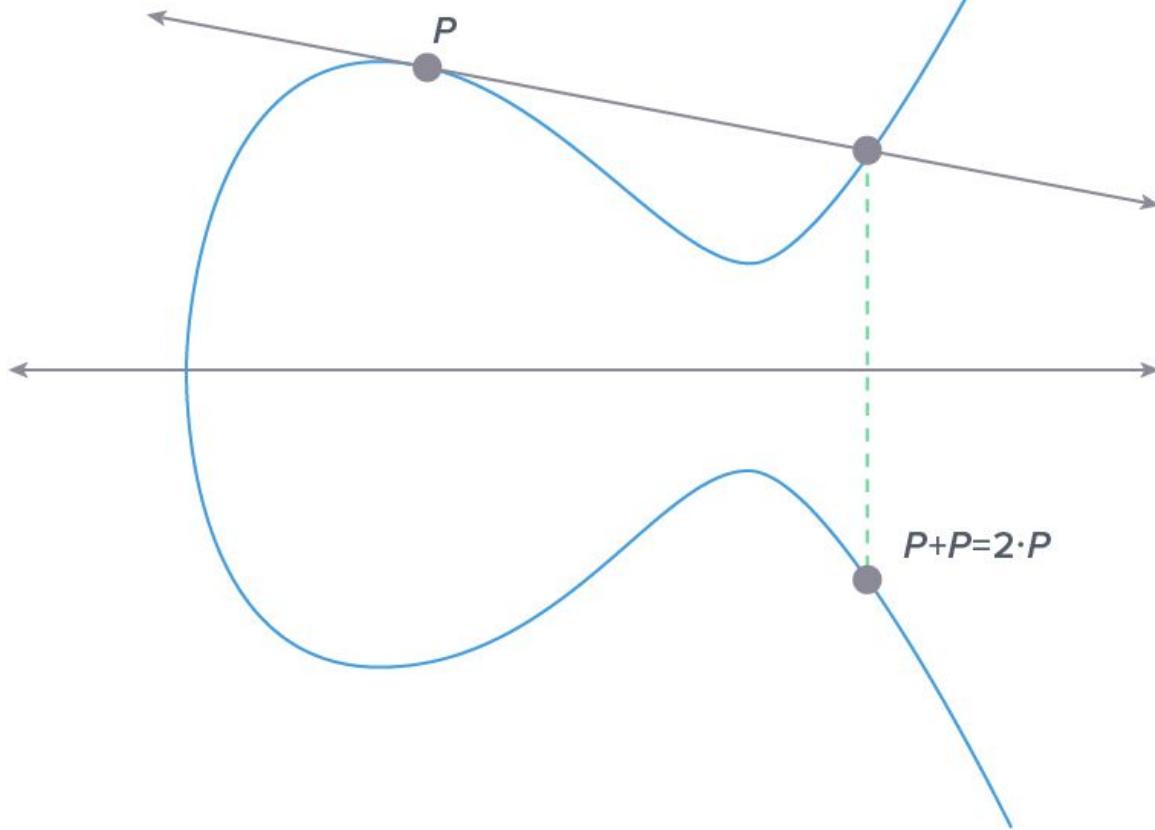
To understand how your keys and addresses work together we must introduce *Elliptic Curve Cryptography* (ECC) first. There are different ways to build a public-key cryptography scheme. Bitcoin and most other cryptocurrencies use Elliptic Curve Cryptography (ECC).



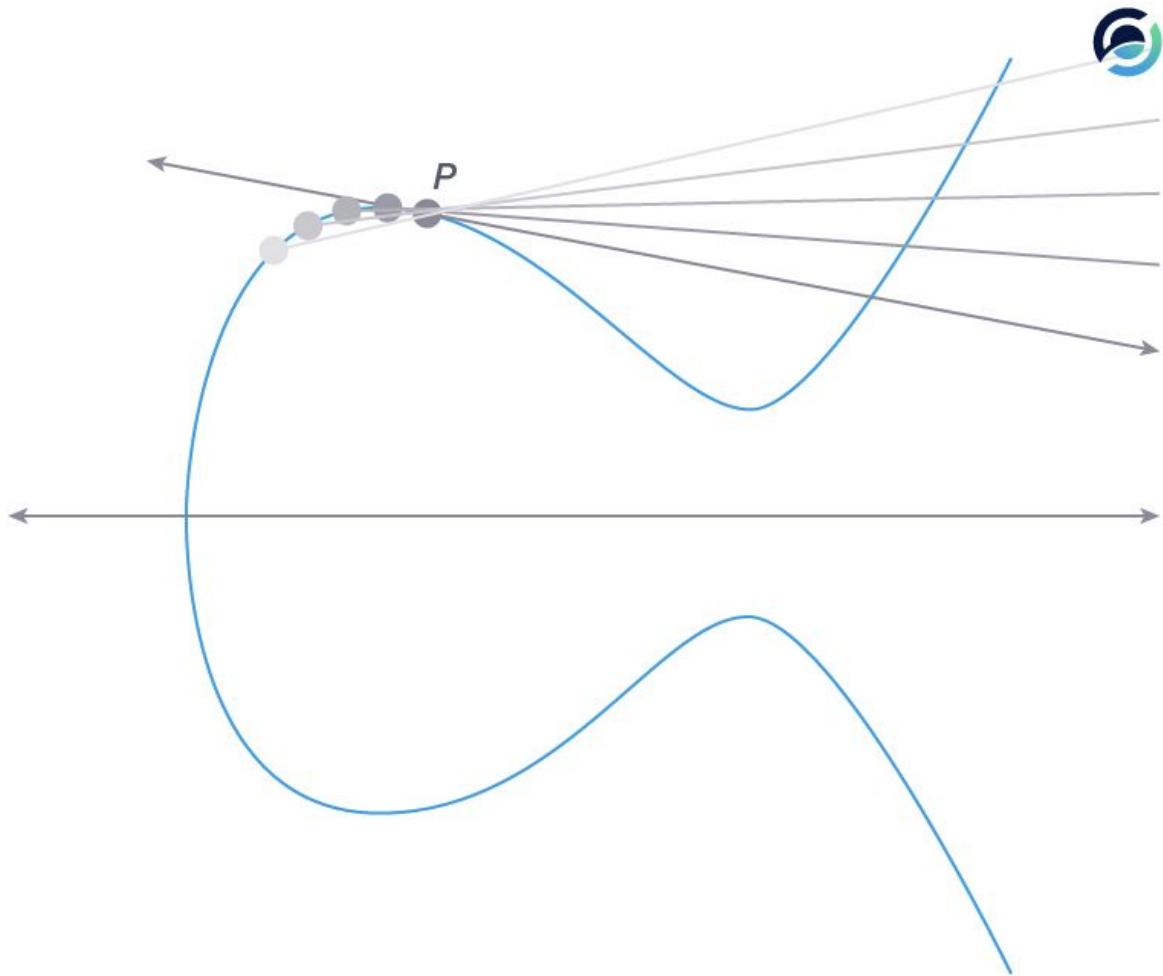
Bitcoin, Ethereum and many other currencies use a curve called secp256k1 and it looks like the one on the left. The equation for this curve is  $y^2 = x^3 + 7$ . What makes elliptic curves useful is that you can do math with them, and the math you do on the curves has some special properties.



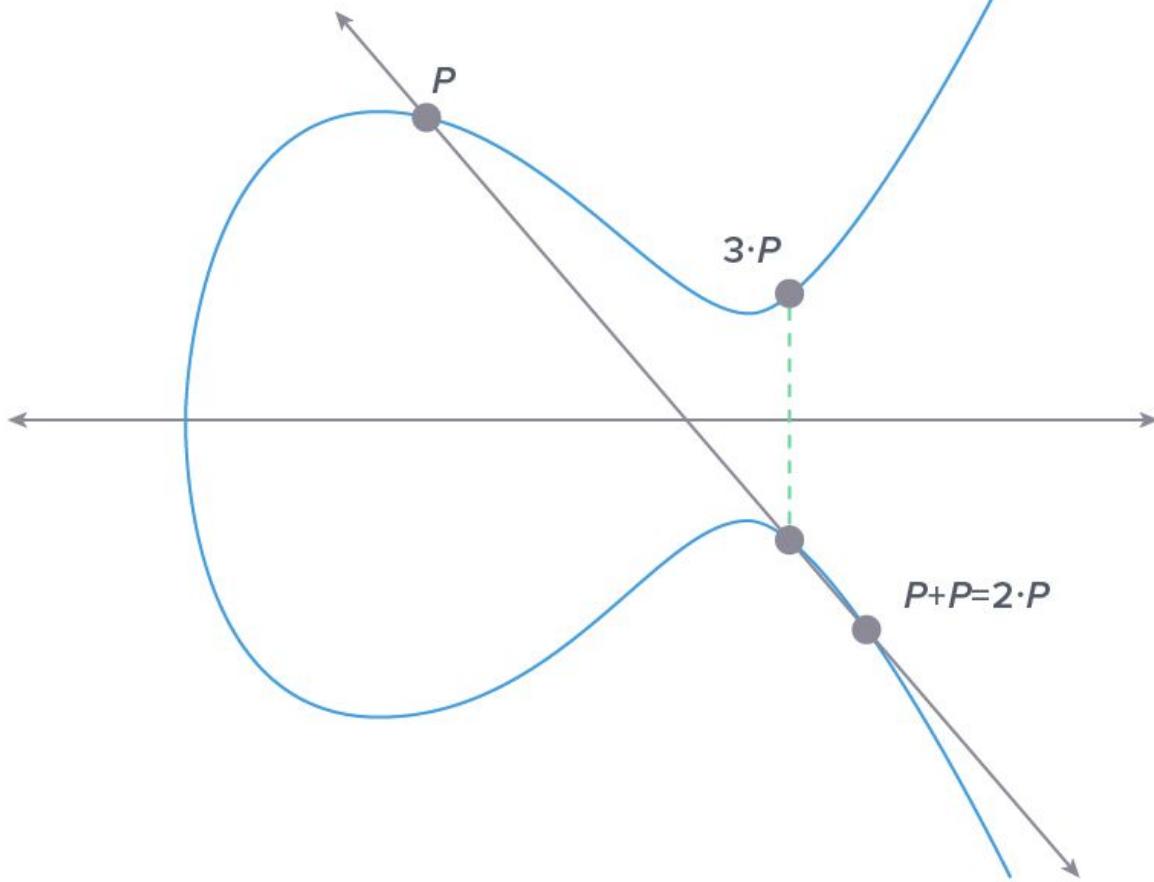
The graph above shows an example of adding two points on the curve together. When we want to add point P and Q together, we first connect them with a straight line. This straight line will intersect with the curve at a third point. Now we must project the third point onto the other side of the x-axis (multiply the y-coordinate by -1) and we get the sum of point P and Q: R. The key takeaway is that the sum of two points on the curve is a third point on the curve and it is easy to compute.



When we want to multiply a point on the curve we must add it to itself. To multiply point  $P$  by two we add it to itself once. In this case, we can't really connect two points, so we go for the tangent line (the one with the arrows).



Why the tangent line? you might ask... If you look at a random point close to P (the lightest gray), connecting the two points will result in the lightest grey line. Moving this point closer and closer towards P (from light to dark) brings the connecting line closer to the tangent line. The closer two points on the curve get, the closer their connecting line resembles the tangent until they become the same.



Now the addition to itself is easy. We take the intersection of the tangent line with the curve once again and project it onto the other side of the x-axis - easy.

If we want to multiply  $P$  by 3 we need to add  $P$  and point  $(P + P)$  together. To multiply  $P$  by four we can add point  $(P + P)$  to itself and so on.

The key takeaway is that multiplying a point is an easy task - it is just the addition of points to themselves. The interesting part is that division on the curve is a *computationally hard* problem. There is just no algorithm to calculate how many times  $P$  was added to itself or in terms of multiplication by what number it was multiplied in order to get to a certain point. This is already enough to understand the basic concept of public-key cryptography based on elliptic curves.

## Private Key

When you set up a wallet, the first step is generating your private key. Your private key is a very large random number, 256 bit long. This number is so large you could assign almost every atom in the observable universe a unique private key. Your private key should be as random as possible. Creating random numbers is harder than it might sound, but this step is crucial to the safety of your funds.

## Public Key

Next, your public key is derived from the large, random number you have generated as your private key. This is where we need to multiply points on the curve. As we said earlier, adding points together on an elliptic curve is straightforward. Bitcoin uses a base point on the curve for every key pair.

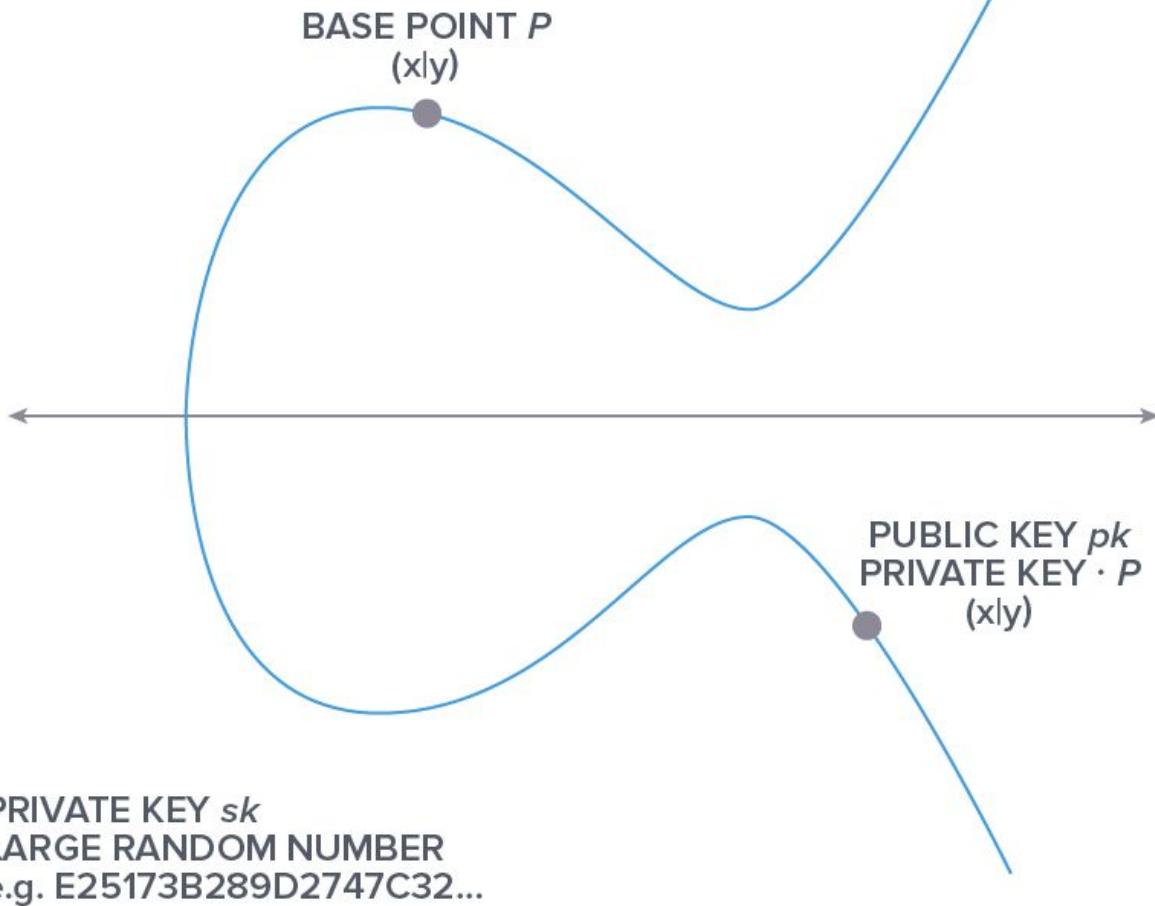
Its coordinates are

$x = 55066263022277343669578718895168534326250603453777594175500187360389116729240$

and

$y = 32670510020758816978083085130507043184471273380659243275938904335757337482424$

This base point is now added to itself as many times as your private key dictates. If your private key was the number "3", then you would perform the calculation we just demonstrated above. If you add the base point to itself as often as your private key says (private key  $P$ ) you get your public key.



**PRIVATE KEY  $sk$**   
**LARGE RANDOM NUMBER**  
e.g. E25173B289D2747C32...

To recap: Your private key is a large random number. Your public key is a point on the elliptic curve that you get when you multiply the base point  $P$  with your private key.

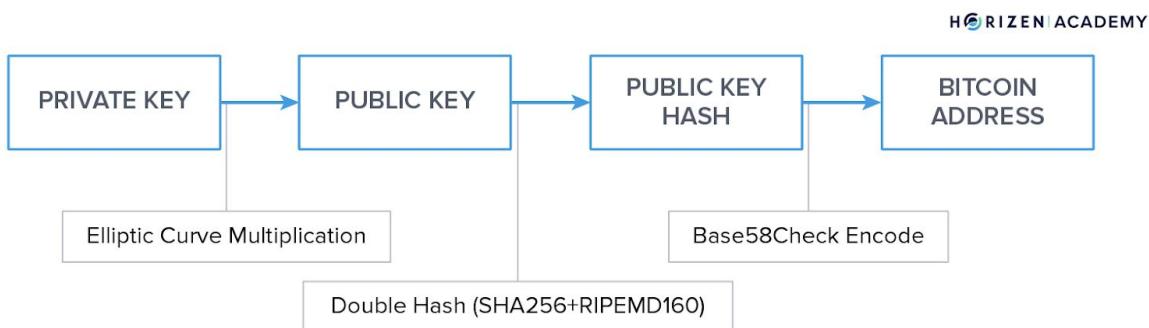
A property that is necessary for any public-key cryptographic scheme, is that it is computationally infeasible to derive the private key from the public key. It is easy to calculate the public key, a point on the curve, by multiplying the base point  $P$  with a large random number (your private key). But if an adversary knows the base point  $P$  and your public key, he cannot divide them and say how many times  $P$  was added to itself to get your public key.

At this point, your public key is pretty large, 512 bit, and it is easy to compress it to half the size. The elliptic curve is symmetric about the x-axis. There are only two possible y-values for every x-value

that only differ in their sign (being positive or negative). If you leave out the y-coordinate and add the information of the point having a positive or negative value for y, the public key becomes half the size but carries the same information.

## Address

Lastly, to get your address, your public key is hashed. First, using the SHA256 hash function and a second time using RIPEMD160. After adding a byte to state if this address will go to the main- or test-net and calculating a checksum there is a final step before you get your address.



When we look at how a computer is working at the hardware level it is zeros and ones. Data is always stored in a binary format no matter what type of data you're looking at (images, sound files, and even your bitcoin address). There are different ways to convert a string of bits into data that humans can digest. Humans do best with a string of numbers or alphanumerical characters.

Base58Check is a way to convert bits into alphanumeric characters, but it excludes the four characters 0, O, I, and l. Base58Check removes these characters from your address to reduce errors when copying addresses manually and proofreading them.

You can generate as many addresses as you like from a single private key, and wallets do this for you automatically. It is a feature to enhance your privacy, as it makes it harder for a third party to link all your payments together. We will talk about this concept and how it works with change addresses in the chapter on privacy on the blockchain.

## Digital Signature

To wrap up this article we want to come back to digital signatures. You might hear that your keys are used to encrypt and decrypt messages. This is not really the case. The information contained within transactions is not encrypted in any way. It is available to anyone on the blockchain, which makes the system so transparent.

What your private key is actually used for is signing transactions. You can only spend funds you received in a transaction if you provide a digital signature that proves your knowledge of the private key corresponding to the address used to receive the funds.

We cover how digital signatures work and how you can prove that you know the private key without revealing any information about it in our Expert Section. We combine the concept of hash functions and point multiplication on the curve for this. Although it is not exactly rocket science it is a rather complex matter.

## Summary

When you set up a wallet the software will first generate a large random number that is your private key. The base point  $P$  on the elliptic curve is multiplied with your private key to get your public key, another point on the curve. Your public key is then hashed and the characters I, l, 0 and O are removed to improve readability. To spend your money you need to provide a digital signature that proves your knowledge of the private key that belongs to your address.

Our next article is about the Peer-to-Peer network: The infrastructure public blockchains are built upon.

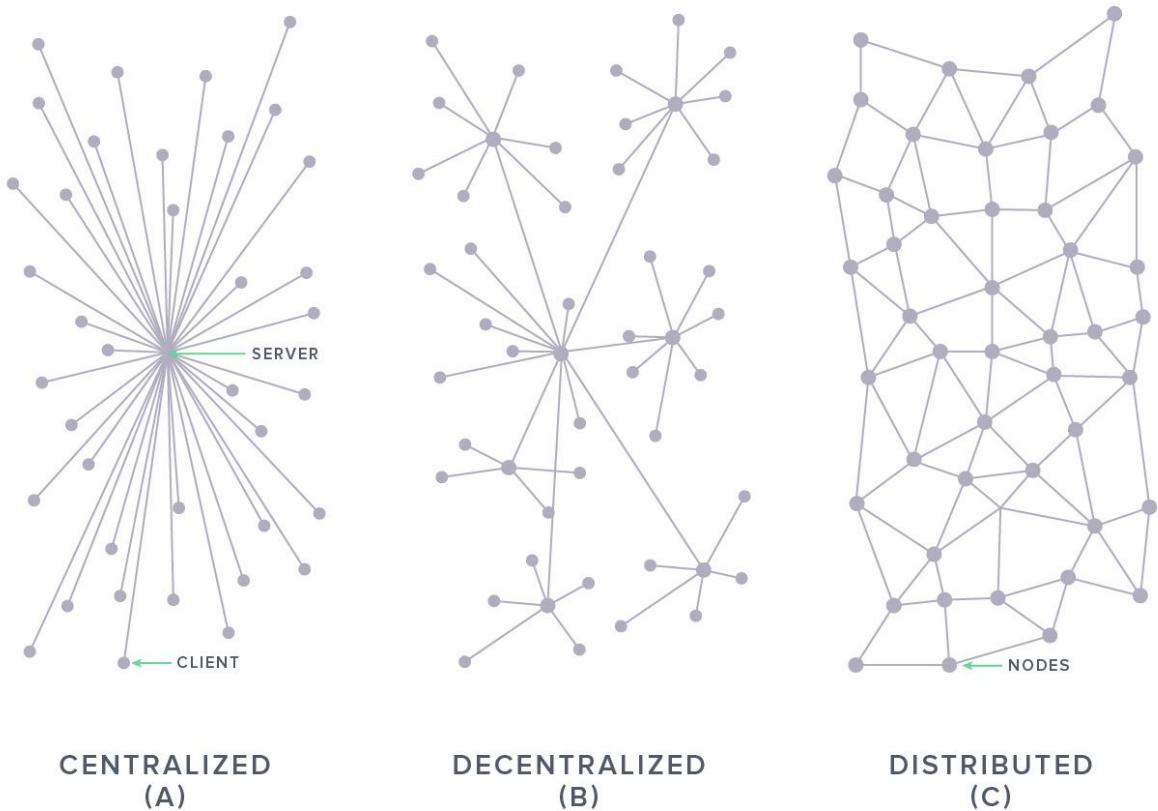
## The Peer-to-Peer (P2P) Network

The Internet that we are experiencing today is highly centralized. Most data that we as the users of the internet produce, end up in the hands of a few large corporations. But there are a number of truly distributed systems out there living on the internet. Although not exactly the same, for the purpose of this article, we will use the term distributed network and Peer-to-Peer network interchangeably.

One example of a truly distributed system on the internet is BitTorrent. Like any other technology, Peer-to-Peer networks have enabled legitimate use-cases, such as the reliable exchange of open-source software as well as illegitimate use-cases, such as pirating music and movies. Distributed systems have some major advantages over their more centralized counterparts, most notably their robustness. Peer-to-Peer networks have a high level of redundancy built-in. Single points of failure are missing and the system can survive even if a majority of the network is shut down. We've seen the tremendous difficulty that authorities have had taking some of these networks offline with services like BitTorrent or Napster. That is due to the fault tolerance you get from a peer-to-peer architecture.

But there are some downsides compared to centralized systems. The high level of redundancy and the need for communication as well as coordination between peers comes at the cost of efficiency. Taking a look at data storage is the most obvious example here. Many nodes, in case of the Horizen network more than 25,000, store a copy of the blockchain. This is not very storage efficient but makes the blockchain highly resilient against attacks and gives it its immutability.

In computer science, the [CAP theorem](#) describes the cost of a robust and scalable distributed network as the time it takes for the network to reach consistency. It takes some time for an event (like a transaction) to be broadcast to every node on the network. In a second step, all the nodes have to reach consensus on the order in which events happened. In our next article on consensus mechanisms, we will introduce you to the two main methods to achieve consensus in distributed networks.



We have found [a simple yet great visualization](#) of a distributed system that demonstrates the process of new peers joining a network and syncing with all other nodes. It lets you add and delete random nodes to show the robustness of the overall system. With a Peer-to-Peer network architecture, every node is equal to every other node. Every node in a P2P network acts as both a client and a server opposed to traditional client-server models. While a server can experience downtimes during which the clients cannot access its data, in a P2P network you just connect to a different peer if one goes offline.

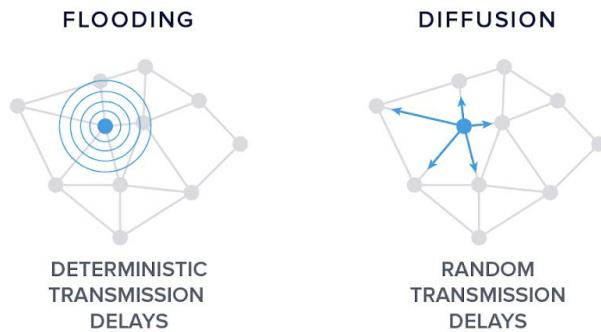
## Variations

The variations of distributed networks mostly regard the data structure that is being maintained and the *broadcasting mechanism* or message propagation that nodes use to communicate and exchange data. The two most commonly used spreading protocols are *flooding* and *diffusion*. With flooding, nodes propagate a message with a constant or *deterministic* transmission delay. Diffusion is a refined version of flooding. In networks using diffusion, such as Bitcoin and most other cryptocurrencies the nodes propagate messages with a random transmission delay.

This makes it harder for an eavesdropper to determine the origin of a message and thereby identify a node's geographical location and possibly owner.

## BROADCASTING PROTOCOL

HORIZEN ACADEMY



Many distributed systems are built with an append-only data structure and blockchains are an example of this. The blocks in a blockchain are cryptographically connected data containers in an append-only log. It is infeasible to change or delete data from the past.

In other distributed networks it can be desirable to delete data, for storage efficiency or to maintain only relevant information. In the case of blockchains, this would be considered a devastating bug. If the transaction ledger were to change this would destroy the use-case of digital currency, because it would imply changing balances. This is clearly not desirable for a censorship-resistant form of global money.

Most distributed networks use a variation of diffusion for message propagation. A peer that receives a message will broadcast it to all his peers, which in turn broadcast it to all their peers. Within a few rounds of propagation, the entire network will receive the message as it spreads exponentially among the nodes. The communication protocol needs to fulfill a set of desirable properties, such as low latency, fairness (all nodes experience roughly the same latency) and anonymity. Anonymity in this context should prevent an adversary from learning the IP address a message originated from through monitoring the network over time.

### Incentive

In order for a distributed network to function all network participants need some form of incentive to act according to the protocol of the network. With a blockchain for cryptocurrencies, the incentives

are monetary rewards. The miners are rewarded with an economically valuable token for verifying transactions and achieving consensus across the network. Non-mining nodes are usually not incentivized.

## Secure Nodes and Super Nodes

Horizen also incentivizes node operators financially because running a node comes at a cost. It takes time to set up, you need to set up or rent a server, and the node operator has to update the software occasionally. On the Horizen network, miners receive 60% of the total block reward (12.5 ZEN). The other 40% is used to fund the infrastructure (Secure Nodes 10% and Super Nodes 10%) and the non-profit Zen Blockchain Foundation (20%) for the development of the protocol.

On a permissioned blockchain that a consortium of companies is running, e.g. to track a supply chain, the incentive to maintain the ledger is to have access to valuable business data without having to trust a third party. We believe that the Web 3.0 will benefit largely from the emergence of distributed networks and that we will see a transition from centralized services to decentralized ones.

## Summary

Peer-to-Peer networks offer great robustness or fault tolerance at the expense of efficiency. Every node is performing the same task on the network and acts as a client and a server at the same time. If one of your peers goes offline you just connect to another one. If you run a node and happen to go offline for a while, you need to reconnect at some point to get updated by your peers on the blocks that you missed to become fully functional again.

In the next article, we will introduce you to the two most common mechanisms to reach consensus on a history of events in a distributed setting. You have probably come across the term Proof-of-Work and mining before. We will explain what Proof-of-Work and Proof-of-Stake are about and look at mining in detail for the last article of this chapter.

## Consensus Mechanisms

The distributed network of a blockchain comprises many thousands of participants - some of them altruistic, others rational, and some of them malicious. In this challenging environment, one has to solve a crucial problem to make digital money work: all nodes need to agree on a single history of transactions. Without consensus on who owns what, the network would be worthless.

The consensus mechanism of a blockchain allows the network to agree on a single version of history. The history in the case of a blockchain for cryptocurrencies is the order in which the transactions on the network happened.

When a network participant creates a transaction, the transaction is broadcast to the entire network. Each node records the transaction and adds it to their version of the ledger. The different versions kept by different nodes look slightly different. If you are in the US and broadcast a transaction the nodes that are closest to you will receive it earlier than a node based in Asia. What you get is a set of slightly different versions of the same transaction history. Eventually, all network participants need to agree on a given order and this is what the consensus mechanism of a blockchain does.

There are many approaches to achieving consensus in a distributed network but the two most commonly used ones are the Proof of Work (PoW) and Proof of Stake (PoS) algorithm. The generalization Demiro Massessi has formulated applies to both:

*"The main difference between consensus mechanisms is the way in which they delegate and reward the verification of transactions. (...) In one way or another, blockchain consensus algorithms boil down to some kind of vote where the number of votes that a user has is tied to the amount of a limited resource that is under the user's control."* - Demiro Massessi

# CONSENSUS



Decentralized blockchain needs a way for users to agree on the current state of the blockchain.



Consensus in the blockchain network is based on scarcity. Controlling more of a scarce resource gives more control over the blockchain's operation.



Several different consensus mechanisms have been proposed for blockchain.

The most common are:

- Proof of Work
- Proof of Stake

Other consensus mechanisms:

- Delegate Proof of Stake
- Practical Byzantine Fault Tolerance
- Proof of Activity
- Proof of Elapsed Time

## Proof of Work - PoW

Miners are the entities who work to achieve consensus within the network in a Proof of Work blockchain. They collect all transactions that are forwarded to them through the P2P network and save them in their *mempool* (memory pool). While they collect incoming transactions, they verify if the transactions are valid according to the protocol and add them to the block they are currently working on. At the same time, they are solving a computationally expensive puzzle. We will cover what this puzzle actually is in the next article on mining.

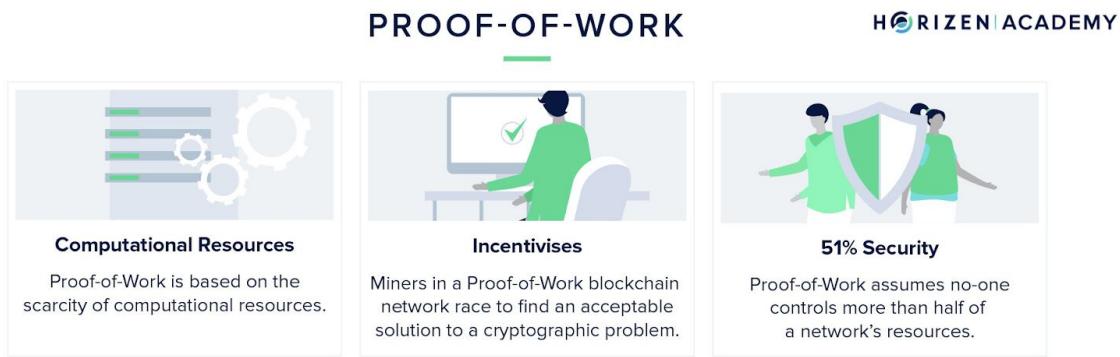
The miner who solves the puzzle first broadcasts her block to the network and gets to extend the blockchain by one block. The miner receives newly created coins for his work and he gets to write the history of the blockchain for the past couple of minutes. In Horizen this happens every 2.5 minutes on average, in Bitcoin every 10 minutes. The chance of solving a block is directly proportional to the computational power a miner has. If a miner had 10% of the computing power of the network, she would solve every tenth block on average.

Computational power is the limited resource in a PoW-based blockchain. It takes real-world resources, namely mining hardware and electricity, to mine a cryptocurrency. It is a highly competitive environment, in which each miner constantly wants to increase his share of the computational power or *hash rate*. This competition is what keeps the chances of a single actor controlling the majority of hash rate low.

Miners require computational power because the problem they are trying to solve can only be solved through random guessing. PoW only works, because this puzzle is *optimization-free* and *approximation-free*.

Optimization-free means there is no shortcut to trying out a large number of different solutions until you find a valid one. You cannot calculate a solution, you need to find it through a trial-and-error approach.

Approximation-free means that it is not possible to have a part of the solution or to "be close to solving the problem". You either have a solution or not, making it a binary situation.



All nodes and miners verify the validity of a new block as soon as it is discovered and broadcast. If it is valid, they add it to their local copy of the blockchain and delete all transactions that are now

already recorded on the blockchain from their mempool. The mempool only ever contains valid but unconfirmed transactions. Then they start working on the next block and the process is repeated. This is how the network agrees on a single version of the history of all transactions in a Proof of Work blockchain.

## The Longest Chain Rule

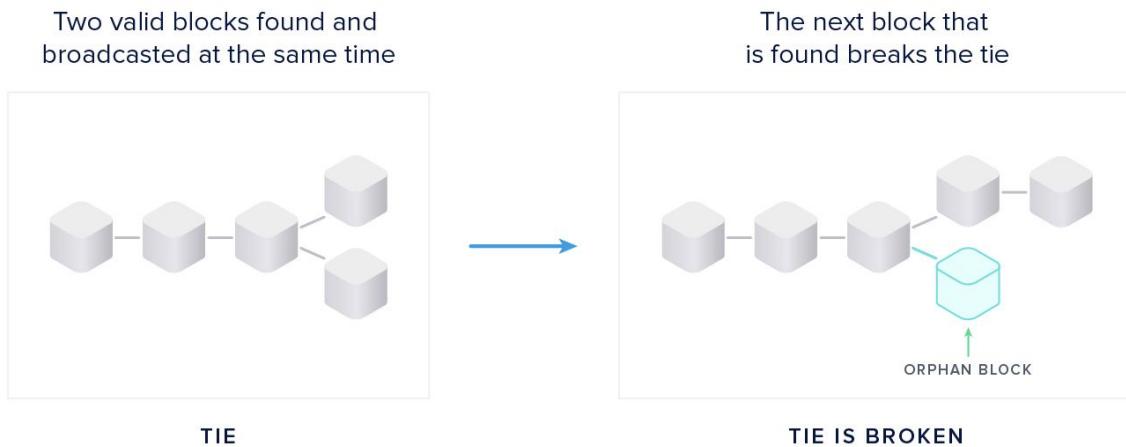
Now you can imagine a scenario in which two miners find a block at almost the same time. In this case, all the nodes and miners on the network save both versions of the block. This is a tie situation: both blocks are valid at this point, but somehow the tie must be broken - we need a single version of the truth.

The miners will start building the next block on top of the block they received first. The tie is broken when the miners find the next block. The block that is built on top of will become accepted as the single truth by all miners and nodes. The other block is disregarded and called an *orphan block*. This procedure of breaking a tie between two concurring blocks is called the *Longest Chain Rule* or *Nakamoto Consensus*.

If 80% of miners receive block A first and the other 20% block B, then the chances of block A getting extended are 80% (assuming all miners have the same computational power). In a way, the miners vote with their computational power on one version of the history. This aligns perfectly with our quote from the beginning of the article:

*"In one way or another, blockchain consensus algorithms boil down to some kind of vote where the number of votes that a user has is tied to the amount of a limited resource that is under the user's control."* - Demiro Massessi

## LONGEST CHAIN RULE



Proof of Work is one of the most secure consensus mechanisms but it is only secure if there is a sufficient amount of hash rate on the network. The Bitcoin protocol has proven how secure Proof of Work consensus can be for over 10 years. We already talked about game theory in our article describing blockchain as a protocol to transfer value and in the article on smart contracts.

Game theory "is the study of mathematical models of strategic interaction between rational decision-makers" ([Wikipedia](#)). Miners are rational decision-makers in a PoW blockchain. The incentives, to act according to the rules of the mining protocol are embedded in the protocol itself and outweigh the potential gains of behaving maliciously. This is what makes a mature PoW blockchain so robust.

In our article on mining, we will talk about this process in more detail and you will learn what the puzzle that we are talking about in the context of PoW is.

### Proof of Stake - PoS

In a PoS blockchain, there are also entities collecting transactions and creating new blocks. The process, as well as the terminology in this setting, is a little different though.

Whereas miners are mining blocks in PoW, *validators* are *forging* blocks in PoS. The chance of validating a new block is proportional to the stake a validator has. The stake refers to the number of coins a validator is willing to lock up for the time they want to be a validator.

To become a validator, a user needs to send funds in a special type of transaction. These funds are locked in a deposit called the *validator pool* and only released if the validator acts according to the rules of the protocol. If the validator were to include fraudulent transactions in his block, he would lose his stake and the right to forge blocks in the future.

The scarce resource in a PoS environment is the native currency of the blockchain. The more money you stake on a PoS blockchain, the higher your chances of validating a new block. In (most) Proof of Stake blockchains, there is no block reward. The validator's incentive to do achieve consensus is solely based on collecting the transaction fees attached to each transaction.



## Comparing POW and POS

Skeptics question the overall security of the Proof of Stake consensus model because it doesn't consume real-world resources to be a validator. There is no cost associated with building a new block on top of both branches when there is a temporary fork. This is referred to as the *Nothing at Stake* attack.

In PoW, there is a real-world cost (electricity) to every block that gets mined. It remains to be seen if PoS blockchains can provide the same security guarantees over an extended period of time that Bitcoin with its PoW blockchain has shown for over a decade now.

## PROOF OF WORK

-  Distributed consensus among untrusted and unknown nodes
-  Incentives are rewarded within the system for work done outside the system
-  Relatively high cost of entry, but high returns
-  Slow transaction rate
-  Low efficiency, consumes more energy
-  Empirically proven

## PROOF OF STAKE

-  Distributed consensus among untrusted and unknown nodes
-  Incentives are rewarded within the system for escrow inside the system
-  Low cost of entry, but low returns
-  Fast transaction rate
-  High efficiency, consumes less energy
-  Experimental

Another difference between PoW and PoS is that in a PoS blockchain each validating node needs to be identifiable. The staked coins must be held accountable for any malicious acts.

In a PoW blockchain, there is no need to have miners or nodes identifiable. In fact, it is a feature, that if a node receives a block, there is no information included about who the miner of the block was. It might have been the node you just received the block from, or it might have been relayed several times before it reached you.

It does not and it should not matter who the miner was. All that matters is that the block and all its transactions are valid. You only need to trust the math in order to trust Proof of Work.

Supporters of Proof of Stake refer to the high energy cost and limited throughput of PoW blockchains and thus consider the PoS consensus mechanism to be more sustainable.

Right now there is no PoS blockchain available that can support the claim of being secure over an extended period of time while storing significant value. Time will tell if PoS blockchains can deliver on their promise of being equally secure while more scalable than PoW blockchains.

## Summary

In Proof of Work consensus, miners are voting on a version of the history with the weight of the computational power they control. In Proof of Stake, validators are voting on a version of the history with the amount of coins they stake on the blockchain. While PoW has proven secure for over a decade, the security of PoS blockchains remains to be proven.

In our next article on mining, we will finally explain what the "puzzle" or "computationally expensive task" is, that we have been vaguely talking about up until now.

# Mining

In our last article on consensus mechanisms, we talked about Proof of Work (PoW) and introduced the general idea behind mining. In most articles mining is described as performing a computationally expensive task or solving a "puzzle". In this article, we want to explain what this puzzle is.

If you don't know what hash functions are you might want to read our article about them first. It will certainly help to understand mining.

As you might have picked up in our article on Consensus Mechanisms, not every blockchain has miners. Only in Proof of Work, there is mining and therefore miners. They work to secure the blockchain against attacks and protect the history recorded against changes.

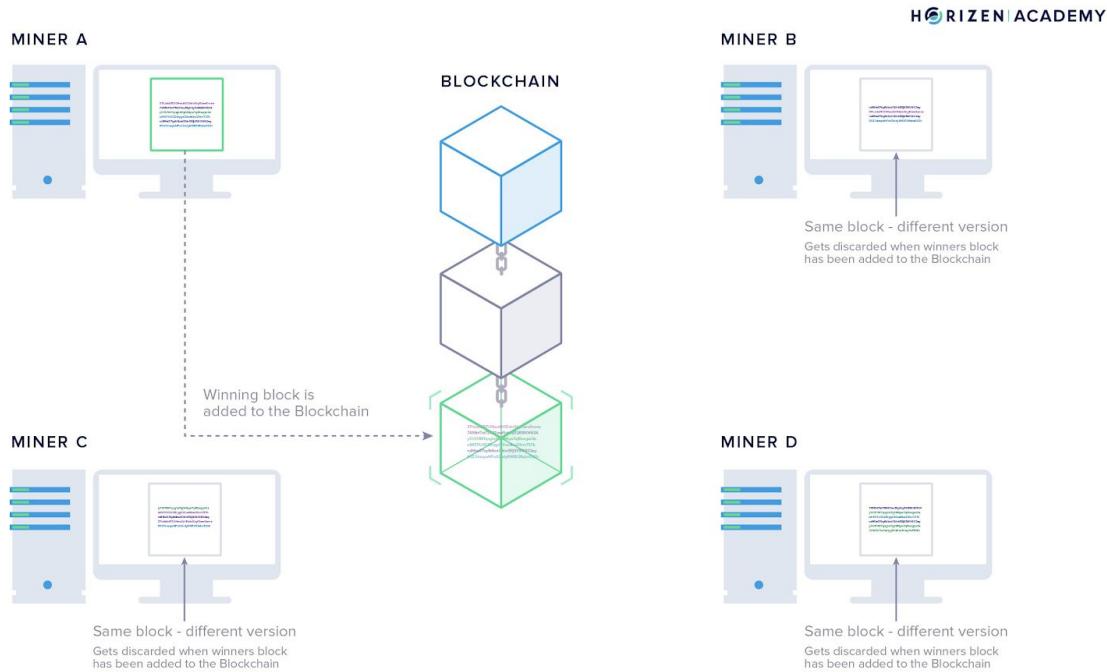
## What Does the Miner Do?

The main job of a miner is to collect all transactions that are broadcast to the network. The miner verifies if the transactions are valid according to the protocol and then places them in a data container - the block. The first transaction in a block is special, the *coinbase transaction*. In the *coinbase transaction*, the miner rewards his own address with the block reward, 12.5 BTC or 12.5 ZEN at the moment (actually 7.5 ZEN, as 10% of block rewards go to Secure Node and Super Node operators respectively, 20% to the Horizen Treasury). These coins did not exist before.

Usually, one of the criteria that is used to check the validity of a transaction is if the sender has sufficient funds. The coinbase transaction rules are slightly different. The protocol allows this first transaction to have no input, which means having no sender in this context. This is how the miners get to reward themselves for their work and how new coins are created. Every ZEN or bitcoin in existence started out as a block reward.

After the miner includes the coinbase transaction he places all other transactions he received in the block he is working on. Usually, the miner adds the transactions in the order they arrived in. In times of high network activity and full blocks (a block has a maximum file size, the *block limit*) miners might choose to include only the transactions with the highest transaction fees attached because they get to keep those as well.

Many miners are working on a block simultaneously. Each miner has a slightly different block, either because he received the transactions in a different order than his competitors, or because he chose to include a different set of transactions based on the attached transaction fees. The miner that solves the puzzle first, gets to extend the blockchain with his block, including the coinbase transaction that rewards him the newly created coins.



## Finding a Nonce

So what is the puzzle that miners are trying to solve? This is where we need to talk about hash functions again.

One of the important properties of cryptographic hash functions is being pseudorandom. You must not be able to predict a change in the output that results from a change of the input. As an example: If you are hashing the input "1" and get the output "00002" and hash the input "2" afterward, the second output better not be "00004" but something like "73968", a seemingly random (pseudorandom) number.

In order for a block to be valid, its hash needs to be below a certain *target*. This target is basically just a number and the block hash has to be smaller than that number. In this context, you will often hear the term *difficulty*. While the target is an actual value the block hash has to be below of, the

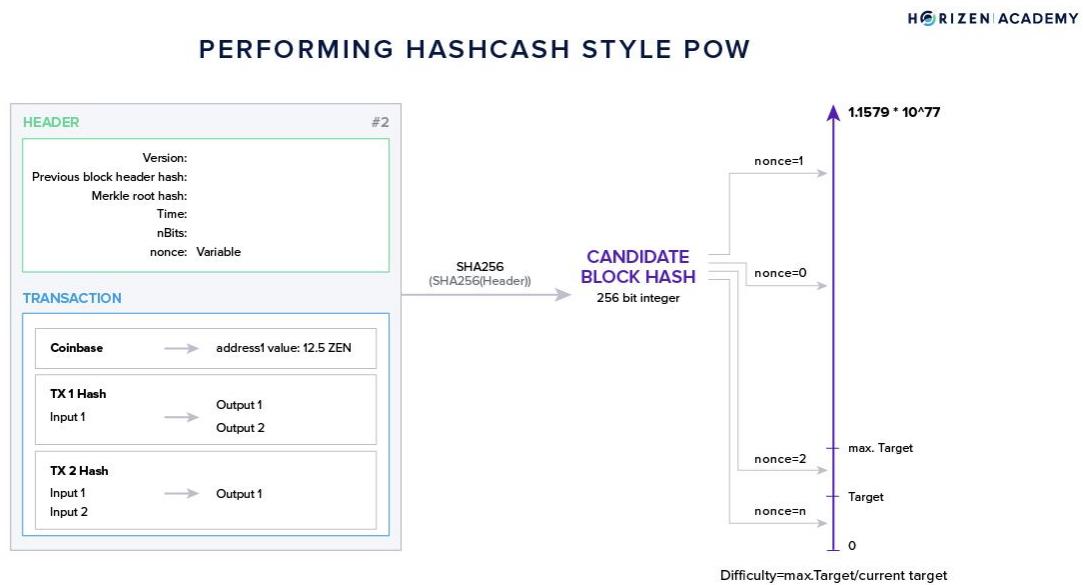
difficulty is a relative measure of the current target compared to the highest possible target. When the target is lowered, the difficulty is increased and it is harder to find a valid block hash.

Most of the data in the block is fixed, but there is a special data field in each block header, the *nonce*, that doesn't carry any important information. Its sole purpose is to be a variable (a number used **once**), that the miner can put different values in, to change the output of the hash function - the block hash.

What follows is a trial and error approach of getting the block hash below the target. The block hash can be interpreted as a regular number. The lowest number a 256-bit integer can represent is 0, the highest number is  $1.1579 \times 10^{77}$  -

115792089237316195423570985008687907853269984665640564039457584007913129639936

When a miner first attempts to solve the block, they will put a random value in the *nonce* data field. For the sake of simplicity, let's say the *nonce* value the miner starts with is "0". The block hash will be a seemingly random value in the possible range of the hash function.



So the process looks like this:

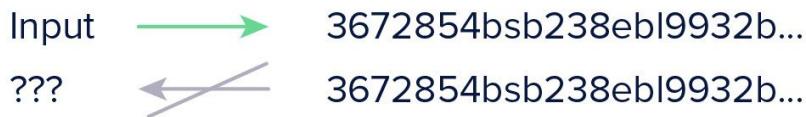
- The miner uses the hash function to calculate the hash of the block - the *candidate block hash* as we named it in the graphic above. In the example, the block hash using 0 as a nonce doesn't meet the target.
- The miner now increments the *nonce* by one and hashes the block again. The resulting hash is still above the target, so the miner increments the nonce once more. This happens over and over again.
- Finally, one of the miners will find a *nonce n* that produces the desired result: a block hash below the target.

The hash rate on the Horizen network is at 208.68 Mh/s at the time of writing. This means that all miners combined try 208,680,000 different values for the *nonce* every second. You can check the current value [here](#).

Miners can't cheat this process of trial and error because of the properties of cryptographic hash functions. Remember this graphic from our hash function article?

HORIZEN ACADEMY

## 1. ONE-WAY FUNCTION



## 2. PSEUDO RANDOM



## 3. COLLISION RESISTANT



## 4. DETERMINISTIC



There is no way to calculate a valid nonce from the desired output because the hash function is a one-way function.

It's also infeasible to approximate a valid nonce from prior outputs because the hash function is pseudo-random and you can't predict changes in the output from changes to the input. Every participant on the network can easily verify if the solution is valid because the hash function is

deterministic and will produce the same result for every node that verifies the block. Performing a single hash operation is also very quick.

If you would like to play around with hash functions you can do it [here](#). You can try to find the lowest number, that produces a hash with a leading zero. This is what miners do all day long, but they are looking for hashes starting with many more zeros.

## Block Time

Every Proof-of-Work protocol defines a block time, an interval within which a new block should be created. With Horizen this is 2.5 minutes, with Bitcoin 10 minutes. When a miner solves a block he broadcasts it to the network immediately, because he wants to collect his block reward. This is why it doesn't take exactly 10 minutes to create a new block. 2.5 minutes is the average time that it takes to find a valid nonce and in turn a valid block hash. Some blocks are solved quicker and some take a bit longer. If more miners join the network and the hash rate increases, they will find a valid nonce faster on average. The protocol evaluates the average block time and adjusts the difficulty accordingly every 8064 blocks. If it takes less time on average to solve a block the difficulty will increase, if it takes longer the difficulty will decrease.

## Hardware

When Bitcoin was released in 2009, people were using their PC's CPU, the central processing unit, to mine bitcoin. It is the most versatile processing unit in a computer but not the most efficient. After a while, people switched to using GPUs (Graphical Processing Unit) for mining. GPUs are more specialized and efficient for certain tasks, at the cost of being less versatile. Nowadays, the mining industry is running on ASICs (Application Specific Integrated Circuits). It is the most efficient hardware to mine cryptocurrencies because it is designed with one goal in mind, calculating hash values with a specific hash function. For hardware versatility and efficiency have trade-offs. If you want something that can perform many tasks, it will not be very efficient. If you want something that is super efficient at one task, it will not be very versatile. You can find a great article on mining hardware [here](#), written by David Vorick, lead developer of Sia.

## How Does This Protect the Network?

By now you know that it requires a huge amount of work to solve a single block in a blockchain. You also know that each block references the preceding block by including its block hash, therefore

chaining them together. You also know, that changing one little piece of information in a block, will alter its block hash completely.

If an attacker wanted to tamper with any record on the blockchain, they would have to find a valid nonce for the block they edited, as well as all the following blocks. The attacker would have to do all this by themselves, and at a faster rate than the network is performing the mining process. The longest chain rule that we talked about in our last article on consensus mechanisms determines which branch of the blockchain is the valid one in case a fork occurs. As long as the attacker does not control a majority of the hash rate, he won't be able to change the blockchain.

A great tool to understand the tamper-proof character of the blockchain is this [Blockchain Demo](#). We encourage you to play around with it!

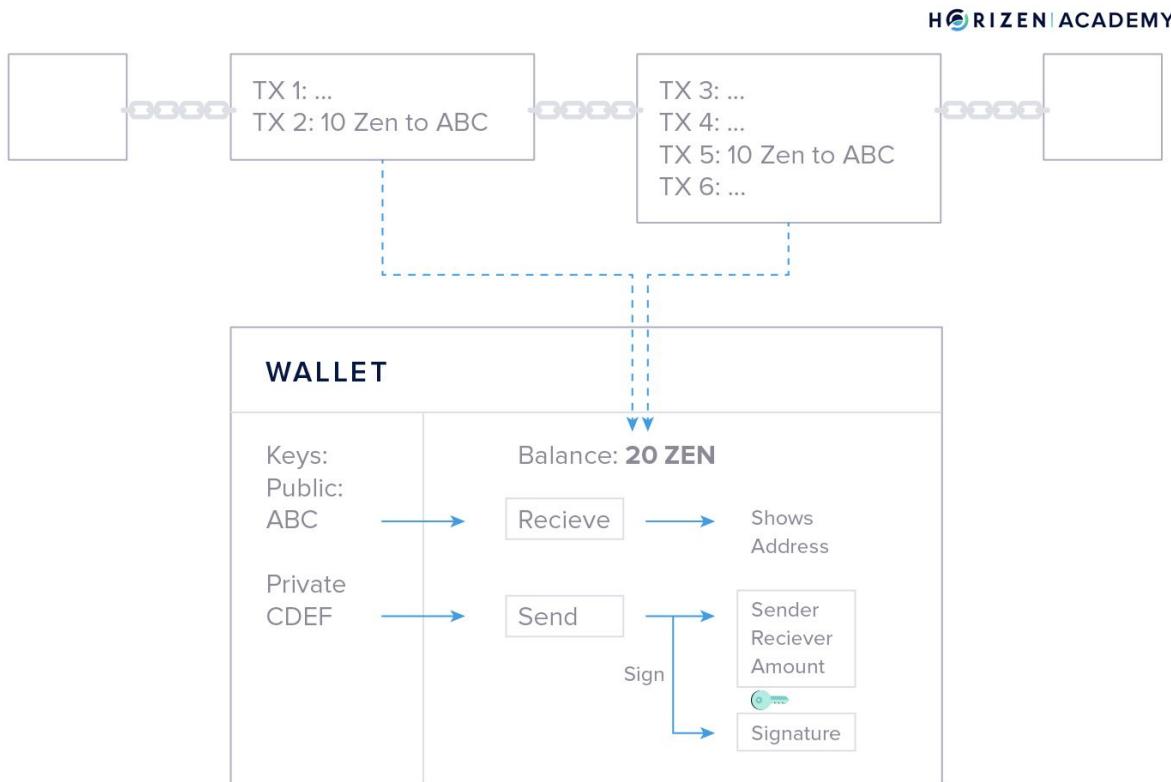
## Summary

The miners of a Proof-of-Work blockchain secure the network with their computational power. Adding a block to the blockchain requires a large number of random guesses to find a nonce that will produce a valid block hash. The block hash must be below the current target. It is extremely difficult to tamper with data on the blockchain. An attacker has to redo the work of finding a nonce that results in a valid block hash all by themselves. Not only does the attacker have to do this for the block they want to manipulate, but also for all subsequent blocks until the malicious chain is the longest chain and will be recognized as valid by the entire network. Because this type of attack is so incredibly difficult, blockchains are considered to be the most secure data structure that we have at our disposal today.

This was the last article of our chapter on how a blockchain works. Next, we will talk about Wallets, a tool that helps you manage your keys and create transactions.

# Wallets

A wallet is an app for generating, managing, and storing cryptographic keys - your public and private key. You can check your balance, receive, and send funds with a wallet.



If you are unsure about a wallets main functionality you can read our introduction to wallets. The main differentiator between the different types of wallets is the physical location your keys are stored in.

## Your Mnemonic Phrase

Your mnemonic phrase is a backup of your private key that is used by most wallets. It is a list of random words given to you when creating a wallet, usually 12 or 24. If you break or lose a device with a wallet - no matter if mobile, desktop or hardware wallet - your mnemonic phrase is usually your last line of defense against a loss of funds.

This implies, that any attacker that gets their hands on your recovery phrase will be able to do the same. Therefore, you must protect your mnemonic phrase as well as you would protect your funds themselves.

**Important Note:** You should write the phrase of words down on paper or save them in any analog format you see fit, but do not save them as a text file on your computer or a screenshot. You don't want to make it too easy for any potential attacker to steal your money.

## The Different Types of Wallets

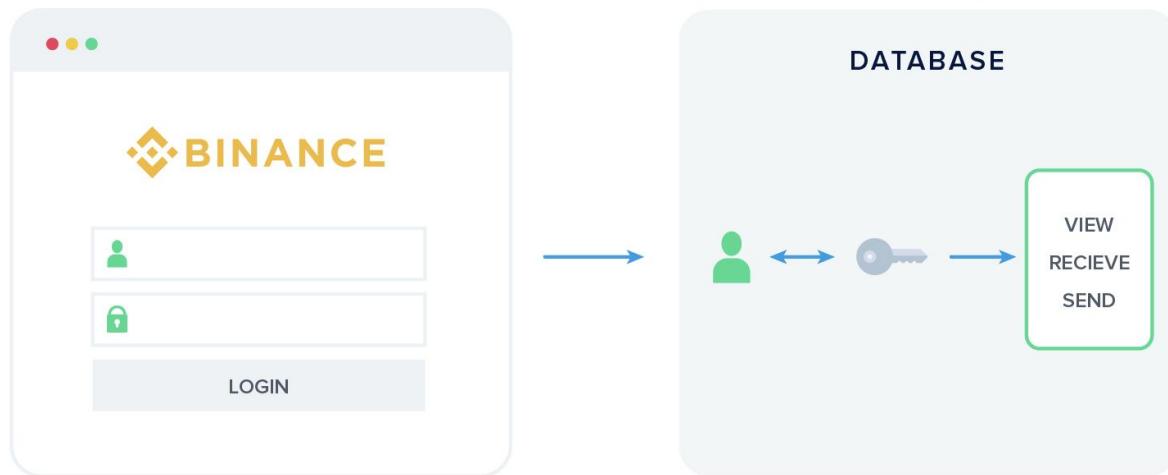
In this article, we want to give you an overview of what types of wallets there are and help you find the right wallet for you.

HORIZEN ACADEMY

Type of Wallet	Location of Your Keys	Where the Software is Running	Examples
<b>Web Wallet</b> hosted	The keys to access your funds are stored by a trusted third party	On the server of a third party	Most exchanges e.g. Coinbase, Binance, Bittrex etc.
<b>Web Wallet</b> non-hosted	You store your keys. You have different options of how to store them	Runs in your browser	MyEtherWallet
<b>Desktop/Mobile Wallet</b>	The keys are stored on the device you are using your wallet on	Runs a separate program on your device	Sphere by Horizen, Swing, Arizen, Coinomi, Paytomat
<b>Paper Wallet</b>	Keys in their pure form. Printed on paper	Can be used with all the types mentioned above	Paper wallet generators available for most coins
<b>Hardware Wallet</b>	Keys are stored on a specific device and never become visible	The interface runs in a browser or as a separate program. Signing is done on the wallet itself	Ledger, CoolWallet, Xeeda, Trezor etc.

## Hosted Web Wallets

We can distinguish between hosted and non-hosted web wallets. With hosted web wallets, your keys are stored online by a trusted third party. These parties are mostly exchanges such as Coinbase, Binance or Bittrex. When you create an account with these entities they will create an entry in their internal database linking your account to a set of key pairs for the different coins they have listed.



An advantage with a hosted web wallet is the option to recover your password in case you forget or misplace it. Losing your private keys (together with your mnemonic phrase) leads to a loss of funds in all other cases.

While this may sound reassuring, there are some drawbacks with hosted wallets (also called hot wallets/cloud wallets). Remember: if you don't control your keys, you don't control your funds. There is always a third party risk, no matter how trustworthy the party might seem.

First, they are a more attractive target for hackers than individuals because their honeypot is much bigger.

Second, a sudden change in regulation might not allow them to have you withdraw your funds in a worst-case scenario. It is unlikely, but definitely not impossible.

Thirdly, there is always a chance of an entity going bankrupt or stealing money. With the major exchanges like the ones mentioned above this risk seems small, but it does exist. Just ask former [Mt. Gox](#) customers. For the reasons above we do advise everybody to store the keys to their funds themselves. This means storing them in a wallet where you have control over your private keys.

There is a responsibility that comes with being in charge of the safety of your funds yourself, but enabling you to do this is one of the main motivations for the existence of cryptocurrencies!

You will need to keep some funds on an exchange permanently if you plan to trade often. If you want to do this right, then your level of expertise should be exceeding this article by far anyways.

## Non-Hosted Web Wallets

Besides hosted web wallets there is also a range of non-hosted web wallets. The most popular non-hosted web wallet is likely MyEtherWallet, which can store Ether (ETH) and all ERC-20 tokens (tokens that are "living" on the Ethereum blockchain). Those wallets provide an interface to check your funds or create transactions in your web browser, but you have to provide the keys with each login.

### How would you like to access your wallet?

- View w/ Address Only
- MetaMask / Mist
- Ledger Wallet
- TREZOR
- Digital Bitbox
- Secalot
- Keystore / JSON File [?](#)
- Mnemonic Phrase [?](#)
- Private Key [?](#)
- Parity Phrase [?](#)

There is a range of options to access your wallet with MyEtherWallet (often abbreviated as MEW). The first option requires your address but only lets you view your funds.

MetaMask is a browser plugin that provides the option to make ETH payments within your browser and the ability to login to MEW. It also provides a function detecting phishing sites and warning you when you are about to open one.

The next couple of options, Ledger Wallet, Trezor, Digital Bitbox, and Secalot are hardware wallets. We will get to those later in the article.

Accessing your wallet with a keystore /.json file is possible but not recommended. The file contains your private key and when you create your wallet you have the option to download it. If it gets into the wrong hands they will have access to your funds so saving it on your desktop is not the ideal

solution. If you want to use this method, you should encrypt the .json file and store it on a separate device like a USB drive. To use it, connect the drive, decrypt the file, then select the file in your browser and voila. After that, you can disconnect your storage medium of choice again.

The last two options are more or less the same with regards to safety. You can either enter your private key directly or your mnemonic phrase (which yields your private key when hashed) which is both problematic if your machine is compromised.

In conclusion, a non-hosted web wallet is quite convenient and just as secure, as the method, you choose to provide your keys with.

## Desktop and Mobile Wallets

If you start off with the question "where are your keys?" the desktop and mobile wallet will give you the same answer: on the device. Phones and tablets are more powerful than ever, the difference between a desktop and a mobile wallet is marginal. It is also arguable whether one is safer than the other.

By now you know the tasks a wallet performs: viewing, receiving and sending. If you want to use crypto for everyday transactions there is almost no way around keeping some funds in a mobile wallet. As mentioned before, when creating your wallet you will get a mnemonic phrase that you should keep safe. Usually, there is a PIN, password or Face-/Touch-ID protection to access the wallet. You should never keep more funds in a mobile wallet than you are willing to lose. It's the same as with cash: you don't carry around all your money in a wallet. You withdraw as much as you are comfortable handling in cash and keep the rest in your account (or under the mattress :P).

With desktop and mobile wallets, there is a choice between single- and multi-currency wallets. Those should be rather self-explanatory terms. The former allows you to store one coin, while the latter supports multiple currencies. Some of the more popular examples for desktop include Coinomi and Sphere by Horizen. For mobile, there are Coinomi, Mycelium or Paytomat to name just a few.

If you got your first coins on an exchange, I would recommend you to transfer your funds out of the exchange (hosted web wallet) onto a desktop, paper, or hardware wallet. Send a fraction first to make sure everything works as planned. If your first transaction works then you can send the rest. You are now protected from third-party risk, but have full responsibility for your funds yourself.

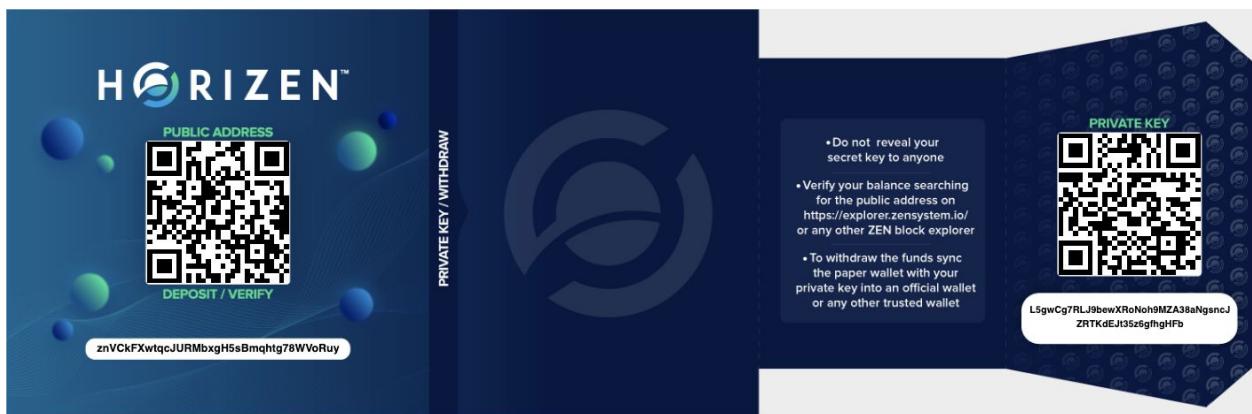
## Paper Wallets

Grabbing our golden thread again and asking "where are the keys" give you a simple answer with a [paper wallet](#): in your hand! A paper wallet is your public and private key pair printed on paper.

Almost every cryptocurrency offers a paper wallet generator. To create a key pair you generally first have to create some entropy (a term for disorder), in other words: you want your keys to be as random as possible. This is mostly done automatically, but sometimes you will find features where you have to [randomly move your mouse](#) or hit keys on your keyboard to create randomness.

When printing your paper wallet you shouldn't use a shared printer like the one in your office. In a best-case scenario, the printer doesn't even have an internet connection. Printers usually keep a copy of the files they printed last, and an attacker might exploit this.

You will end up with something looking like this after printing the wallet.



There is only one thing left to do: send your coins to the public key. After that, you have a perfect gift or long term storage for your coins. The main risk with a paper wallet is you actually losing or destroying the wallet by accident. If you don't have a mnemonic phrase to recover the private key you are at risk of losing all funds on the wallet by accident. So choose wisely where to store your paper wallet. Print several copies if you feel uncomfortable having only one and store all of them in separate, safe places.

## Hardware Wallets

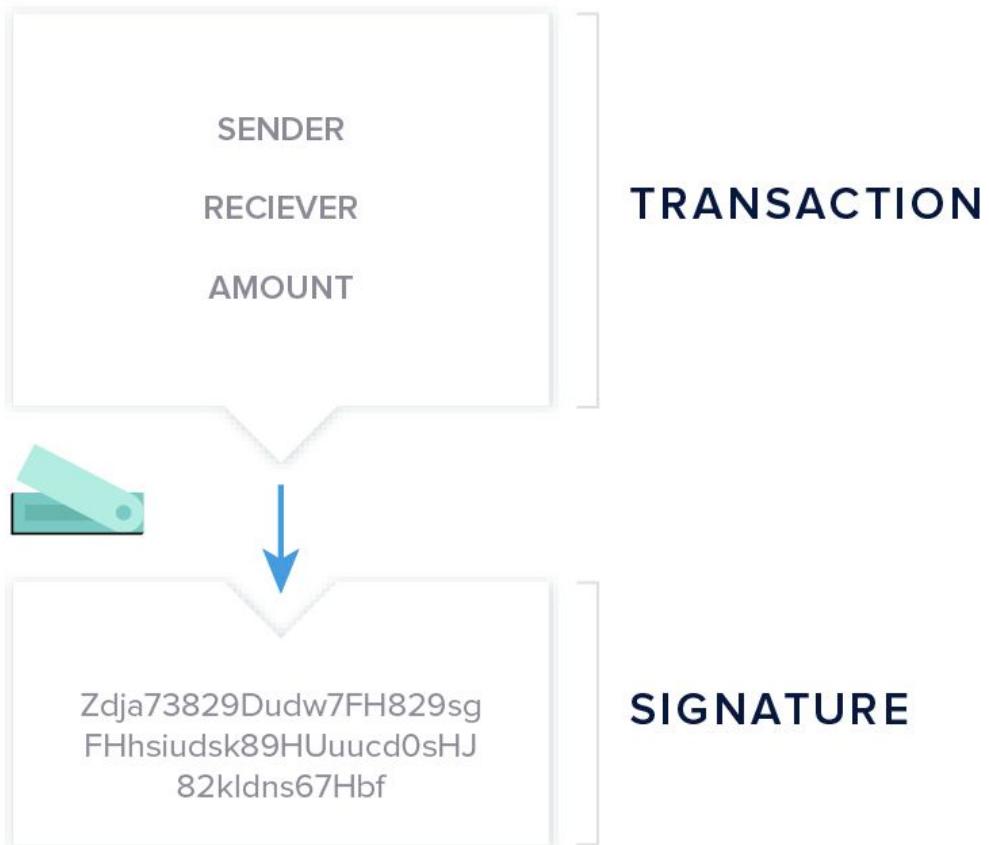
Moving on to everybody's darling: hardware wallets. With a hardware wallet, your keys are stored on the device in something called the "secure element". The secure element is a place to store data (here, keys) that cannot be directly accessed by your computer or any other device even when it is connected. Although it does look like a simple USB drive, it can actually do a little more than just

providing storage for your keys. To use a hardware wallet you usually have a few options of which interface to use with it. Like MyEtherWallet, a few other wallets offer hardware wallet support. Additionally, you have the native wallet apps provided by the producer. In the case of Ledger, for example, the native App is called Ledger Live.

### How Does a Hardware Wallet Work?

The interface generates an address when you want to receive funds. Using this feature is pretty straightforward: if you click the receive button the process runs in the background and the address is displayed for you to share with the sender.

If you want to send money the app creates the raw transaction that needs to be signed. The unsigned transaction is now sent to your hardware wallet, where it gets signed with your private key. The signature is then returned to your computer and the complete transaction including the signature broadcasted to the network.



Your private key(s) do not leave the device, so they are not visible to the computer you are using your hardware wallet with at any time. This is why a hardware wallet is considered the most secure way of storing crypto, especially large amounts.

If your device ever breaks, you have your mnemonic phrase as a backup. At the risk of being repetitive: your mnemonic phrase, under all circumstances, must stay private and in a secure location. A copy at a trusted family member or in a bank vault might be a good idea in case of a fire, flooding or a playing dog.

## Summary

There are many ways to store your cryptocurrencies. Usually, there is a trade-off between convenience and security. The most important question is: where are the keys? A wallet is only a piece of software, an interface, that helps you perform the basic functions of cryptocurrencies: view your balance, create an address to receive funds, and create transactions to send funds.

With a hosted online wallet you are trusting a third party to handle your keys. You have the option to recover your password if misplaced, but there is always a significant third-party risk. If you don't control your keys, you don't control your funds!

With desktop, mobile, paper or hardware wallets you own the keys and nobody but yourself is responsible for keeping them safe. If your device breaks you have a mnemonic/recovery phrase to recover access to your money. The mnemonic phrase is as sensible as your private key itself and if it gets in the wrong hands, your money can be stolen. This should not scare you, but make you cautious.

This whole movement of cryptocurrencies aims to give you back the power over your money, but...



# Transactions

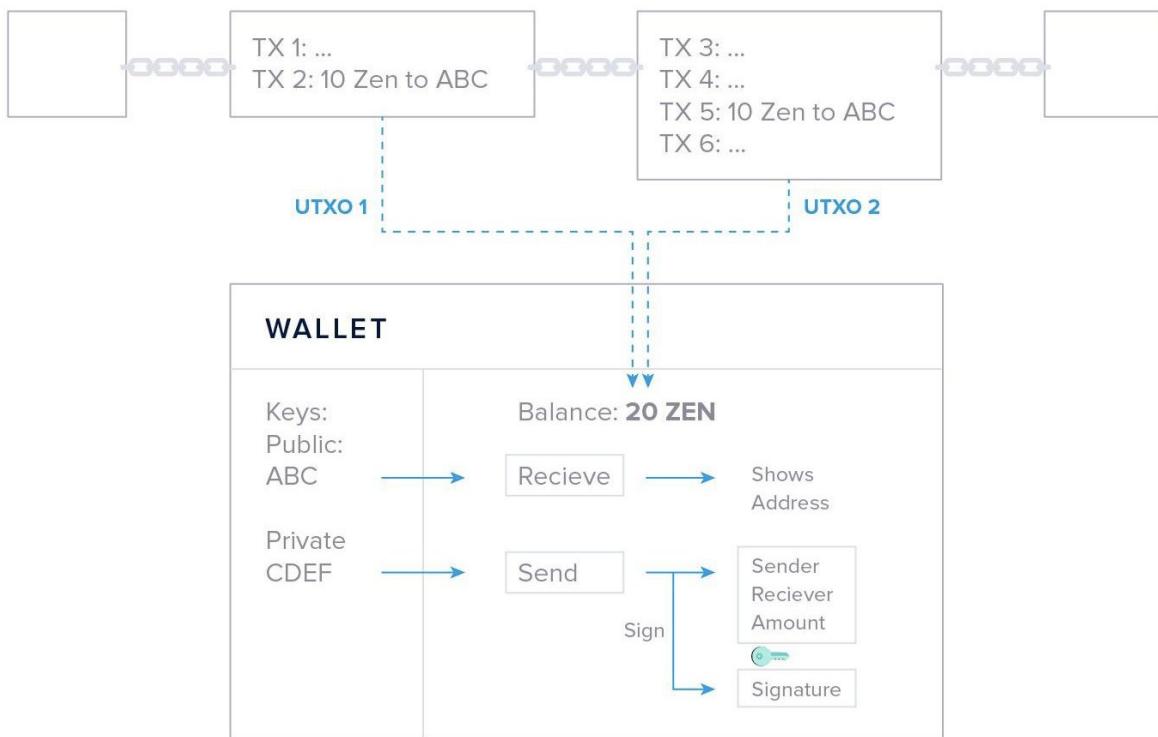
You should have a pretty good understanding of blockchain if you have read through our first chapters in the Advanced Level and made it this far. After we have covered the ideas and value propositions of blockchain in the first chapter we looked at the concepts that make blockchain work. In this chapter, we will look at a small subunit of the blockchain, a tiny piece in the puzzle. The blockchain is made of many blocks which in turn are containers for transactions.

## The UTXO Model

Before we get into how transactions (TXs) on a blockchain work we want to introduce you to the UTXO (Unspent Transaction Output) model that we already talked about this in the Beginner Level article on transactions

When you think about how your bank does the accounting for your bank account it is pretty intuitive. You hold a certain amount of money in your account which has an account number. If you receive money the amount is added to your balance. If you spend money, then the amount you spend gets subtracted from your balance. With cryptocurrencies, the accounting works a little different.

The blockchain does not create an "account" for you to maintain a balance. There is no final balance stored on the ledger. The blockchain only stores individual transactions and to check your balance, there is an additional step involved. Your wallet does this automatically whenever you open it. What happens in the background is that your wallet scans the ledger for all transactions to your address(es) and adds them up.

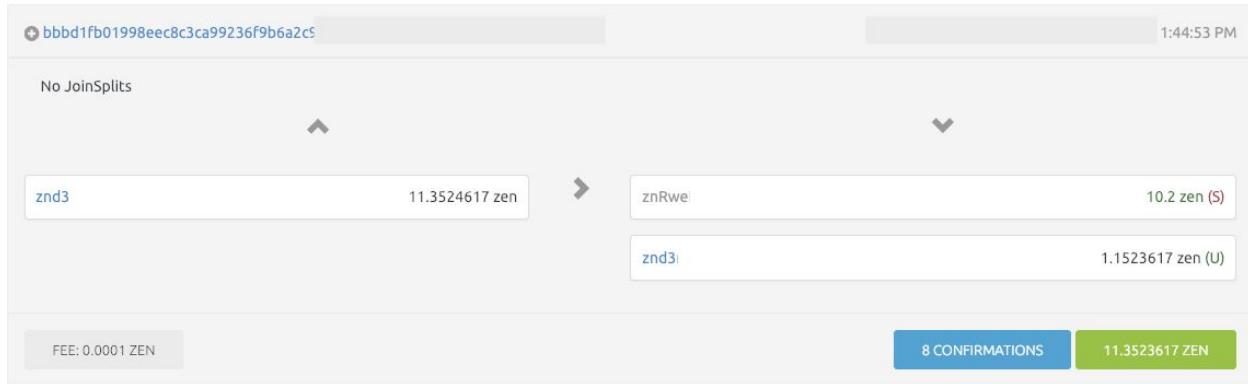


Each transaction on the blockchain has one or more inputs and one or more outputs. Let's have a look at an actual example throughout a series of four transactions:

Usually, a [block explorer] will show you the most recent transactions first. For this example, we will go through the transactions as they happened - in chronological order.

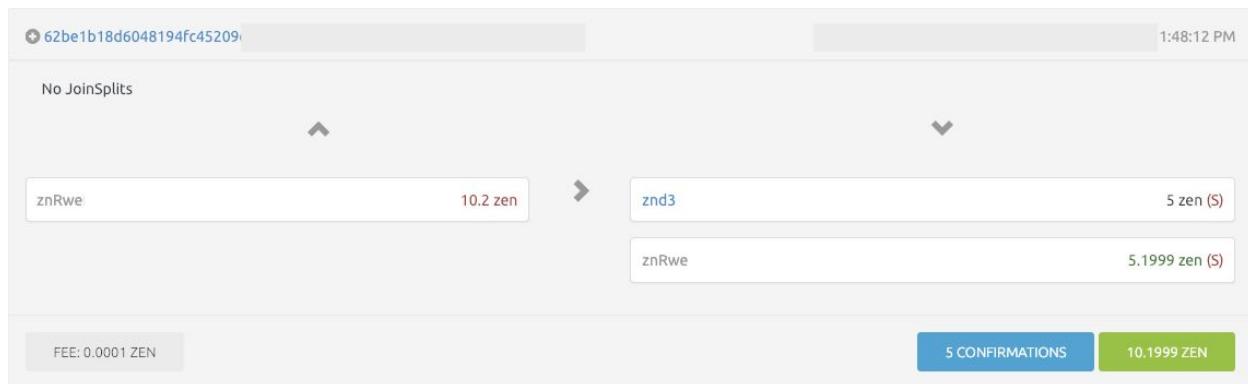
We created this simple example, only involving two different addresses. We shortened the addresses for better readability. The address we are concerned with here is the grey one: znRwe... Let's say this is Bob and the other one (blue), is Alice.

## Bob Receives His First Transaction



In the [first TX](#) (above), Bob's address (znRwe) is funded when he receives 10.2 ZEN. The TX has one input and two outputs. The first output (10.2 ZEN) is what Alice actually wanted to transfer to Bob, the second output is called the *change output*. The input that Alice was using, was an output of a transaction she received before. When she still had her money untouched, it was an *Unspent Transaction Output* (UTXO). A spent transaction output is indicated by the (S), a UTXO is indicated with a (U) following the amount. But we will get back to this in a minute. Alice didn't have a UTXO that was exactly 10.2 ZEN so she used one that was larger and sent the remaining ZEN back to herself, just as you would receive change in a store if you were to pay \$45 with a \$50 bill.

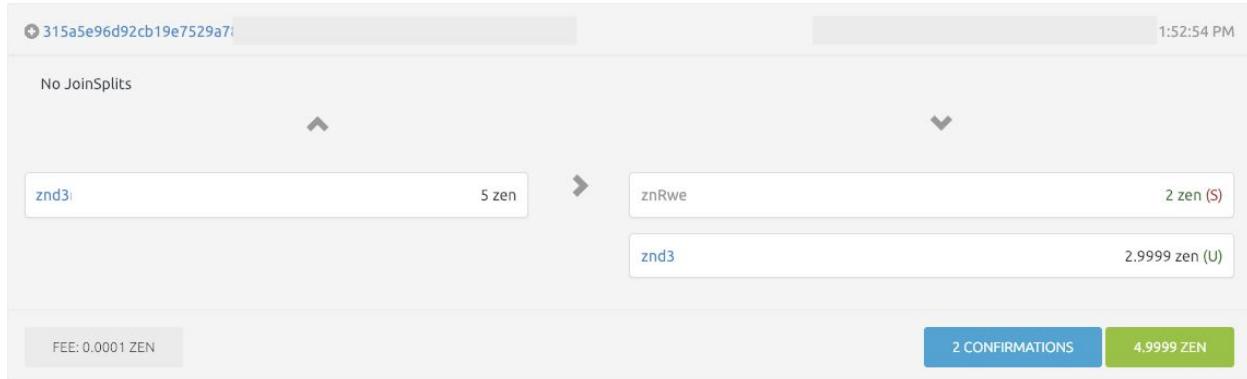
## Bob Sends His First Transaction



In the [second transaction](#), Bob spends his UTXO of 10.2 ZEN and creates a transaction with two new UTXOs: one of 5 ZEN to a different address and one of 5.1999 ZEN back to his own address -

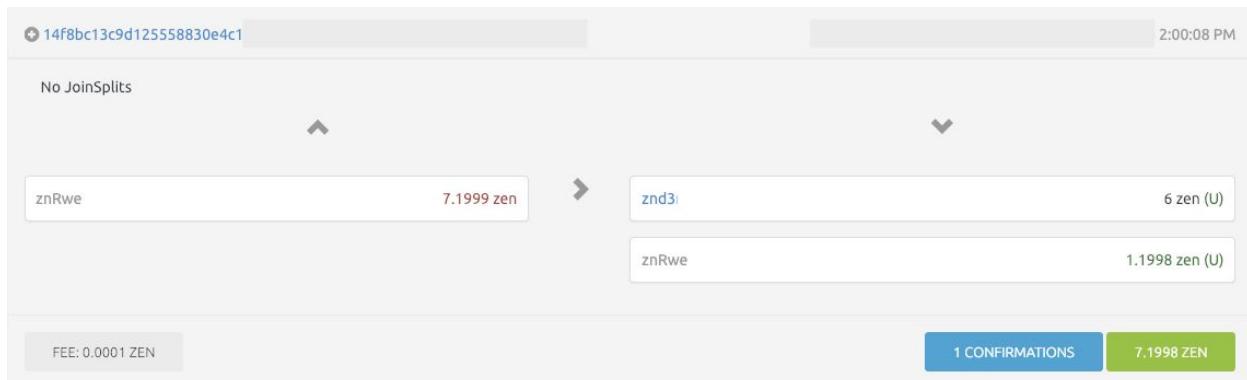
the change output. The difference between inputs and outputs - 0.0001 ZEN - is consumed as the transaction fee. He now owns 5.1999 ZEN on his znRwe... address.

## Another UTXO



In a [third transaction](#), Bob receives another 2 ZEN, increasing his balance to 7.1999 ZEN. He now has two UTXO's at his disposal for further transactions: one of 5.1999 ZEN and another one of 2 ZEN. If he were to open his wallet at that point, it would show him a balance of 7.1999 ZEN by looking at all transactions on the blockchain, filtering out the ones that involve his address and then adding up all unspent outputs.

## Spending Two UTXOs at Once



In the [last transaction](#) of this example, Bob wants to spend 6 ZEN. Neither of the two UTXO's he has at that point is sufficient for that purpose. Although the block explorer shows only one input for the last transaction there were clearly two inputs used: the 5.1999 ZEN and 2 ZEN one from the two examples above.

Combined both UTXO's are worth 7.1999. Bob created two outputs with it: the 6 ZEN output he actually wanted to spend and an additional output for the change of 1.1998 (1.1999 minus the transaction fee of 0.0001). You can see that by now both TX outputs are spent, indicated by the (S) next to them in the second and third screenshot.

## Summary

Most blockchains use the UTXO model for accounting. There are a few exceptions, such as Ethereum, which actually uses an account model. The output of a transaction addressed to you is what you will use as an input to create an outgoing transaction.

When people ask, what a ZEN or Bitcoin actually is, "a UTXO" would be the accurate answer. An unspent transaction output or UTXO that you can unlock with your private key IS your coin. There is no abstraction on top of this. It might take a moment or a second read to get familiar with this, but it's a neat and simple concept and we hope the example above helped you understand what the UTXO model is. If you have fully understood the UTXO model it will help a lot with understanding the following articles.

The next article covers the block explorer, a tool that allows you to access information on a blockchain, like how a web browser lets you access information on the internet. It is the tool we took the screenshots above with.

## Block Explorer Continued

The block explorer is a graphical tool to view and explore data on the blockchain. There is a block explorer available for almost every public blockchain. It allows you to browse the history of a given chain. It might be most interesting to go to the very start of blockchain technology and have a look at Bitcoin in this example.

If you know how to navigate one block explorer, you will know how to navigate all of them. It is quite intuitive and helps to understand how a blockchain is structured and how it works. To really appreciate this article you should be familiar with the basics of a blockchain, transactions, and mining. You can always skip back to one of our previous articles covering these topics if you are unsure about something. If you are familiar with how a blockchain works, this article should help you to connect some dots.

### What You Will Find in a Block Explorer

The information that you will find in a block explorer is structured either by blocks, addresses or individual transactions. Let's look into a first example to make all this more tangible.

LATEST BLOCKS							SEE MORE →
Height	Age	Transactions	Total Sent	Relayed By	Size (kB)	Weight (kWU)	
539772	6 minutes	1547	4,696.10 BTC	SlushPool	882.23	2,917.89	
539771	12 minutes	1035	3,550.71 BTC	AntPool	532.23	1,876.79	
539770	21 minutes	426	675.26 BTC	F2Pool	179.94	577.37	
539769	23 minutes	550	1,513.18 BTC	BTC.TOP	302.54	1,047.33	

There are a number of block explorers out there, oftentimes several versions for the same blockchain. Since we want to go all the way back to the beginning of cryptocurrencies in this example we will look at Bitcoin and use one of the [most popular explorers](#) out there.

You will see an overview of the most recently created blocks on the landing page. Let's go through what we see here:

- **Height:** the height is the number the block carries. The very first block created was block #0, after that came block #1 and so forth. As you can see at the time of writing there have been quite a few blocks mined.
- **Age:** Is the age of the block. It's self-explanatory, but there is a piece of interesting information between the lines: you might know that the block time, the interval in which new blocks are mined, is 10 minutes with Bitcoin. How come it took only 6 minutes to mine block 539.772 and only 9 minutes for block 539.771? This is because the block time is an average. There will always be blocks created slower and a few that will be produced a lot faster, but on average it will take 10 minutes. In our article about mining, we explain how the block time is adjusted using the *target*.
- **Transactions:** the number of transactions included in this block.
- **Total Sent:** The 1547 transactions that took place add up to almost 4700 Bitcoins sent.
- **Relayed By:** The miner (or mining pool) that solved the block and earned the block reward.
- **Size:** the amount of storage the block takes up
- **Weight:** another metric that refers to the size of a block. It is a more technical metric but as you can see: the more transactions in a block and the bigger the file size of a block, the bigger the weight.

## The Genesis Block

Let's have a look at a single block. The height of the very first block is 0 as we said before. If we want to look at the very first Bitcoin block, we can use the search function and enter "0". If you haven't opened the site yet [this link](#) will take you directly to the *genesis block*. The very first block of a given chain is usually called the genesis block.

## Block #0

Summary	
Number Of Transactions	1
Output Total	50 BTC
Estimated Transaction Volume	0 BTC
Transaction Fees	0 BTC
Height	0 (Main Chain)
Timestamp	2009-01-03 18:15:05
Received Time	2009-01-03 18:15:05
Relayed By	Unknown
Difficulty	1
Bits	486604799
Size	0.285 kB
Weight	0.896 kWU
Version	1
Nonce	2083236893
Block Reward	50 BTC

## Transactions

4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b	(Size: 204 bytes) 2009-01-03 18:15:05
No Inputs (Newly Generated Coins)	 1A1zP1eP5QGefi... (Genesis of Bitcoin) - (Unspent) 50 BTC 50 BTC

Let us go through the information we can find here. The genesis block contains only one transaction. This is the minimum amount you will see since every block has a so-called *coinbase transaction*, the transaction that is used to issue new bitcoins to the miner that solved the block to reward them for their efforts.

The reward for mining Bitcoin, in the beginning, was 50 BTC. The block reward reduces by half every four years (more precisely every 210,000 blocks). This has happened twice so far. A miner today receives 12.5 BTC per block.

Other than the transaction that rewarded the miner (Satoshi Nakamoto himself in this case) there is no transaction in this block, that is why the estimated transaction volume is 0. Today there is a transaction fee attached to most transactions, but this wasn't always the case, especially when the network was still in its infancy.

We have talked about the height before so we will get to the timestamp now. The first bitcoin was mined on the 3rd of January 2009, the birthday of the very first cryptocurrency.

The "Relayed By" field shows who mined the block. Today's mining pools and their reward addresses are mostly known. This is how the block explorer "knows" who mined the block: it compares the address that received the newly generated coins (1A1zP1...) in the coinbase transaction with a number of known addresses from mining pools. In this case, we do know who the miner was, although the block explorer doesn't tell us: Satoshi Nakamoto himself.

We talk about the *difficulty* in our mining article in more detail. The difficulty describes the threshold the block hash has to be smaller than or equal to in order to be considered valid. It can be interpreted as the number of leading zeros the block hash must have. As you can see on the right, the genesis block hash had 10 leading zeros. Today (as of the time of writing) blocks like block 539.772 need to have at least 18 leading zeros.

*Bits*, *Size*, and *Weight* all refer to the size of the data of a given block, not too spectacular.

*Version* refers to the software that was run at the time the block was created.

Now we get to another interesting information: the *nonce*. When miners try to solve a block, what they are doing is putting different values in the Nonce (**N**umber used **o**nce) field and calculate the hash of the block. If the result doesn't have the minimum amount of leading zeros required (most often it will have none at all) they increment the nonce and hash the block again. This is repeated until a miner finds a nonce that produces a hash below the target. In the case of the very first block, Satoshi's computer tried more than 2 Billion values (if he started at 0) before he found a value that produced the desired result. He received the very first 50 bitcoin as a block reward for this.

## The First Bitcoin Transaction

What you will most likely use a block explorer for is checking a transaction status. You can find a given transaction either by searching for one of the addresses involved (the sending or receiving address) or by the *transaction ID* (TXID). Most wallets will show you the transaction ID for all of your transactions.

If we want to look at the very first cryptocurrency transaction ever done we need to go to Block #170, where Satoshi Nakamoto sent Hal Finney 10 BTC.

## Transactions

b1fea52486ce0c62bb442b530a3f0132b826c74e473d1f2c220bfa78111c5082		2009-01-12 03:30:25
No Inputs (Newly Generated Coins)	→	1PSSGeFHDnKNxiEyFrD1wcEaHr9hrQDDWc
		50 BTC
50 BTC		
14184fc596403b9d638783cf57adfe4c75c605f6356fb91338530e9831e9e16		2009-01-12 03:30:25
12cbQLTFMXRnSzktFkuoG3eHoMeFtpTu3S	→	1Q2TWHE3GMd6BZKafqwxXtWAvgFt5Jvm3 12cbQLTFMXRnSzktFkuoG3eHoMeFtpTu3S
		10 BTC
		40 BTC
50 BTC		

It is the first block on the bitcoin blockchain that contains more than one transaction. We can find the coinbase transaction on the very top. Below is the first transaction as you would expect it. One person sending coins to another one.

## Summary

We hope this article helps you navigate the block explorers of this world. We encourage you to play around with this tool for a while. I know it helped me a lot with understanding how the information in a blockchain is organized and I'm sure it will do the same for you.

## Atomic Swaps

If you want to exchange one cryptocurrency for another one you will most likely go to one of the centralized exchanges. First, you have to check, whether the exchange offers the trading pair that you are interested in. Then you transfer your funds to their address, which requires you to trust the exchange as a third party. Most exchanges offer BTC and ETH pairs. To swap two lesser-known cryptocurrencies or tokens, you will oftentimes have to buy bitcoin first which you will then convert into the desired cryptocurrency.

A special transaction type that we would like to talk about in this article is the Atomic Swap. It is a type of transaction that touches on the topics of scalability and interoperability at the same time.

Atomic Swaps are a technology that allows you to trade Peer-to-Peer without a third party. They also do away with the trust required for you to arrange a swap with an unknown user. Someone would have to send their funds first, and the counterparty could decide to not fulfill their side of the deal. Atomic Swaps enable peers to do a trustless exchange of different currencies according to predefined and agreed upon conditions without having to fear losing their funds.

## The Technology Behind Atomic Swaps

From our article on public-key cryptography, you know that a valid transaction requires a signature. This signature can only be created by the person that has access to the private key. When you send a transaction you usually sign it and broadcast it to the network afterward.

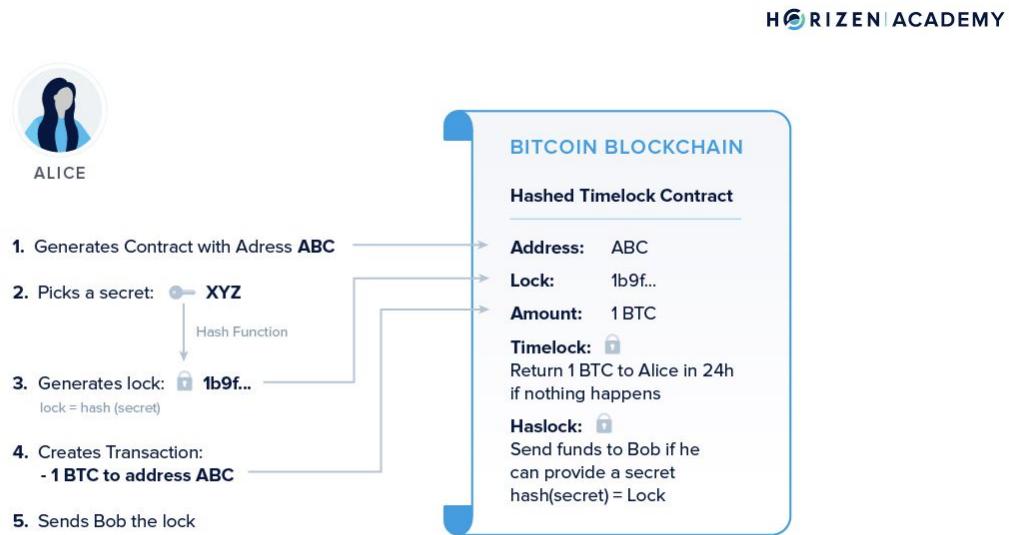
HTLCs are based on a technology called *state channels*. We explain those in detail in our Expert Level, but for now, all you need to know is that they allow you to exchange signed transactions securely. Only once the participants decide they are done transacting, the final *state* is broadcast to the blockchain.

The "Hashed" part of HTLC means that a hash serves as a lock for the contract, to protect it from a third party accessing it. The "Timelock" part refers to the contract having an expiration date.

Two conditions must be met to perform an Atomic Swap between two crypto assets: Both assets and their underlying blockchain need to support the same hashing algorithm, like SHA-256 in the case of Bitcoin and both blockchains need to support some kind of programmability, that allows an HTLC to be deployed.

## The Process

The process of an Atomic Swap would generally look like the following. Let's assume Alice has some BTC and Bob has some ZEN. Both agree to swap a certain amount of their assets.



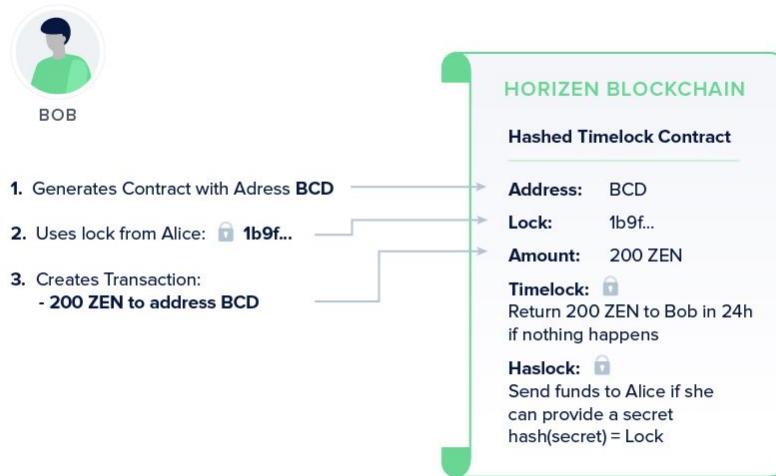
First, Alice creates an HTLC on the Bitcoin Blockchain that we will simply call contract. The contract comes with an address, otherwise, you could not interact with it. Next, Alice generates or picks a secret - in this example her secret is XYZ. Using a hash function she generates a lock (here: 1b9f...), which is simply the hash of the secret.

Now she deposits the amount of bitcoin she and Bob agreed to exchange in the contract where they are locked. Lastly, she sends the lock to Bob.

The contract can enforce two outcomes:

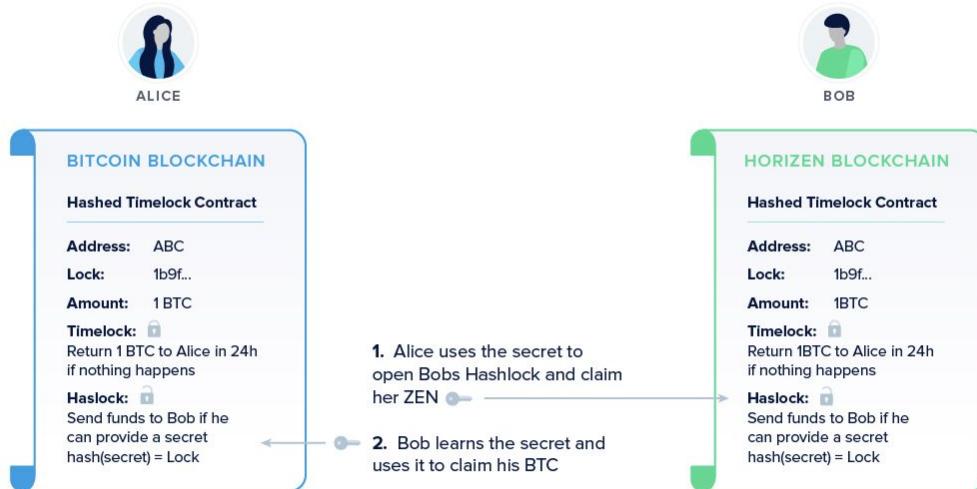
- If nothing happens for some time, say 24 hours, the money is returned to Alice. This is necessary, so she doesn't lose her money in case Bob never responds. It is the *timelock* component of the HTLC.
- If Bob can provide the secret, the contract will automatically transfer the bitcoin to his address. Because hash functions are one-way functions, Bob doesn't know the secret yet, although he knows its hash.

Now it's Bob's turn.



Bob also creates a Hashed Timelock Contract, but this time on the Horizen blockchain. His contract also has an address, *BCD* in this example. Alice sent him a lock (the same one she used) so he goes ahead and locks his contract with the same lock. Lastly, he deposits the amount of ZEN the two agreed to exchange. Just like the contract Alice created, Bob's contract can execute in two ways: either Bob gets refunded after a timeout period or Alice provides the secret and the funds are distributed to her address.

Now both contracts are set up on their respective blockchain. Alice has to take the next step within the locking period, otherwise, both are refunded at nothing happened at all.



Alice will now use the secret she chose earlier to unlock the hashlock of Bob's contract on the Horizen blockchain and the ZEN will be released to her. This is a public operation, auditable on the blockchain. Bob can, therefore, see the secret now, and uses it to unlock the bitcoin locked up in Alice's contract. By providing the secret the HTLC will automatically release the funds to Bob's previously specified Bitcoin address.

Without having to trust each other Alice and Bob have now successfully exchanged their coins, without having to involve a middleman. At no point could one of the two steal the other's money. When Bob receives the lock from Alice, he can verify that she used the same lock by looking at the Bitcoin blockchain.

If Alice would not redeem her ZEN, then both of them would automatically get refunded. And Alice cannot claim Bob's ZEN without Bob learning the secret.

## Atomic Swaps Today

There are few wallets that enable users to do Atomic Swaps, yet. So far there have been only a small number of Atomic Swaps completed. Here is a short list:

- The first on-chain Atomic Swap was done on September 20th, 2017 between Litecoin and Decred

- The first off-chain Atomic Swap occurred November 2017 between Litecoin and Bitcoin on the Lightning Network
- A detailed explanation of an Ethereum - Bitcoin Atomic Swap can be found [here](#)
- Most recently on the 7th of December 2018 TenX showcased an ERC20 to Bitcoin Lightning Atomic Swap using their open-source software COMIT

The technology is very promising, but to achieve adoption wallets will have to build a user-friendly interface to use Atomic Swaps. This will take a while, but the technology is very promising and could very well enable a new class of use cases for cryptocurrencies.

## Summary

Atomic Swaps present an alternative to the centralized exchanges used today. At no point is there a third party involved that has access to a user's funds. The exchange process is entirely trustless and almost instant. Hashed Timelock Contracts are the heart of Atomic Swaps. I suspect that it won't take long until Atomic Swap compatibility will be a necessary feature for a blockchain to gain serious adoption.

This was the last article on transactions in the Advanced Level. To learn more about how transactions work and what types of transactions there are you can visit our Expert Level. Here we will continue with an introduction to privacy-preserving techniques on the blockchain.

# Privacy on the Blockchain

One of the great value propositions of blockchains is the transparency that such public ledgers offer. There are many cases though where it is desirable to conduct private transactions. In this article, we want to introduce you to four concepts that aim to increase the level of privacy for transactions on public blockchains. It is a common misconception that cryptocurrencies are anonymous. Most of them are pseudonymous, meaning that real-world identities are represented by addresses. Addresses and identities can be connected through ongoing data analysis.

## Why Privacy?

There are many legitimate reasons to create private transactions on a blockchain. If you have a medical condition and need to purchase your prescriptions on a regular basis you have good reason to do these transactions privately. If you have a business, you don't want to reveal your revenue streams to your competition and if you are buying a present for your spouse, you might not want him or her to see it before they actually get the present. There are many good reasons to wish to transact privately, and we believe that privacy is and should be treated as a basic human right.

For this article, we assume that you are familiar with the UTXO model that many blockchains use for accounting. If you are not, feel free to check out our article on it before you continue reading.

## Change Addresses

<a href="#">1LDHjMRwEv4AvvUYojswpNxyrnSyjcFwWIR</a>	0.015 BTC		<a href="#">1KVPHe4CpQBemEELUZ5d1VZdutKXGuXQTp</a>	0.01442771 BTC	
			<a href="#">1LDHjMRwEv4AvvUYojswpNxyrnSyjcFwWIR</a>	0.00051579 BTC	

*Change addresses* were introduced so people you are transacting with don't have access to your entire transaction history just by looking up the address you used for transacting with them. Most wallets automatically generate change addresses for you when you create a transaction. In the example above of a regular bitcoin transaction, you can see one input and two outputs to the transaction. The first output went to a different address and is the amount, the user wanted to spend. The remainder of the UTXO went back to the same address the funds originated from.

<a href="#">1LDHjMRwEv4AvvUYojswpNxyrnSyjcFwWIR</a>	0.015 BTC		<a href="#">1KVPHe4CpQBemEELUZ5d1VZdutKXGuXQTp</a>	0.01442771 BTC	
			<a href="#">12HdenPxQww7qg1LjhGc5Dy2zHv3V6Ydbg</a>	0.00051579 BTC	

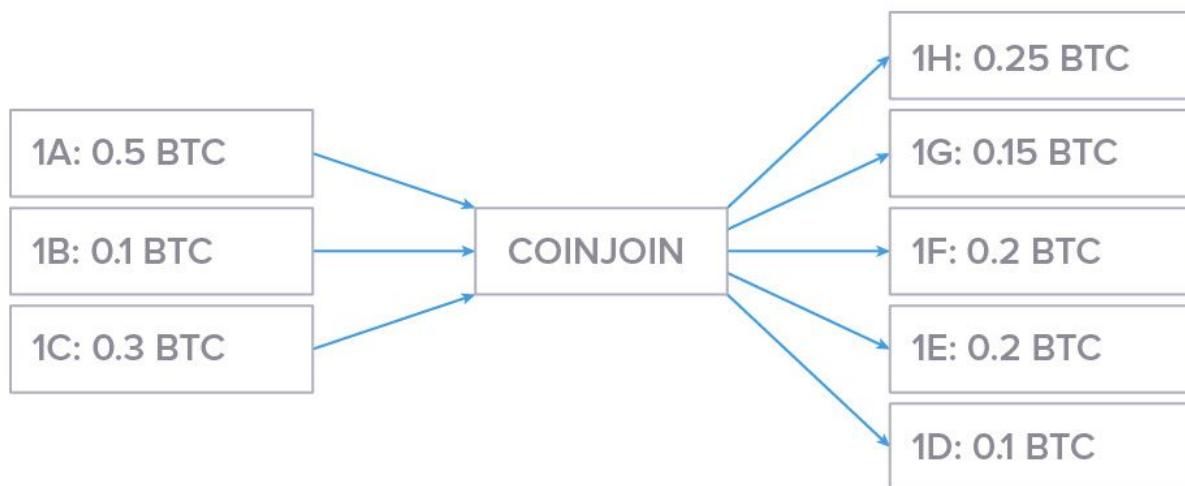
A wallet that supports change addresses will generate a new address, every time you are receiving funds, no matter if they are change or regular transactions. The example above shows a transaction with the exact same amounts as before, but this time the change went back to a newly created change address. This feature improves privacy by making it harder to trace the transaction history of a given user.

## Coin Mixing

Coin mixing protocols like SharedCoin, TumbleBit or CoinJoin (used by Dash) were the next evolutionary step to improve privacy. Several inputs are mixed by creating a single transaction from them, oftentimes during several intermediary transactions.

HORIZEN ACADEMY

### RANDOM SIZE COINJOIN TRANSACTION



Coin mixing doesn't require any changes to the basic blockchain protocol. In the graphic above you can see the schematics of a coin mixing transaction. A number of inputs are combined in a mixing pool (center) and later distributed to their destination addresses. A coin mixing transaction makes it harder for an attacker to figure out who was sending money to whom.

The level of privacy provided by mixing services is far better than using regular transactions, but one can link input addresses to output addresses by monitoring the amounts of coins in a mixing

transaction. There are tools available online to do this kind of chain analysis. Another downside of coin mixing is that many mixing services are centralized and run by a third party. CoinJoin based techniques solve this issue - they have no central party.

By now there are many iterations of coin mixing protocols that improved upon the privacy promise step by step. With CoinJoin for example, every user has to send the same amount to the mixing pool, which makes amount tracking significantly harder. Introducing [Confidential Transactions](#) will improve privacy even more by hiding the amounts transacted.

## Ring Signatures

Ring Signatures were first introduced by Rivest, Shamir, and Taumann in 2001 and the general idea has been used for a number of privacy protocols since then. We will use the White House Leak Dilemma to demonstrate the value proposition of their concept.

Imagine a high ranking White House official (Alice) wanting to leak a secret to the press about the president. She needs to make sure, the journalists who receive the leak have a way to verify the source of the information without revealing her identity. A potential solution would be using a Ring Signature scheme to sign the message. To construct the ring signature all she needs is her private key and the public keys of some other officials, e.g. other members of the cabinet (Bob and Carol).



The verifier (journalist) can verify that the message was indeed signed by a high ranking official, but he cannot determine who constructed the signature - Alice, Bob or Carol.

In the context of cryptocurrencies, a user can collect a bunch of public keys, create a transaction and sign it using his private key. The set of verifiers, being the nodes on the network can verify that the transaction is valid and that one of the group members has signed the message. They cannot tell who signed the transaction, which makes Ring Signatures great for private transactions. Monero is

the most notable cryptocurrency making use of Ring Signatures which are part of the CryptoNote protocol that Monero is built on.

## Zero-Knowledge Proofs

Zero-Knowledge Proofs (zk-Proofs) were known long before blockchain technology emerged but with distributed ledgers, a whole new set of possible use-cases came around. Simply speaking, a Zero-Knowledge Proof lets you prove to a *verifier* that you know something, without revealing that knowledge. A non-digital example of what this can look like can be constructed with a seeing person as the *prover*, a blindfolded person as the verifier, and two balls of a different color. Other than the color the balls are identical - same haptic, same weight, same size.

The seeing person (prover) wants to convince the blindfolded person (verifier) that the two balls are of different colors, without revealing the colors. They sit down at a table and the blind person shows the prover one of the balls. The blind person puts both balls under the table now and shows one ball in a second round - either the same one as before or the other one. If she chooses to show the same ball, the prover will know because of the color. If she were to switch balls under the table and show the other ball, the prover could tell with certainty.

## ZERO KNOWLEDGE PROOFS



In the second round, the prover would have a fifty-fifty chance of getting the right answer if he had to guess. He would have to guess in case the claim that he is trying to prove (the balls are of a different color) was false. At this point, the blind person cannot be sure if the claim is correct, or if the prover just got lucky.

If they repeat the game several times, the chance of getting the answer right every time through guessing decreases exponentially. After just ten rounds of the game, the probability of the prover calling the right ball every time through luck has decreased to 1 in 1024 ( $1 / 2^{10}$ ). The blind person

can be pretty sure by now, that the two balls are indeed of different colors although the prover has not shared any information about the colors themselves.

The idea of using Zero-Knowledge Proofs for cryptocurrency transactions is the following: You construct a proof that the transaction you want to send would be considered valid by a verifying node without revealing any of the actual transaction data. This allows the sender, receiver, and the amount to remain private. Another use-case that is perfect for the application of zk-Proofs is identity verification. E.g. you could prove to an entity that you are of a certain age without revealing any personal data like your DOB. Horizen uses zkSNARKs for its shielded transactions. zkSNARKs are a special type of Zero-Knowledge Proofs, namely *Zero-Knowledge Succinct Non-interactive ARguments of Knowledge*.

- *Succinct* refers to the proofs being short in the sense of easy to compute and verify.
- *Non-interactive* means that the prover and verifier don't have to be online at the same time. With the ball-example above, the prover and verifier have to go back and forth several times before the verifier actually has proof of the claim. With non-interactive proofs, the prover can construct the proof entirely on his own without the need for communication in the process. This proof can be written to the blockchain to be verified at any time.
- *Arguments of knowledge* describes the proof being computationally sound, i.e. no adversary can construct a false proof even if he has access to huge computational resources.

To use private transactions with Horizen, you will just use a different address type. In your wallet, you can either generate t-Addresses (transparent Addresses) or z-Addresses (shielded Addresses). When you send funds to a z-Address, the amount and sender are recorded on the blockchain, but not the receiving address. If you transact between two z-Addresses, no information about the transaction gets publicly recorded - neither the sender, receiver nor the amount. If you want to try this feature, you can download our flagship app [Sphere by Horizen](#). Make sure to activate full mode in the settings otherwise, you won't be able to generate z-Addresses.

## Summary

There are many ways to reclaim your privacy on a public blockchain. The approaches like Change Addresses and Coin Mixers don't provide strong privacy, but they help make it harder to trace transactions to their origin and link real-world identities to addresses on the blockchain. Ring Signatures and Zero-Knowledge Proofs are more advanced concepts, that actually allow you to

transact entirely private, even on fully open and public blockchains. Horizen offers what we call selective privacy - depending on what address type you use, you can decide if you want to transact transparently or privately.

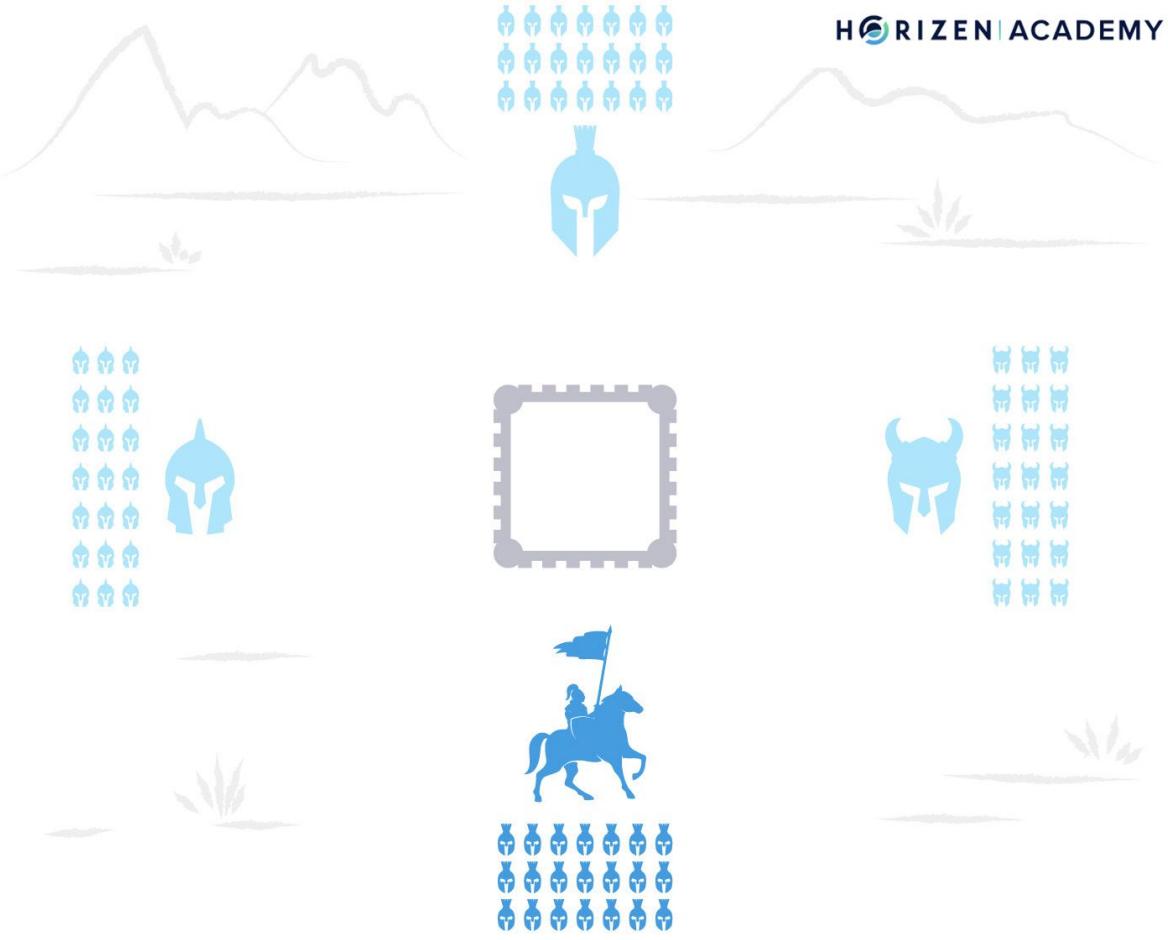
# Attacks

Blockchains are generally considered to be very secure, but the level of security they provide is proportional to the amount of hash power that supports the network. The more miners there are and the more powerful their mining hardware is, the harder it is to perform an attack on the network. In this article, we want to cover the most common attack scenarios on public blockchains.

## The Byzantine Generals Problem

Before we get into the different attack scenarios we would like to introduce you to a sort of thought experiment, namely the Byzantine General's Problem that remained unsolved for centuries until blockchain technology was introduced.

Imagine you are a general a few centuries ago and you want to attack a castle with your army. The castle is very robust and the army inside strong. You have arranged a number of other armies to support the attack and each of those armies is going to attack from a different side. The armies are separated by distance, each of them several miles apart. If they all attack at the same time the chances of victory are very high. If the attack is uncoordinated then they will most likely suffer defeat.

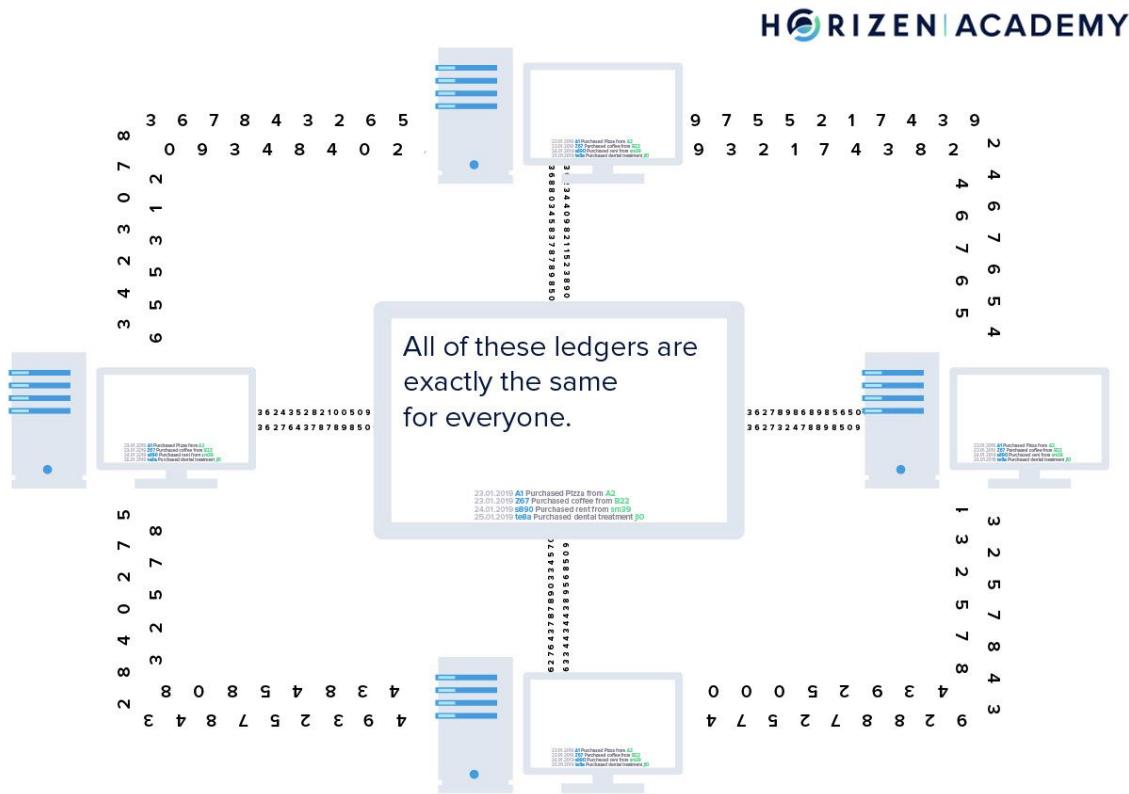


You as the general have the following problem: How can you make sure all armies are attacking at the same time. In other terms how can you achieve consensus on the time of the attack? You cannot give signals with flags, torches or smoke, as those signals could be picked up by the enemy.

You could send messengers on horseback, but what if one of them is captured or killed before he reaches the intended General? To know the other generals received the message, you could have them send a messenger back with a confirmation. The messengers carrying the confirmation could be captured or killed as well on their way back to you.

The other generals wouldn't know if you received their confirmation, so you would have to send out confirmations of the confirmations but what if those messengers get captured? Even without the risk of imposters transferring fraudulent messages and traitors confirming to attack with the intent of not doing it this situation was thought to be impossible to solve. Nobody could know with absolute certainty if the other generals intended to attack at the same time or not.

Blockchain technology solved this dilemma - although this would not really have helped the Byzantine army.



Each General now has a ledger of events that is synchronized with the other General's ledgers. No central party is in charge of the coordination. Every time a block gets mined, all the participants agree on the order of events for the last couple of minutes. Getting back to our general's problem, now they have a way of knowing if all of them are going to attack, or if they should retreat collectively.

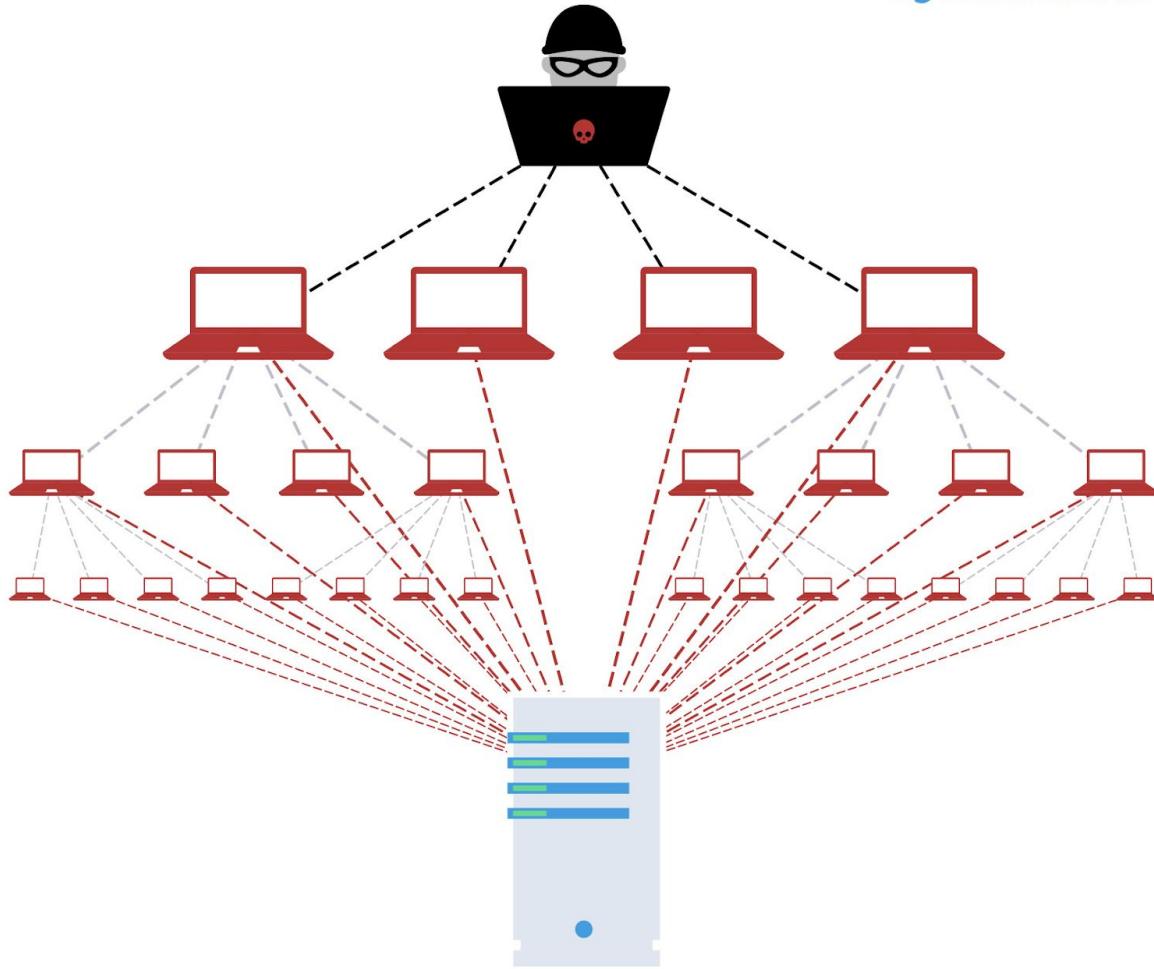
Now that we have talked about the general problem a consensus mechanism aims to solve, let's look at some simple and intuitive attack scenarios and how we address them.

## DDOS Attack

A Distributed Denial-of-Service (DDOS) attack in computing is an attack, where a perpetrator seeks to make a network resource unavailable to its users, by flooding the network with a large number of requests in an attempt to overload the system. It is an attack that not only blockchains but any online

service can suffer from. In a simple form, the DOS (Denial-of-Service) attack, all these requests originate from the same source. This makes it somewhat easy to prevent. If a single IP-address sends a huge amount of requests that cannot be justified by legitimate reasons, you can have a measure in place that automatically blocks this IP-address. In the case of a DDOS attack, the *distributed* part refers to a large number of different sources that the malicious requests originate from.

HORIZEN ACADEMY



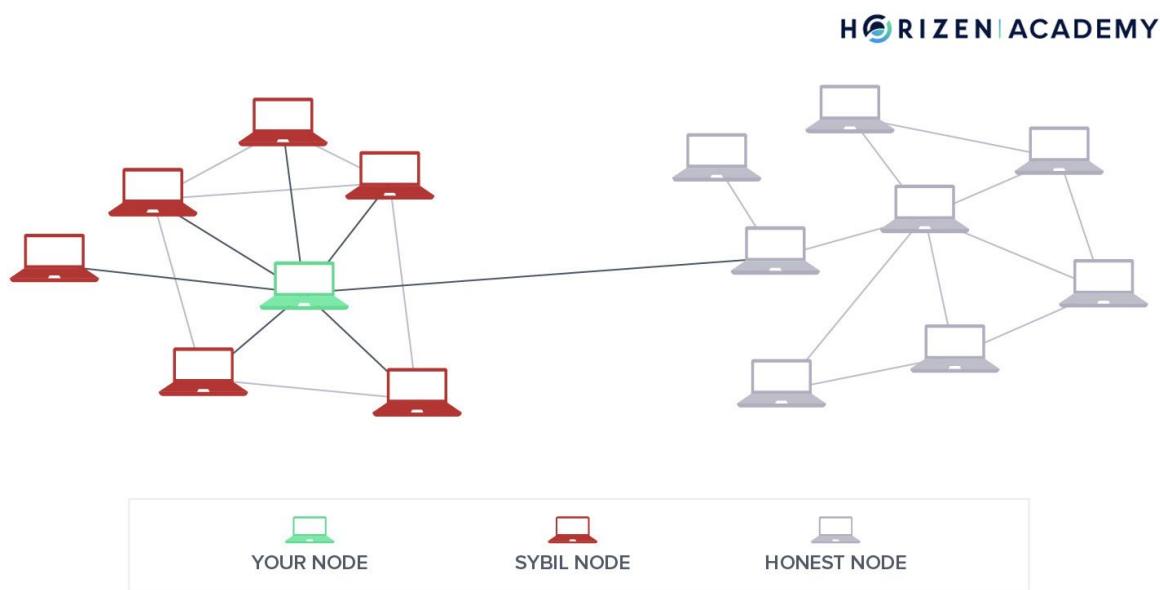
A DDOS attack is much harder to tackle because to do so you need to differentiate between legitimate and malicious requests. This is a very hard problem. In the context of blockchains, this comes down to an almost ideological question. The motivation to introduce transaction fees was to eliminate spam. Some people argue that as long as the requests have a transaction fee attached they cannot be considered spam by definition. While there are certainly situations where you could

consider transactions to be spammy, it would be a slippery slope to start blocking them. One of the greatest value propositions of public blockchains is their censorship resistance. Starting to pick transactions that are not included - no matter of what criteria this censorship is based on - would be a dangerous precedent for any blockchain.

## Sybil Attack

A Sybil Attack is an attempt to manipulate a P2P network by creating multiple fake identities. To the observer, these different identities look like regular users, but behind the scenes, a single entity controls all these fake entities at once. This type of attack is important to consider especially when you think about online voting. Another area where we are seeing Sybil attacks is in social networks where fake accounts can influence the public discussion.

Another possible use for Sybil attacks is to censor certain participants. A number of Sybil nodes can surround your node and prevent it from connecting to other, honest nodes on the network. This way one could try to prevent you from either sending or receiving information to the network. This "use case" of a Sybil attack is also called Eclipse Attack.



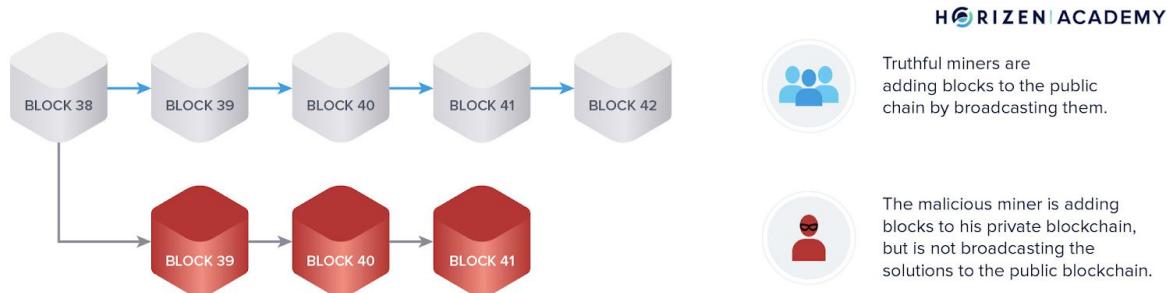
One way to mitigate Sybil Attacks is to introduce or raise the cost to create an identity. This cost must be carefully balanced. It has to be low enough so that new participants aren't restricted from joining the network and creating legitimate identities. It must also be high enough that creating a

large number of identities in a short period of time becomes very expensive. In PoW blockchains, the nodes that actually make decisions on transactions are the mining nodes. There is a real-world cost, namely buying the mining hardware and consuming electricity, associated with creating a fake "mining-identity". Additionally, having a large number of mining nodes still doesn't suffice to influence the network meaningfully. To do that you would also need large amounts of computational power. The associated costs make it hard to Sybil attack Proof-of-Work blockchains.

## 51% Attack

The best-known type of attack on public PoW blockchains is the 51% attack. The goal of a 51% attack is to perform a *double spend*, which means spending the same UTXO twice. To perform a 51% attack on a blockchain, you need to control a majority of the hash rate, hence the name. A malicious miner wanting to perform a double spend will first create a regular transaction spending their coins for either a good or for a different currency on an exchange. At the same time, they will begin mining a private chain. This means they will follow the usual mining protocol, but with two exceptions.

- First, they will not include their own transaction spending their coins in their privately mined chain.
- Second, they will not broadcast the blocks they find to the network, therefore we call it the private chain.



If they control a majority of the computing power, their chain will grow faster than the honest chain. The Longest Chain Rule in PoW blockchains governs what happens in case of such a fork. The branch, that has more blocks to it and accordingly represents the chain created with a larger amount of computing power is considered the valid chain.

Once the attacker has received the good or other currency bought with their coins, they will broadcast the private branch to the entire network. All honest miners will drop the honest branch and start mining on top of the malicious chain. The network treats the attacker's transaction as if it never happened because the attacker did not include it in his malicious chain. The attacker is still in control of their funds and can now spend them again.

This has happened to many smaller blockchains in the past. In fact, Horizen suffered from a 51% attack in early June 2018. We immediately started to work on a solution to mitigate the risk of a 51% attack on smaller blockchains that are not secured by as much computing power as for example the Bitcoin blockchain.

[We came up with a solution](#) that penalizes delayed block submissions. There is no legitimate reason for a miner, to broadcast several blocks to the network at once. Our protection mechanism makes these attacks very costly. So costly that it does not make any economic sense to perform such an attack on our network. Many other blockchains are now looking to implement a similar protection mechanism with their protocol.

## Summary

Blockchains have solved the Byzantine General's Problem of achieving consensus on the order of events in an untrustworthy environment. There are different ways a blockchain can be attacked. Performing these attacks becomes more difficult over time as more computing power is added to the network and it becomes more robust.

In a DDOS Attack, a perpetrator wants to slow down or halt the network by spamming it with a large number of transactions. In a Sybil Attack, a malicious actor controls many fake identities and tries to either meddle with online elections or to manipulate the communication in a P2P network.

In a 51% attack, a miner controlling the majority of computing power on the network tries to spend coins twice, by writing a private version of the blockchain first, before broadcasting all blocks at once to the honest miners.

The attack scenarios presented in this article, except for the 51% attack, are not endemic to blockchains and have been around since the beginning of distributed peer networks. There are many measures in place to mitigate the risk of the different attack scenarios out there. We will look at them closely in the Expert Level. We hope this last article didn't leave you with a wrong

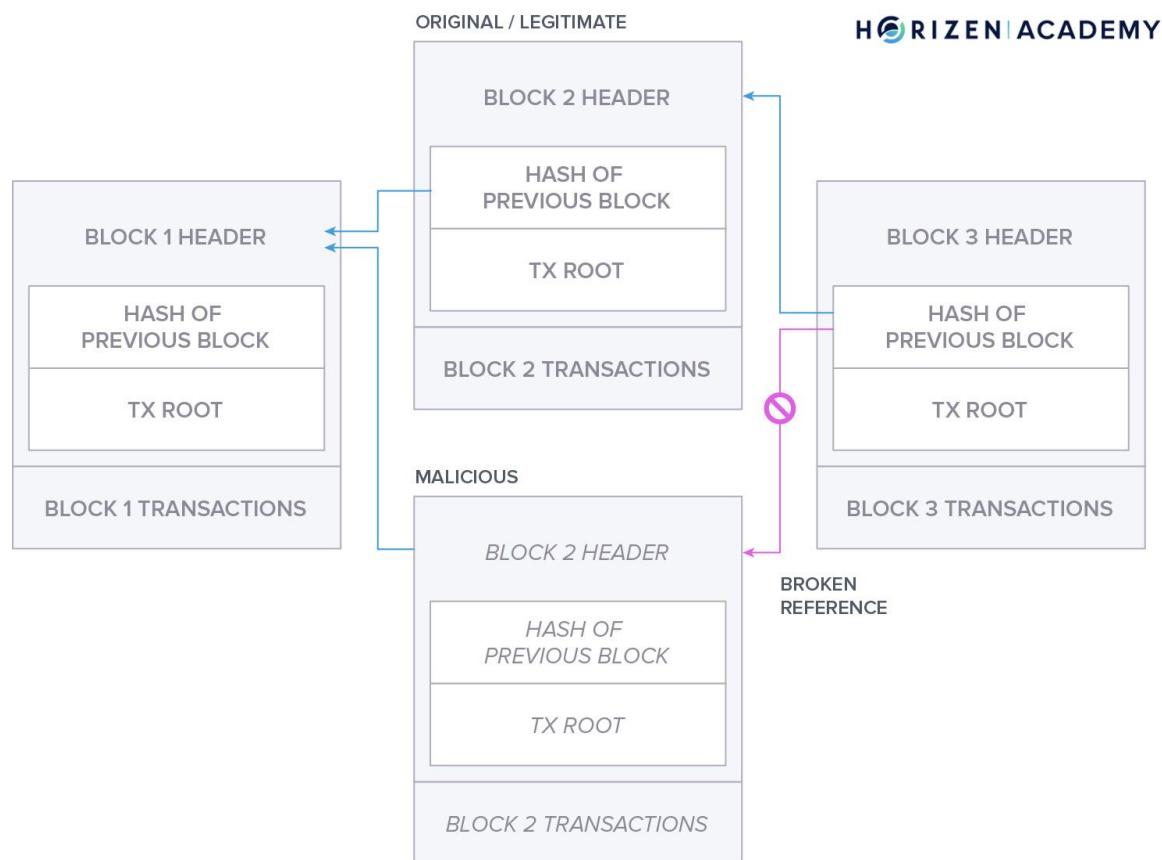
impression on blockchain security. Blockchain technology is highly secure, but as with anything else in the digital realm, there are no invincible protocols.

# Summary Advanced Level

In this last article of our advanced Technology Section, we want to summarize what topics and concepts we covered throughout this level.

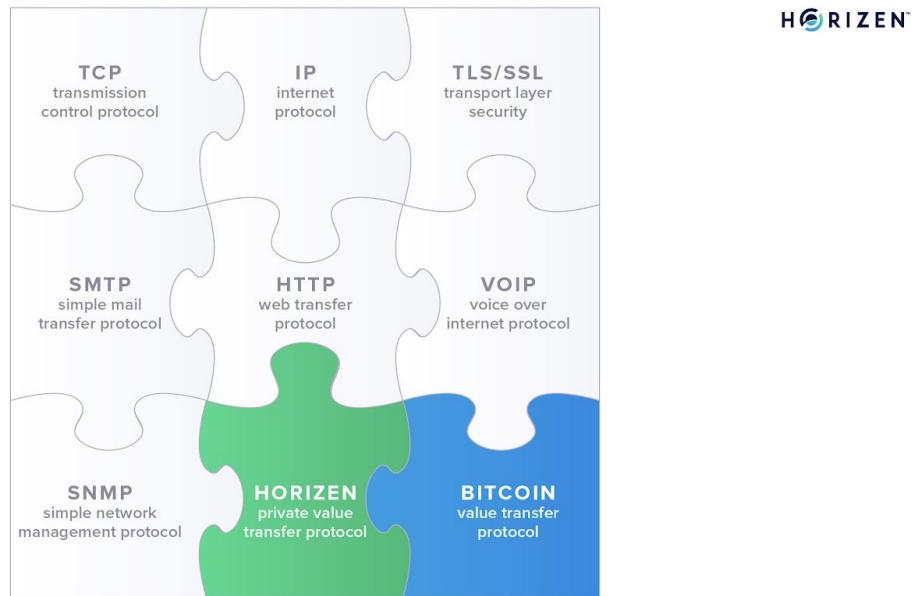
## What is a Blockchain?

In our first chapter - "What is a Blockchain" - we looked at it from three different perspectives. First, we looked at it as a computer scientist and described the blockchain as a data structure. A blockchain is like a linked list, a common data structure with the difference that the references that link the individual blocks are cryptographically secured. This makes it infeasible to tamper with the data recorded on a blockchain.

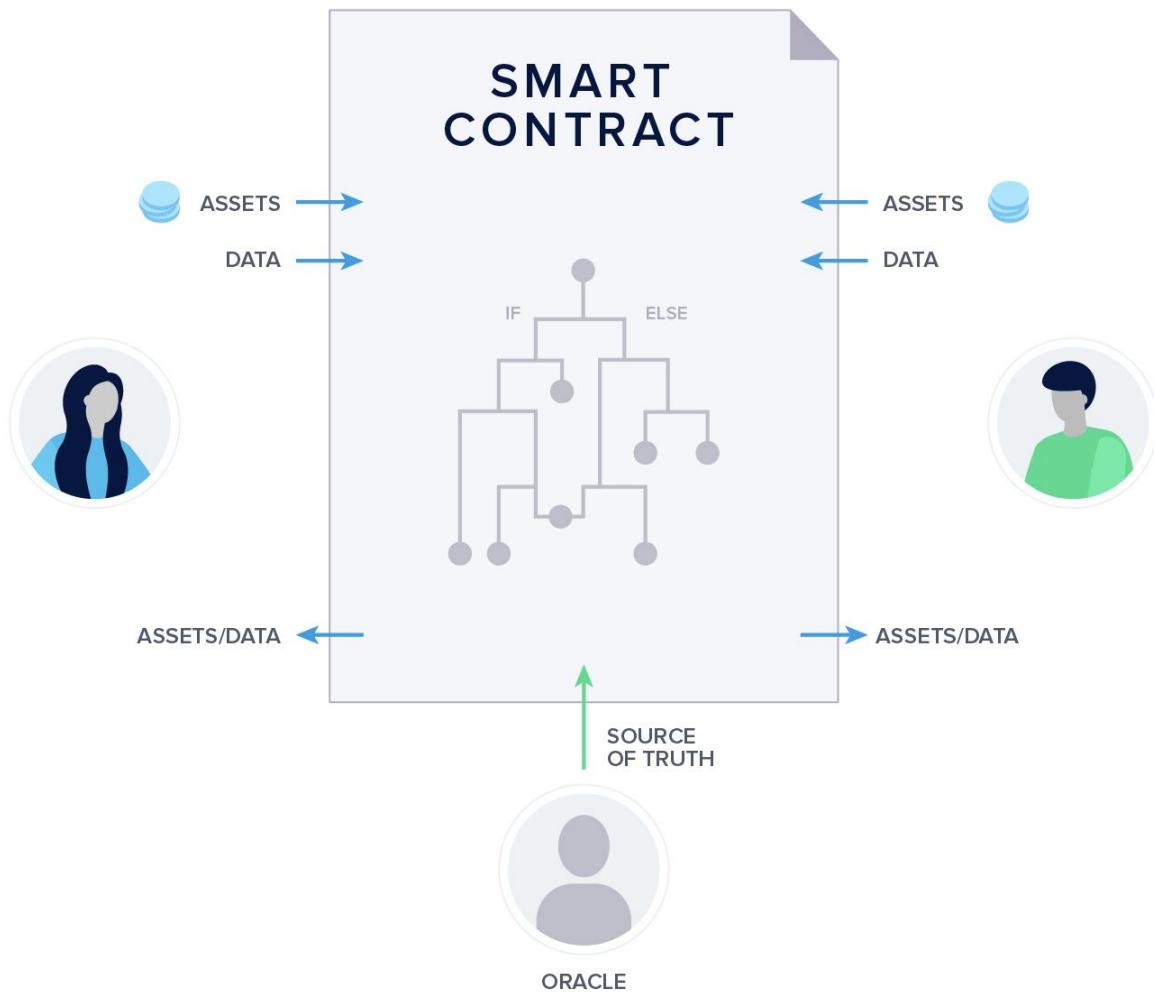


Second, we looked at the blockchain as a protocol to transfer value. There are many different network protocols that allow devices to communicate in standardized ways. Without those

standards, it would be hard for developers and engineers to build interoperable software and hardware. Blockchains are a new class of protocols that standardizes the way people can exchange value over the internet without any intermediaries such as banks or other payment providers like PayPal.



Lastly, we introduced you to Smart Contracts. Smart Contracts are software stored and running on the blockchain. They ensure objective execution on the basis of mutually agreed-upon terms enforced by code. They have the potential to reduce middleman which in turn reduces cost and saves time. Before we can see widespread adoption we have to overcome some hurdles, like solving the Oracle Problem that describes the challenge of reliably submitting real-world data to the blockchain.

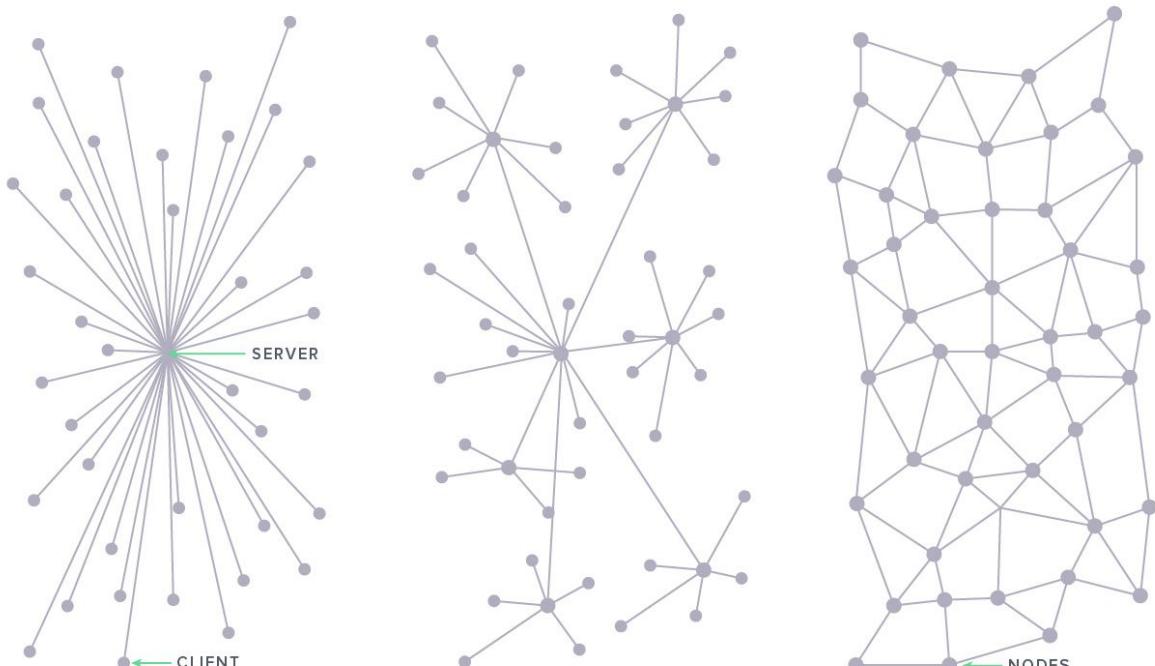


## How Does a Blockchain Work?

In the second chapter, we provided an overview of the different elements of a blockchain first, before we looked at each element more closely. The first two concepts we covered were hash functions and public-key cryptography. While hash functions are used to verify the integrity of data public-key cryptography is used to verify ownership.



Next, we talked about distributed Peer-to-Peer networks. There are thousands of nodes that maintain a copy of a blockchain, in the case of our node network more than 25,000. Together with the cryptographically secured data structure, this is the reason blockchains are so robust. If one of your peers goes offline you just connect to another one. If you run a node and happen to go offline for a while, you can reconnect at any time to get updated by your peers on the blocks that you missed and become fully functional again shortly.



The consensus mechanism of a blockchain allows the network to agree on a single transaction history. The two most commonly used consensus mechanisms are Proof of Work (PoW) and Proof of Stake (PoS). Both have in common that the voting power of an entity is tied to a limited resource. For PoW blockchains this limited resource is computational power, for PoS blockchains it is ownership of the native currency. While PoW has demonstrated robustness over an extended period of time with Bitcoin, PoS is yet to prove this robustness.

## PROOF OF WORK

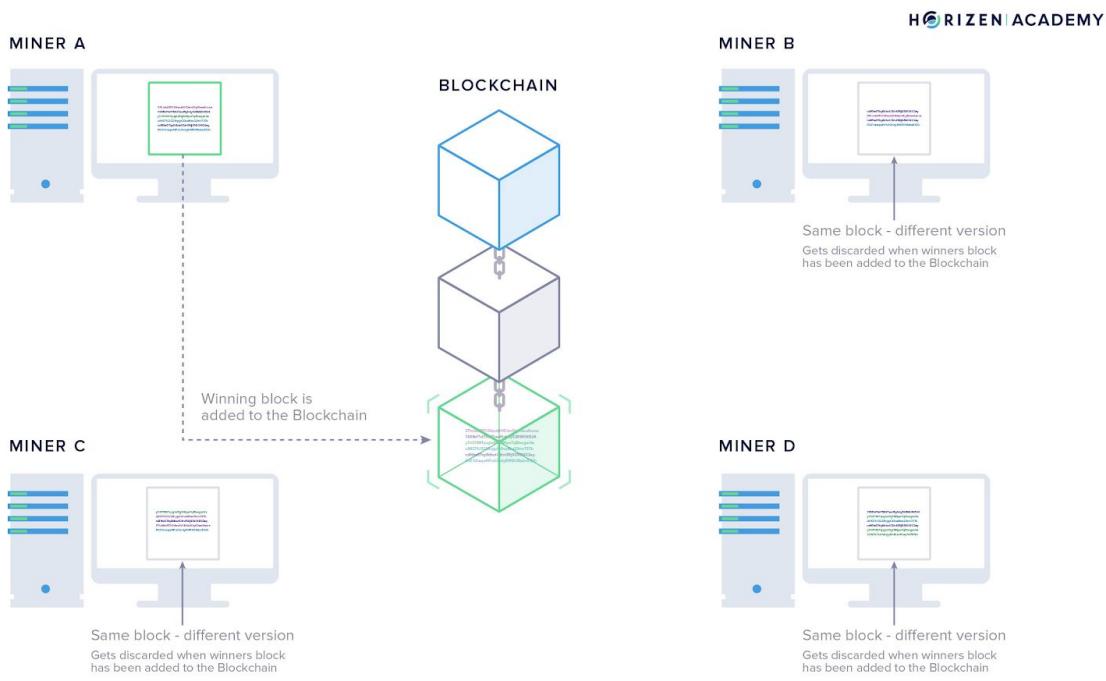
-  Distributed consensus among untrusted and unknown nodes
-  Incentives are rewarded within the system for work done outside the system
-  Relatively high cost of entry, but high returns
-  Slow transaction rate
-  Low efficiency, consumes more energy
-  Empirically proven

## PROOF OF STAKE

-  Distributed consensus among untrusted and unknown nodes
-  Incentives are rewarded within the system for escrow inside the system
-  Low cost of entry, but low returns
-  Fast transaction rate
-  High efficiency, consumes less energy
-  Experimental

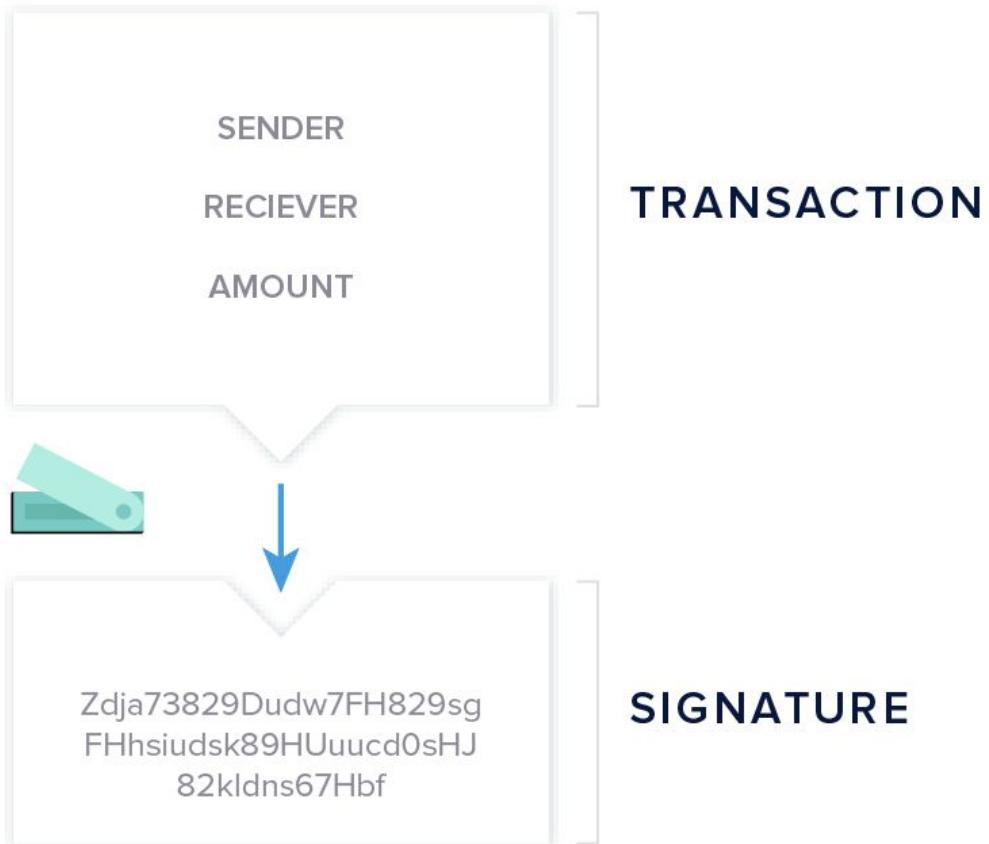
For the last article in the chapter on how blockchains work, we talked about mining. Miners work to secure the blockchain against attacks and protect the history recorded up to that point against changes. Adding a block to the blockchain requires a large number of random guesses to find a nonce that will produce a valid block hash. The block hash must satisfy the current difficulty requirement on the network. This makes it extremely difficult to tamper with data on the blockchain. If somebody wanted to change the record, he would have to redo the work by himself. Not only would he need to do the work, but also at a faster pace than all honest miners combined.

The miners are in a competition and their chance of finding the next valid block is proportional to the amount of computational power they control. Miners receive newly created coins as a reward for their efforts. It is one of the great innovations introduced with Bitcoin that a distributed network can pay its participants for its own maintenance.



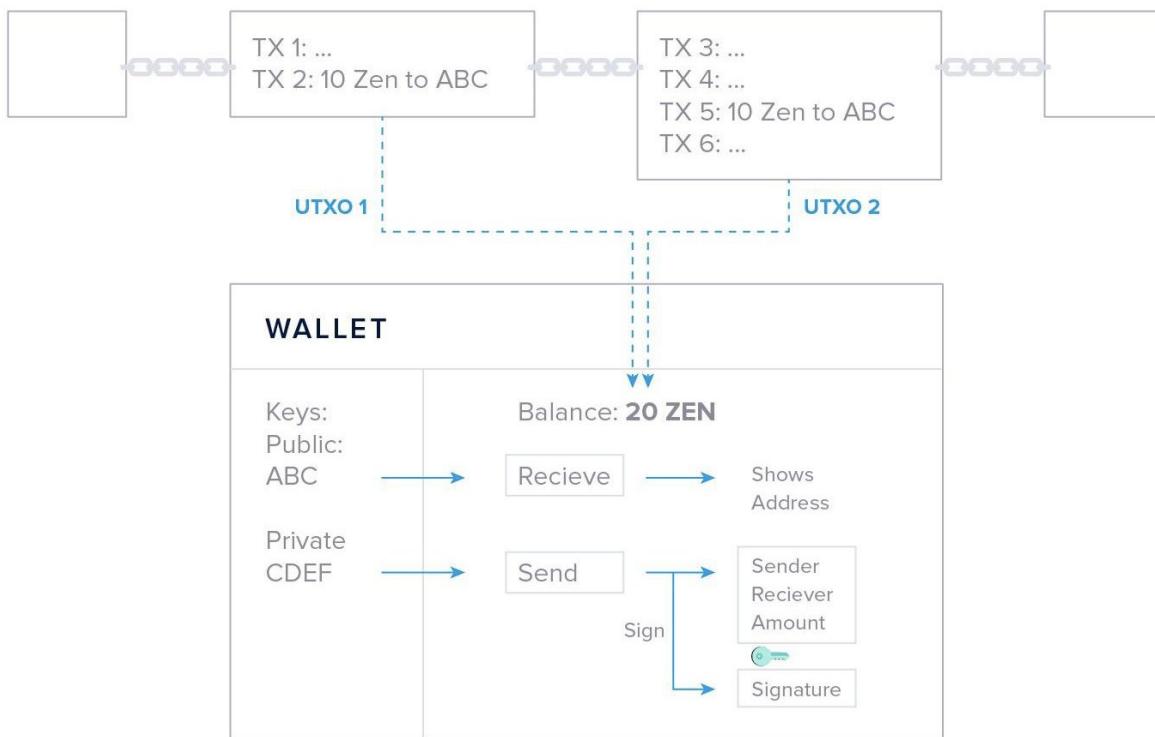
## Wallets

We dedicated the third chapter to cryptocurrency wallets and the different types there are. A wallet is an app to generate, manage, and store a pair of cryptographic keys for you. You can check your balance, receive, and send funds with a wallet. Usually, there is a trade-off between convenience and security. Funds on a mobile wallet are convenient to spend but not very secure, just like cash in your pocket. Large amounts of money stored on a hardware wallet are less convenient to spend but very secure. The most important question when considering the safety of a wallet is where the keys are stored? If you don't control your keys, you don't control your funds. You authorize the spending of your funds with your private key in a step called signing a transaction and you have to keep your private key safe at all times.



## Transactions

In the fourth chapter, we talked about transactions. With the first article, we introduced you to the UTXO (Unspent Transaction Output) Model. It is the accounting method that is used in most blockchains. The blockchain does not maintain a balance for every address. Instead, your wallet goes through the transaction history on the blockchain and takes all the incoming transactions that you haven't spent yet - your UTXOs - and adds them together in order to generate your available balance.



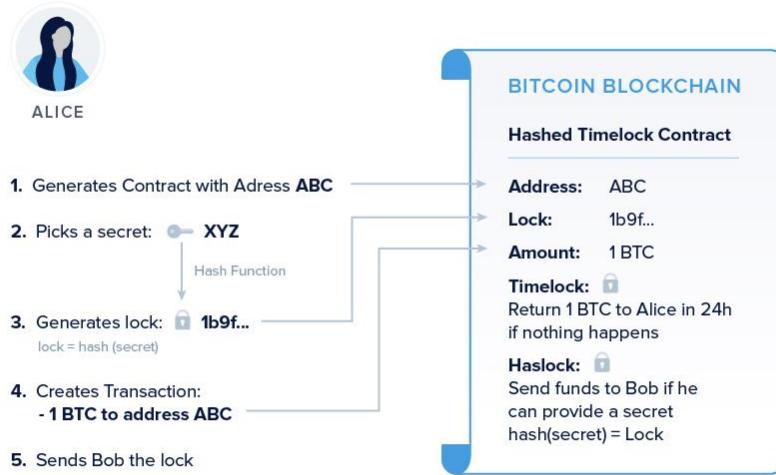
In the next article on transactions, we talked about the block explorer, a graphical tool to view and explore data on the blockchain. There is a block explorer available for almost every public blockchain out there. It allows you to browse the history of a given chain with all the transactions that ever happened, sorted by addresses or blocks.

## Transactions

b1fea52488ce0c62bb442b530a3f0132b826c74e473d1f2c220bfa78111c5082		2009-01-12 03:30:25
No Inputs (Newly Generated Coins)	→ 1PSSGeFHDnKNxIeyFrD1wcEaHr9hrQDDWc	50 BTC
50 BTC		
14184fc596403b9d638783cf57adfe4c75c605f6356fb91338530e9831e9e16		2009-01-12 03:30:25
12cbQLTFMXRnSzktFkuoG3eHoMeFtpTu3S	→ 1Q2TWHE3GMdB6BZKafqwxXtWAWgFt5Jvm3 12cbQLTFMXRnSzktFkuoG3eHoMeFtpTu3S	10 BTC 40 BTC
50 BTC		

We also showed you a very special kind of transaction in this chapter, an Atomic Swap that allows users the trustless exchange of two different cryptocurrencies between two separate blockchains. At its heart, Atomic Swaps rely on Hashed Time-Locked Contracts or HTLCs. Atomic Swaps present an alternative to centralized exchanges used today. At no point is there a third party involved that has

access to a user's funds. The exchange process is entirely trustless and depending on the users can be almost instant.



## Privacy on the Blockchain

The human right to privacy is one of Horizen's core values. We looked at different methods to preserve your privacy on the blockchain. We started with simple methods such as using different addresses for every new transaction and coin mixing protocols that combine a number of transactions from different users to obfuscate the link between sender and recipient. A more advanced technique to achieve private transactions is the use of Ring Signatures. With Ring Signatures, a group of people signs a transaction and the verifier will know for sure that one of the group members sent the transaction but he won't be able to tell which one.

The last privacy-preserving technology we talked about were Zero-Knowledge Proofs. It is the technology Horizen uses for its shielded transactions. Simply speaking, a Zero-Knowledge Proof enables you to prove to a verifier that you know something, without revealing that knowledge. We used the example of a seeing person convincing a blindfolded one that two balls are of a different color, without revealing the colors.

## ZERO KNOWLEDGE PROOFS



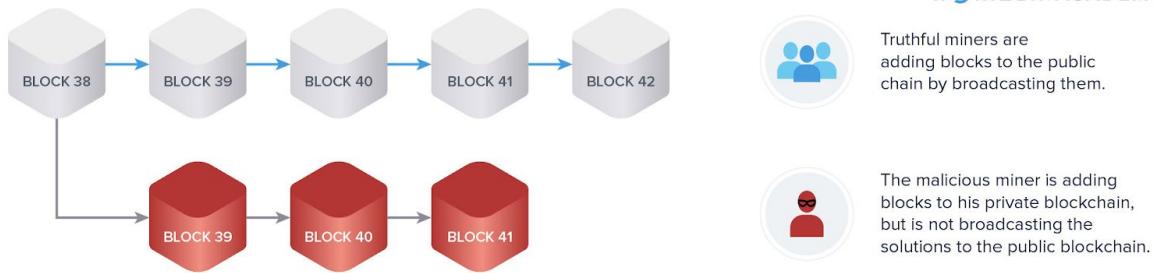
### Attacks on Blockchain

The last article in our advanced level covered different attacks on blockchains. We introduced you to the Byzantine Generals Problem, a thought experiment where a number of generals want to coordinate an attack without a reliable means of communication. Blockchains pose a solution to the general's problem because they allow a group of untrusted individuals to achieve consensus - even when communication can be faulty.

Next, we talked about DDOS attacks, where an attacker tries to slow down or even halt a service by sending a large number of fraudulent requests to the network.

In a Sybil Attack, an attacker creates a large number of fake identities. These fake network participants can be used to manipulate the communication between peers.

Lastly, we talked about the most common type of attack, the 51% or block reordering attack. A miner controlling a significant share of the total hash power of a network can attempt to perform such an attack. The malicious miner creates a transaction on the honest chain spending his funds. In the meantime, he mines blocks privately, meaning the miner doesn't broadcast the blocks to the network. Once the chain, which does not include the transaction spending the malicious miner's funds becomes the longest chain he broadcasts it and according to the longest chain rule it will be recognized by all miners as the new valid chain. The attacker managed to gain control of the funds once again and can now spend them a second time.



Truthful miners are adding blocks to the public chain by broadcasting them.

The malicious miner is adding blocks to his private blockchain, but is not broadcasting the solutions to the public blockchain.

# Final Remarks

We covered many concepts in the Advanced Level and we hope you have learned a lot. If you like what you have found here, please go ahead and share your knowledge as well as our website with your friends and family. If you had trouble following along at some point, don't worry! It is a complex topic and took all of us a while to wrap our heads around. You can always come back to re-read these articles. It will make a lot more sense reading it a second time with a little break in between.

If you feel comfortable about everything you have read and would like to keep learning, there is more! Move up one level and check out our Expert Content. We have structured it the same way, but added more detail to the topics and split some of them up to look at the individual components more closely. We designed the content in a way where you can either read it from top to bottom (which we can only recommend) or jump to articles that you are especially interested in.

We hope you enjoyed this series of articles. Please let us know if there is anything that you find confusing. The content provided is and will be work in progress. We are always open to suggestions and constructive feedback so drop us a message at [academy@horizen.global](mailto:academy@horizen.global) if you want to share your thoughts with us.

**Your Horizen Team**



[academy.horizen.global](http://academy.horizen.global)