Bachelor thesis
Computer Science

Radboud University

Comparing Bitcoin and Ethereum

Author:
Lotte Fekkes
s4496426

First supervisor/assessor:
Dr. Lejla Batina
lejla@cs.ru.nl

Second supervisor:
ir. Louiza Papachristodoulou
L.Papachristodoulou@science.ru.nl

Second assessor:
Dr. Joeri de Ruiter
joeri@cs.ru.nl

January 16, 2018

Abstract

Cryptocurrencies are very popular right now and are often mentioned on the news. But how these online currencies really work, is often not known. This thesis will explain and compare the current biggest cryptocurrencies Bitcoin and Ethereum. Bitcoin has the highest value of all cryptocurrencies and has been widely accepted as a means of payment. Ethereum is growing fast and is used for many different applications besides an online currency. The thesis first covers some basic cryptographic principles needed to understand certain parts of a cryptocurrency. Literature research is done to explain how the blockchain is applied in Bitcoin and Ethereum, how the different transactions are made and how mining is done. Each cryptocurrency is explained in detail in their own chapter. At the end a comparison is made and discussed in different categories. The results of the comparison are also put in a table for a quick overview. In this comparison it becomes clear that Ethereum is faster and more versatile than Bitcoin.

Contents

Chapter 1

Introduction

Bitcoin and Ethereum are currently the biggest and best known cryptocurrencies in the world. Almost every week, one of these currencies is mentioned in the news and a lot of people are curious about them.
Bitcoin was introduced in 2008 by Satoshi Nakamoto in the paper "Bitcoin: A Peer-to-Peer Electronic Cash System" [31]. Satoshi Nakamoto is a pseudonym and nobody really knows who is or are the inventor(s) of Bitcoin. The paper introduces the use of a blockchain and explains the main concepts of how the system would work as a safe electronic transaction system.
The Ethereum whitepaper [20] was written by Vitalik Buterin in which he introduced Ethereum, a new cryptocurrency based on the blockchain technology from Bitcoin. Ethereum can be used for far more applications than only an online currency. In 2014 Vitalik Buterin founded Ethereum together with Gavin Wood and Jeffrey Wilcke.
Although the currencies are known by name, it is not widely known how these cryptocurrencies actually work. Therefore, with this study we give an insight in Bitcoin and Ethereum by explaining and comparing the two cryptocurrencies. More precisely, we explored the following aspects:

- What algorithms do these cryptocurrencies use?
- What other necessities are needed to make a online currency work? (And how do they work?)
- How do we compare two different cryptocurrencies?

Once we have the answers to these questions, we are able to explain and compare Ethereum and Bitcoin.

Organization. In this thesis we will first cover some information that is needed to understand the technical details of the cryptocurrencies. This is done in the preliminaries chapter. The next two chapters describe Bitcoin and Ethereum, each in their own chapter, in detail. Chapter six contains the

comparison between Bitcoin and Ethereum and has a table which summarizes the comparison in a quick overview. Finally, we talk about the related work to this thesis.

Chapter 2

Preliminaries

This chapter contains the basic cryptographic principles that are used in all cryptocurrencies. At first, hash functions will be discussed, after which some different tree data structures are discussed. Then a basic explanation of the blockchain is given and finally Proof-of-Work and Proof-of-Stake are discussed.

2.1 Hash functions

Hash functions are used in a lot of different cryptography applications. Hash functions map a message to a smaller set of strings of a fixed length: the hashes. The function is easy and fast to compute, but very hard to invert. These hashes are useful in cryptography because of the following properties [37]:

Collision resistance: It is difficult to find two messages that have the same hash. That is, find two messages a and b, where $a \neq b$, such that, $H(a) = H(b)$. Where H is the hash function that is used.

Preimage resistance: Knowing the output of a hash function h and the hash function itself H. It is difficult to find a message x that hashes to that output. This means that given h and H it is hard to find x, such that $H(x) = h$.

Second preimage resistance: Knowing one message x and its output from hash function H, it is difficult to find another message x', that has the same hash as output from H. Thus, given x and H, finding a message $x' \neq x$, such that $H(x) = H(x')$, is hard.

These properties are necessary for the cryptographic algorithms to work properly and to be secure. If it is easy to find multiple messages that have the same hash, an attacker could make use of this. The attacker could let someone sign a document and later say that person has signed a different

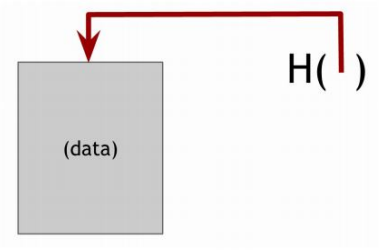

Figure 2.1: Hashpointer as displayed in [32]

.

document, which has the same hash as the first document. The fastest way to find a hash collision is brute force. In Proof-of-Work they make use of this property and if there was a faster way, people could cheat. Hashes can also provide anonymity by concealing the real identity of someone. People only see a hash and that hash does not say anything about a person. Hashes also provide integrity for the message. If the message is changed, the hash will be different than the hash from the original message. Thus, with a hash someone can check if the message is still the same as it was when first published. This is used in hash pointers.
Hash pointers contain a pointer to the position of the data and a hash of this data [32]. See also Figure 2.1. If the data included in the hash pointer are altered, the hash output of the data will be different than the one stored previously in the hash pointer. In this way, anyone can check the validity and the correctness of the data.

2.1.1 Merkle Tree

A binary tree with hash pointers is called a Merkle tree [32]. The leafs in the tree contain the data. The parent node, one level up in the tree, contains a hash of this data and is paired with another parent node. This continues all the way up to the root. In this way, the root node is the hash of all the data in the tree. (See also Figure 2.2.) The root hash can be used in public and it is relatively easy to verify that a certain data block is included in the Merkle tree. This is done by looking only at the block of data in the leaf and the path to the root. One can verify each of the hashes of the tree all the way up to the root. If all the hashes are correct, then the block of data is included in the tree. If there are n nodes in the tree, it takes about $\log(n)$ time to verify a block of data.

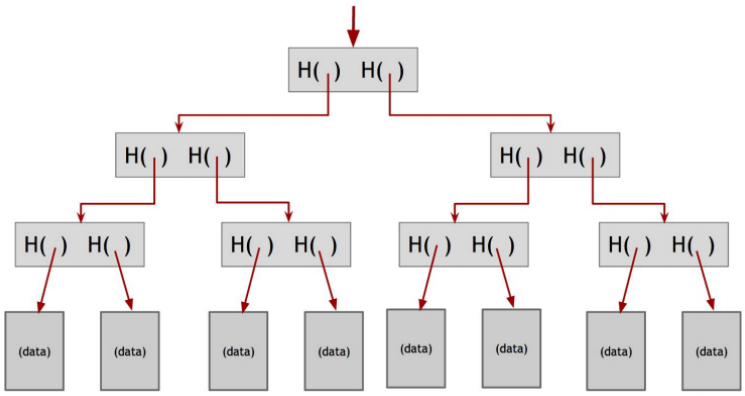

Figure 2.2: Merkle Tree as displayed in [32]

2.1.2 Patricia Tree

A Patricia tree is a radix tree with a radix of 2 [11]. The data structure is mostly used to store strings or key-value pairs. This tree data structure optimizes space compared to a regular radix tree. Less space is needed in a Patricia tree, because any node that is an only child is merged with its parent. This way a node has at least two children or none. The stored string or key is the path taken from the root of the tree to a leaf. See Figure 2.3. In the leaf, the value of the key-value pair can be stored. The keys are compared bit by bit. Each charachter or bit tells which way to follow in the tree [42]. And thus the key under which the value is stored, is encoded in the 'path' [8].

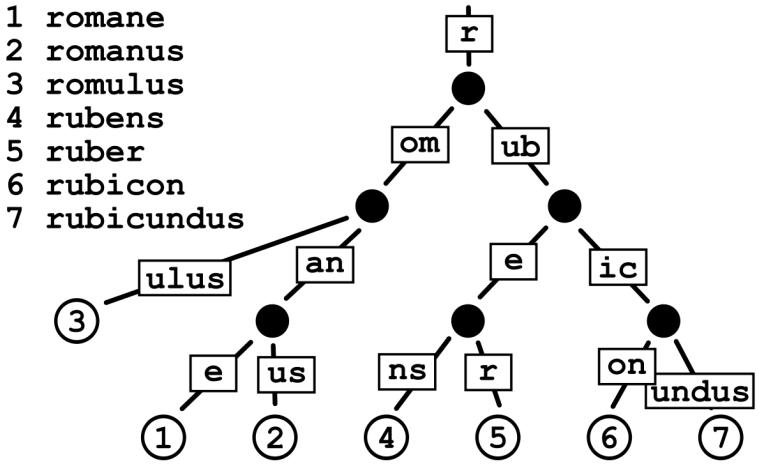

Figure 2.3: PatriciaTrie as displayed in [11]

2.2 Blockchain

The blockchain is a relatively new and innovating technology. In order to understand how different cryptocurrencies work, we must first understand how the blockchain works in general. The blockchain is a structure that saves all records, transactions or events that are shared among the users of the blockchain network [25][2]. These records or transactions are put in blocks. Blocks are a collection of these transactions and have some other fields, specific to the system that uses a blockchain. The specific structure of blocks will be discussed in later chapters. The blocks are linked together by hash pointers that point to the previous block. See also Figure 2.4.
When a user publishes a transaction, all users verify this transaction by consensus and allow it in a block. Because of the consensus, it makes the whole system decentralized. There is not a single party who controls all the transactions. In this way, no data will be lost, because all data is saved by all users in the blockchain and not only by a single party. Whenever a transaction is added to the chain, it can never be erased or changed. And thus this is a trustworthy system for all users. With the blockchain, there is no need for a 'trusted' third party, and it all works in a peer-to-peer network.

It is possible for blocks to not be accepted in the main chain. There could be a fork in the chain and the different nodes will continue working on the longest valid chain, leaving those other blocks abandoned. The reasons for having forks in the chain are various:

- Blocks might have been mined and added in the chain at the same time.
- Some blocks might be invalid or wrong and the users realized that after a few more blocks were added.

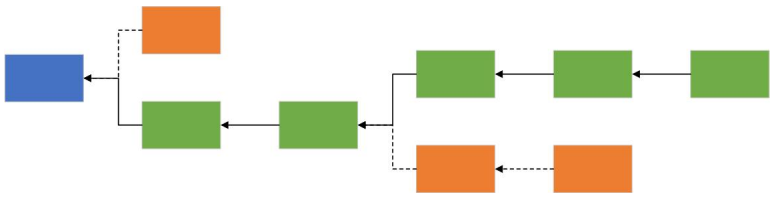

Figure 2.4: Blockchain with the blue block being the first block, the green blocks the main chain and the orange blocks the orphaned blocks. Each block points to the previous block with a hashpointer.

2.3 Proof-of-Work

A proof of work is evidence that you did some work. This is mostly done with difficult (hash)puzzles, but a solution can be verified in an easy way. These puzzles can be made this way mostly because of the properties that hash functions have. See 2.1. The puzzles cost a certain amount of time and computational power. The users are rewarded if they can present a solution and thus prove that they have put some work in it. If they have proven their work, they are allowed to build on the blockchain. Therefore, the blockchain, and its blocks, represent a large amount of work [39]. With this method everybody who participates in the puzzle has a chance to propose the next block.

2.3.1 Proof-of-Useful-Work

Searching for solutions for the puzzles takes a lot of energy that is eventually wasted. Only one miner per puzzle is rewarded for the computations. Proof of useful work is questioning if there is a puzzle that is also useful to society. There are already some "volunteer computing" projects, but in order to be used for cryptocurrencies there are some challenges [32]:

- Not having an equal chance at winning (not progress-free). All spaces or parts of the puzzle need to have a equal chance to be a solution. Otherwise, people who are fast can choose a space in which they want to search. In this way, they will choose a space in which the chance of finding a solution is higher than in other spaces. These users will then always have a bigger chance to find a solution.
- An inexhaustible puzzle space is needed. Otherwise, at a given time there are no more solutions or puzzles. In cryptocurrencies this will mean that there can no longer be any transactions, because the transactions need to be verified. This is done by putting them in a block on the blockchain. But to make blocks, a puzzle is needed. Cryptocurrencies are based on this principle.
- The puzzle needs to be algorithmically generated, no centralized party should be needed. One of the reasons cryptocurrencies are being used is that there is no central party needed. If a central party needs to manage the puzzles, it could also decide to give some users a higher chance at finding solutions than other users.

2.4 Proof-of-Stake

Proof-of-Stake is another method to decide who is allowed to create the next block for the blockchain [19]. Instead of solving a puzzle like in Proof-of-Work, you prove that you own an amount of the cryptocurrency. To do this,

a user needs to lock up an amount of the currency into a deposit. Those users are called validators. Validators are chosen to propose the next block according to how much proof-of-stake they have. Instead of competing to solve a puzzle first and then being rewarded and able to propose the next block, a user/node is selected based on the amount of stake the user has. There are two different kinds of Proof-of-Stake [9]:

- ***Chain-based*** *proof-of-stake:* The chosen validator is allowed to create the next single block for the chain. This newly created block must point to a previous block in the chain, normally the block at the end of the blockchain.
- ***BFT-style*** *proof-of-stake:* In Byzantine Fault Tolerance proof-of-stake, after a block is proposed, all validators still need to vote on a specific block for that round. At the end of the round all validators permanently decide if a block is part of the chain or not. Blocks could still be chained together, but the consensus does not depend on this.

With chain-based proof-of-stake, the nothing at stake problem arises. Validators are only rewarded when they are chosen to create the next block. Validators will thus try to create blocks on multiple chains to gain more rewards. In order to stop this behaviour, punishments can be introduced. Validators are not allowed to create a fork or they will be punished, for example by having to pay a fee or by not getting their deposit back.
A big advantage of proof-of-stake is that it uses less energy compared to proof-of-work. The fact that nobody has to solve a puzzle, leads to saving computational power and thus energy.

Chapter 3

Bitcoin

When people talk about a cryptocurrency, there is a high chance that Bitcoin will be mentioned, since it is the best known cryptocurrency. In this chapter we will explain what algorithms are used and how they work, how the blockchain is applied in Bitcoin and how the mining of blocks works. At last we will discuss some attacks that are possible and the limitations of Bitcoin.

3.1 Introduction

Bitcoin is currently the most used and known cryptocurrency. In the past year, its value has risen very quickly. It has risen to a point that it is now worth more than $15.000 (see Figure 3.1) and quite frequently Bitcoin is mentioned in the news. It uses public-key cryptography to make and verify digital signatures for its transactions. The point of this is that only a user with the private key can sign a transaction to send some bitcoins to somebody else, but anybody in the network can validate a transaction by using the public key. People can create an online wallet or a wallet on some hardware that contains the private keys.

3.2 Signature Algorithms

We already mentioned the use of public key cryptography, but this is very broad. The specific algorithm that Bitcoin uses is Elliptic Curve Digital Signature Algorithm (ECDSA) [38]. This is a variation of Digital Signature Algorithm (DSA) and it uses elliptic curve cryptography. The signatures in this algorithm are very important for the structure of Bitcoin. With these signatures, the transactions can be checked if they are valid. And in this way, nobody but the owner of the bitcoins can send these bitcoins to another address.

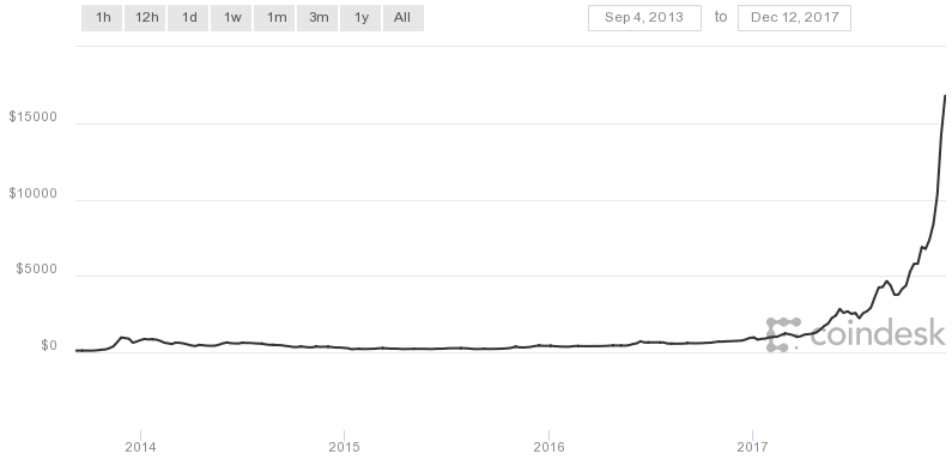

Figure 3.1: Bitcoin value compared to USD for the last few years [1].

3.2.1 ECDSA

To generate keys we need an elliptic curve and a base point. As a curve *secp256k1* is used to have a 256 bit security level and this curve is used as recommended in "Standards for efficient cryptography, SEC2: Recommended Elliptic Curve Domain Parameters"[35]. With this standard, the curve, base point G and the order of G, n is known. Now some private (sk) and public keys (pk) can be generated. The secret key is 256 bits long and the uncompressed public key is 512 bits and compressed 257 bits long.

Signatures

If a user wants to sent a transaction, this will only be valid if he signs his transaction with his private key[31]. The user signs the hash of the previous transaction and the public key of the next owner of the key together. We will call this message m. The algorithm is executed as follows with hash function H [28]:

1. Select a random value k with $1 \leq k \leq n-1$
2. Compute $kG = (x_1, y_1)$ and $r = x_1 \bmod n$, if $r = 0$, go back to step 1
3. Compute $k^{-1} \bmod n$
4. Compute $e = H(m)$
5. Compute $s = k^{-1}(e + sk \times r) \bmod n$, if $s = 0$, go back to step 1
6. The signature of message m is (r, s).

The signature has a length of 512 bits. Only the owner of the bitcoins and of the private key can sign a transaction sending his bitcoins, and the algorithm is secure because of it. No one can fake a signature because the transaction will not be valid.

3.2.2 Hashes

In the algorithm, most of the messages are hashed with a hash-function. In this way, the message can be of any length, but the input for the algorithm needs to be 256 bits and in this way any message can be signed. The hash function that is used is SHA-256, and sometimes when a shorter hash is needed RIPEMD-160 is used [10]. This is mostly used for creating addresses. In Bitcoin almost everything is hashed twice, double SHA-256 or first with SHA-256 and then with RIPEMD-160 for a shorter hash.

3.2.3 Randomness

There is one thing most important in the algorithms that are used, in order to be secure. When a random value needs to be chosen, this needs to be done in a really random way. If this is not the case the whole algorithm can be broken and signatures can be forced.

3.3 Transactions

Transaction

Transactions contain a few different elements all needed to make sure everything is secure. As described in chapter 3 of Bitcoin and Cryptocurrency Technologies [32], a transaction contains the following:

- The **inputs**: Contain a value that is the amount of bitcoins that will be sent to another address. The inputs have a hash of the previous transaction where the coins are coming from and this hash acts as a hash pointer to it.
- The **outputs**: This field contains a value of an amount of bitcoins that are send to a specific address (a public key) and the address itself.
- A **unique identifier**: This identifier is necessary to keep track of all the different transactions.
- **Signatures**: These are very important. Without the signatures a transaction cannot be valid. Every person that sends an input, needs to sign the transaction in order to be valid. In this way, nobody but the person who owns the coins can send them.

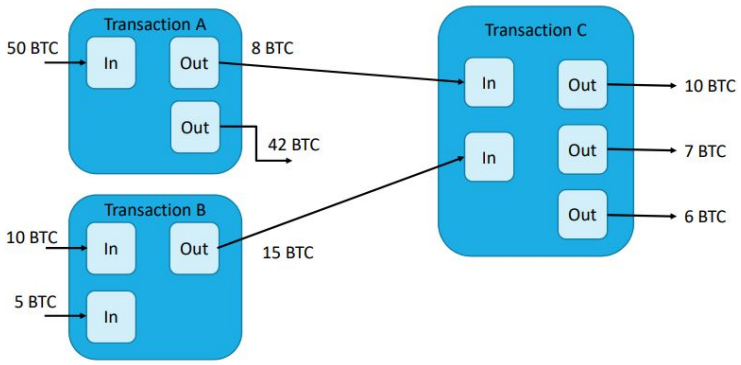

Figure 3.2: A Bitcoin transaction: sending money from one public key (address) to another one [34].

- **Metadata**: In this field some extra information is stored. For example, the size of the transaction, number of in and outputs, the hash of the transaction, which can be used as a unique identifier and a locktime parameter. This parameter can be used to block a transaction until a specific time or block number.

Both inputs and outputs can be more than one address. This input and output must be the same amount in total. Also see figure 3.2. Otherwise a coin can be double spent, which we will talk about later. It is possible that the output is a bit less than the input because of the transaction fees.

Blocks

Transactions are grouped together in a block. In this way, a lot of transactions can be added to the blockchain at one time. This is more efficient than only adding a single transaction. As transactions are being published to the network, a node collects these transactions to put them in their block.
A block consists of a block header, a hash pointer to some transaction data and a hash pointer to the previous block in the blockchain [32]. In the header, some information is stored related to the mining puzzle which is analysed in the next section. The header also contains a nonce, a timestamp and a few bits about how difficult it was to find the block. The hash pointer to the transaction data is a hash of the root of the Merkle tree that contains all of the transactions included in that block. Because of how the blocks are made up, the blockchain is made of two structures, namely a hash directing to the previous block, making the hash chain, and a hash tree which contains all of the transactions in that specific block.

3.4 Mining

There need to be requirements to be able to send transactions and mine new bitcoins in a way that is fair and secure for everybody. These requirements are [32]:

- *Fast verification.* This means that the puzzle itself should take a reasonable amount of time to solve, but anybody must be able to check if a found solution is indeed a valid solution to the puzzle. In order to be efficient, this validation should be as fast as possible. If the validation took a long time and everybody needs to do this, it would reduce the efficiency a great amount.

- *Adjustable difficulty* is a necessary property for the puzzle. The environment changes all the time and the calculating power in machines is improving fast. Thus, it is necessary to be able to change the difficulty of the puzzle or even the puzzle itself. This is done to make sure a solution to a puzzle can be found in a reasonable steady amount of time. With Bitcoin, this time is approximately ten minutes.

- *Progress-freeness.* This means everybody who is participating, should have a chance at finding a solution to a puzzle. This chance should be proportional to the amount of hash power used by a user to solve the puzzle.

The puzzle used in Bitcoin is a partial hash-preimage puzzle. To solve it, somebody has to find a preimage for a partially specified hash output. It needs to be below a certain target value. To find a valid solution a user just needs to try different values and compare it with the target value. The hash is SHA-256 based.
This partial hash-preimage puzzle satisfies the requirements. To verify it is a valid solution, someone only has to compare it to the target value. This validating can be done very quickly and does not need any difficult calculations. The puzzle can also be adjusted in difficulty. This can be done by changing the target space. If this space is made larger, the puzzle will be easier and if the target space is very small, the puzzle will be more difficult. Because a solution can only be found by just trying some values (brute force), everybody that participates with the puzzle has a chance at finding a solution. When someone finds a solution he is allowed to propose the next block, because he did the Proof-of-Work. (See 2.3.)

3.4.1 ASIC-resistance

Since mining started, there has been some specialized hardware on the market. An example of this are Application-specific integrated circuit (ASIC) miners [17]. These chips are specifically made for Bitcoin mining and are

very efficient. Because they can solve the puzzles relatively more quick than normal computers, the difficulty of the puzzles will rise. They have almost completely taken over the mining network, because for a normal computer it is not feasible any more.
A puzzle is ASIC-resistant if it reduces the gap between the most cost effective customized hardware and what normal general-purposed computers are able to do [32]. This means that "normal" people should have the same chance to win as "hardcore" miners.
This can be done with memory-hard or memory-bound puzzles. These puzzles need a lot of memory instead of a lot of computational power. For this you need to think about the time-memory trade-offs. If you take a smaller buffer, you will need more computing time and vice versa. We also need to check for the verification cost. If everybody needs to verify then this can slow down the acceptance of the solution, and this can lead to an increased risk of forks in the chain.
Not everybody wants to make the puzzles ASIC-resistant. Changes are risky since they could lead to a weaker algorithm that could be easier to hack. Another reason against changing it, is that the SHA-256 mining ASICs are already close to its modern limits and will likely not be improved much more in the future. This means it will probably not be very profitable for people to change it all now. The last reason is the irrationality for attackers, because if there is a 51% attacker, the attempt to an attack could lower the exchange rate and confidence in the currency. This would reduce the willingness to mine since the reward for mining with their hardware will be worth much less. (See 3.5)

3.4.2 Mining reward

If a miner finds a solution, he is rewarded with some bitcoins to compensate for the costs he made. In the beginning of Bitcoin the mining reward for finding a new block was 50 bitcoins. This reward will be halved every 210.000 blocks, this is roughly every four years [32]. Currently the reward is 12,5 bitcoin. Because of this halving, the total amount of bitcoins is limited. There are only new bitcoins created in the mining process and in no other way.
This mining reward is a reason why people participate in the blockchain. But when the limit of bitcoins is reached, why would anybody still participate in the hashpuzzle to create new blocks and thus make transactions happen. The answer to this is transaction fees. If somebody makes a transaction he has to pay a small amount. This is a transaction fee and the miner who mines that block will receive those fees. In this way it is still attractive for miners to mine new blocks.
Those mining rewards and transaction fees also make sure the users have good intentions with the network, because only the blocks that are valid

and are in the chain will be rewarded. If a malicious user mines an invalid block, he will not be rewarded because the other 'good' users will continue on a branch that only has valid blocks. And thus the invalid block will not be included in the chain and the miner will not be rewarded for it.

3.5 Attacks

There are two kind of attacks that we will discuss here, that are most important for cryptocurrencies.

Double spent attack

A user should not be able to spend a single bitcoin twice, just as with coins in the physical world. With real physical coins it is more obvious to see how someone would double spent a coin. A person would pay for some good or a service, but does not actually give the coin and then he spends the coin on something else.
In Bitcoin it is a little different. An attacker (Eve) creates two transactions, one in which she pays someone (Bob) for a product and another transaction in which she sends the (same) coins back to herself. Eve then sends the first transaction into the network and an honest node in the network will accept this transaction and put it in a block. Bob sees that the transaction is accepted and sends the product to Eve. After the acceptance of the previous block, another random node is allowed to propose the next block. What if this node happens to be controlled by the attacker, Eve? Eve can choose to ignore the block with the transaction to Bob in it and include a block which has the second transaction included that Eve made with the transaction to herself.
Bitcoins have a hashpointer that points to where the coins come from, and both transactions point to the same coins. Thus, only one of these transactions is allowed in the chain, because bitcoins are only allowed to be spent once. If such a double spent action happens, the chain is forked and the next node has to choose on which block it will build. Typically this will be the block which is proposed first. In this case, that will be the block with the payment to Bob. But in a network, not all messages arrive in the same order they were sent and it could be the second transaction block from Eve arrived first at some node. It is possible for these nodes to build further on that block because they see a transaction and do not know this is a double spent transaction. Both transaction are valid, but not at the same time. So if the next nodes decide to work on the malicious second transaction branch, Bob will not get his bitcoins because nodes will work on the longest valid chain. To prevent this it would be smart for Bob to wait a few blocks before he sends the product to Eve to be sure the block with the transaction for Bob is included in the chain. The longer Bob waits, the higher the chance

is that the block is in the blockchain. With Bitcoin it is common to wait for six 'confirmation' blocks.

51% attack

A 51% attack is not a single intention attack. It is more about what an attacker can do when he controls 51% of the mining power of the network [32]. He will not be able to steal bitcoins from other people, because in order to do this, he will have to revert the cryptography. And this is not possible. If an attacker creates an invalid block in which he "steals" some bitcoins from a user, he can keep pretending it is a valid block and keep building on it. But any honest node will see that the blockchain has an invalid block and will build on the last valid block, even if this is not the longest chain. This will create a fork in the blockchain. If the attacker tries to spend any of the coins, the receiver will not accept this block with that transaction, because the signatures are not valid and he only wants to work with blocks on the longest valid chain and not on a invalid one.
The attacker could hinder a certain person by not allowing any transactions of that person, or to that person, into a new block. Because the attacker controls most of the network he can refuse to build upon blocks that contain these transactions. But the attacker cannot prevent that the transactions are being broadcast to the whole network and thus the other nodes will see these transactions. Thus even if the attack succeeds and none of the transactions from a single person are being included in a block, this will be visible to the other users in the network.
The last kind of consequence of a 51% attack is probably the most disastrous one for Bitcoin itself. Namely destroying the confidence in it. If someone owns 51% of the network and there are some attempted attacks, people will likely lose their confidence in the cryptocurrency. Even if there are no attacks, but it is known that one party holds 51% of the hashing power, it is very likely that the confidence will drop thoroughly and this will drop the exchange rate.

3.6 Limitations

Bitcoin has some (built in) limitations. Most of these were chosen when Bitcoin was proposed in 2009 [32].
The total number of bitcoins is fixed. It is fixed on 21×10^{14} bitcoins. This total, together with the structure of the mining reward will probably not be changed in the future. This is due to the economic implications this will have. A limit on the total amount of bitcoins is also necessary to give the coins any value.
The blocks have a maximum size of a megabyte and a minimum size of 250 bytes. This is hard-coded into the system. Because of this, a block is limited

to roughly 4000 transactions and this translates to about 7 transactions per second, because circa every ten minutes a new block is found.
Something people are afraid of is that the used cryptographic algorithms are fixed and that these might be broken in the future. This is not a problem until the algorithms are indeed broken. A solution could be to extend the scripting language to support new cryptographic algorithms.
The potential changes in the protocols could result in some hard or soft forks in the chain because the network consists of several nodes that will not all update at the same time or update at all.

Hard forks The change in the protocol would make some blocks that would be invalid in the previous version, valid in the new version. This will split the blockchain and nodes will be working on a different branch depending on the version they are working with. This will continue until all the nodes have updated and consider the new valid blocks valid, considering the new protocol. The two branches cannot be joined, so eventually every node is forced to upgrade, because otherwise it is cut out of the Bitcoin network.

Soft forks Another change could lead to soft forks in the chain. The protocol change makes the transaction validation more strict and the updated nodes will now say some blocks are invalid while with the old protocol they were valid. This ensures that the updated nodes will work on a different branch with only validated blocks considering the new protocol and the old nodes will just accept any valid block. Because the nodes chose to work on the longest valid branch an eventual permanent split is avoided.

Chapter 4

Ethereum

In this chapter we will discuss the next cryptocurrency: Ethereum. This currency is rising in popularity and is becoming more known. We will discuss the different algorithms and like in the previous chapter we will look how the transactions, blocks and the blockchain are applied in Ethereum. At the moment Ethereum is still working with Proof-of-Work but is planning to change this. This will be discussed and explained. At last we will discuss some of the possible attacks on this system.

4.1 Introduction

Ethereum is another uprising cryptocurrency that works with a blockchain. But Ethereum is more than just a cryptocurrency, since the blockchain can be used in multiple ways and Ethereum is using that property. Instead of an online currency only, people can also use smart contracts and different applications like games, (domain-)name registration or a decentralized file storage. This is possible because the developers of Ethereum provided a built-in Turing-complete programming language for the blockchain. The programming language can be used to create "smart contracts" which are used to create different applications. All of this is decentralized, because of the technique from the blockchain [20].
Ethereum is more browser-like to access in comparison with Bitcoin. There are a few different browser platforms from which you can access and even some plugins [18].
Ether is the main currency of Ethereum and can be used to pay for computation or transaction fees. On the 2014 presale of Ethereum, the total supply of ether and the rate of issuance was decided by the donations that were gathered [5]. The results were:

- 60 million ether. This amount of ether was created for the contributors of the presale.

- 12 million (20% of the 60 million) were created for the development fund. Most of this ether went to early contributors and the developers of Ethereum. The remaining of this 20% ether went to the Ethereum Foundation.
- 5 ether as the mining reward for the miner. This is now reduced to 3 ether for a block.
- The uncle/aunt reward. This reward is for other miners. If these other miners find a solution, but their block is not included in the chain, they will receive this reward.

The total amount of ether is not infinite. The issuance of ether is restricted at 18 million ether per year. This means the absolute issuance is fixed, but the relative inflation is decreased every year. At some point, the rate of new ether created will reach the average amount of ether that is lost every year, for example by misuse, death of holders or key loss. This loss and issuance will be in balance.

4.2 Algorithms

Ethereum uses different algorithms, in this section we will briefly explain how they work and what they are used for.

4.2.1 ECDSA

Ethereum uses the EDCSA algorithm with the secp256k1 curve [41]. This is the same algorithm and curve that is used with Bitcoin (See 3.2.1). Generating the private key is done by randomly selecting a positive integer of 256 bits. With the private key, a public key can be generated and this key has a lentgh of 512 bits [41]. Ethereum uses a seperate address and not just the public key as an address. This address can be computed by hashing the public key and taking the last 160 bits of this hash [24]. This address represents the account of a user.

4.2.2 Mining algorithms

Right now the only way to create new tokens is via mining. The mining also secures the network by creating, verifying, publishing and propagating blocks in the blockchain [7]. It secures the network by checking that only valid blocks with valid transactions are allowed in the blockchain. After a block is accepted in the chain it cannot be changed and in this way the miners secure the network by only allowing valid transactions in their proposed block.

Proof-of-Work

Ethash is the algorithm used for Proof-of-Work mining [41]. The miner needs to find a nonce that is below a desired target value. This nonce proves that a particular amount of work has been done. Before the actual mining algorithm can be performed, additional computation needs to be done. There is a total of four computational parts needed to mine a block [3]:

- **Seed:** The seed is computed for each block by scanning through all block headers up until that particular block. The computation of the seed hashes (and datasets) can be done in advance, before the actual mining starts and it is recommended to mine faster and smoother.
- **16 MB cache:** With the seed a pseudo random cache of 16 MB can be computed. The items in this cache are 64 byte values.
- **Dataset of 1 GB:** By using the previously generated cache, a dataset of 1 GB is generated. This dataset has the property that each item in it only depends on a small amount of items in the cache. This property makes the verification procedure easy and fast.
- **Mining:** The last part is the actual mining. A (random) nonce needs to be chosen and this nonce will be put in a mix. Random slices of the dataset are put in the mix and the values are hashed together and compressed. The value that comes out of this algorithm is compared with the target value. If this value is lower than the target value, a valid nonce is found. If the value is higher, then the next nonce is chosen and the algorithm will be computed again till a valid nonce is found.

The algorithm Ethash is memory hard. That means it requires an amount of memory to be solved efficiently, but verification can be done with low-memory. Verification uses the stored cache to regenerate specific parts of the dataset that are needed for the solution. In this way it is not necessary to regenerate the whole dataset, which takes more time.

Proof-of-Stake

At the moment, the Proof-of-Stake algorithm is still in development and not everything is definitive. The developers of Ethereum are working on the consensus algorithm called "Casper" [5]. Casper is a security-deposit based economic consensus protocol [43][22]. Bonded validators place a security deposit in order to participate in the consensus by proposing and producing blocks. There is a set of bonded validators. A validator is a user that participates in the mining process by validating transactions and creating blocks. A user can participate and become a validator in the set by depositing an amount of ether as stake. The security deposit has to be some

minimal amount of ether, now set on 32 and can be a maximum deposit of 131072. Depositing a stake is called bonding. After the deposit has taken place, the user is a bonded validator. For each block, a list of validators is generated from the set of bonded validators. The list is pseudo-randomly chosen, with the randomness weighted by the amount of stake that a validator has initially deposited. Thus when a validator has deposited 10% of the total amount of deposited stake, he has a 10% chance at being chosen to propose the next block. From this priority list, the first validator is allowed to propose a block. If the first validator does not respond or does not respond in a set amount of time, the next validator on the list is allowed to propose a block, and so on.
All validators have a validators' code. With this code they can check if the signatures in the block are valid. The validators need the validating code to verify the block they might accept in the chain. The code also applies as an identifier for the validator.
After the proposal of a block, all of the validators still need to vote if that block will actually be accepted in the chain. The validators do this for blocks at every height (block number). A validator can vote on a block by broadcasting a commit to that block. In the commit, they agree to lose their deposit in all histories where that block is not included in the chain. A validator votes on a block that has the highest value-at-loss. They want to bet on this chain/block because that is the chain where the most stake is. The chain is weighted by the length and the amount of ether actively validating it. If a block does not get included in the chain, the validator that proposed the block will lose money equal to the block reward. Thus, validators will only make a block, if they are more than 50% sure that the block will be included. And thus, this way discourages validating blocks on multiple chains, because the validator will not gain anything and will only lose money.
If a validator wants to withdraw, he needs to wait until the next epoch (a period of a large amount of block) and then he will get his deposit back, plus rewards minus penalties. The epoch length will be set on 10800 blocks and this will take about 12 hours.
If something is not right, the validator will lose its deposit according to the slashing conditions [9]. These rules determine when a validator has misbehaved. For example, by voting for multiple blocks on the same height. If they have misbehaved their entire deposit will be removed.

4.2.3 Hashes

In Ethereum, a variant of the family of SHA3 hashes is used. More specifically, Keccak-256 or Keccak-512 as defined in [3]. In Ethash there is also a non-cryptographic hash function used, called Fowler–Noll–Vo hash function (FNV hash function) [27].

A FNV hash function is available in 32-, 64-, 128-, 256-, 512-, and 1024-bit lengths. These different versions come with an initial FNV offset basis and FNV primes. The offset basis is used as start value of the hash. Then, for each byte in the data that needs to be hashed, multiply the *hash* with a FNV prime and XOR the result with the byte of data. When all data is processed, the hash is finished.
Ethash uses the FNV algorithm with a small change, but the basics are the same. The function is used to provide a data aggregation function [4]. Data aggregation takes multiple values and forms a single value. In Ethash the function is used to create the dataset.

4.3 Account

A user needs an account to participate in the Ethereum network. An account can be created by downloading the Mist Ethereum Wallet [6], which is the wallet created by Ethereum itself. When starting the software, a user can create a new account. An account can also be created by using the command line after downloading. Anybody can create as many accounts as he likes. Each account will have its own public and private keys.
All accounts make up the state of Ethereum. The blockchain on a specific time is in a specific state. Each account has a 20 byte address and can add state transitions. As described in the Ethereum Whitepaper [20], an account has four different fields:

1. A **nonce**. The nonce is used as a counter, to make sure that every transaction is only processed once.

2. The current **ether balance**. The amount of ether the user currently has.

3. **Contract code**. If the account is externally owned, this field is empty. If the account is an contract account, this field contains the code that will be executed when a message is send to the account.

4. **Storage**. The storage is by default empty, but can be filled and read from by transactions and by contract codes.

There are two different kind of accounts: externally owned accounts and contract accounts. The first one is controlled via private keys from a user and the contract accounts are controlled by their contract code. The externally owned accounts have no code and can send messages by creating and signing a transaction. A contract account activates its code when it receives a message. This can mean it reads and writes to its storage and/or sends messages and/or creates new contracts.

4.4 Transactions

Transactions in Ethereum are similar to the transactions in Bitcoin, but there are also a lot of different features in Ethereum [20]. The transactions can be made externally by a user or by a contract. The transactions contain different fields:

- A **nonce** that represents the number of transactions sent by the sender.
- **Recipient** of the message is the user who will receive the amount of ether being sent. The recipient is uniquely characterized by an address.
- **Amount of ether** that is sent over from the sender to the receiver.
- **Data** that can be sent over. This field is optional and can be empty.
- **Signature** of the sender. The signature is necessary to verify that the sender is the person who owns the tokens and agrees with sending them over.
- **Value StartGas** indicates the limit of possible executional steps.
- **Value GasPrice** indicates the fee that has to be paid to the miner per step.

The last two values together are used to compute the transaction fee for the transaction, which we will explain more about in the next section.
When a transaction is formed, it is sent to the network and can be put in a block.
If a contract account sends a transaction, it is normally called a "message". A message has the same fields as a transaction except for the GasPrice field. When a contract account receives a message or transaction, it's contract code is executed. When a 'Call' opcode is executed in the code of a contract, a message is created. This message is sent to a recipient and its code will be executed. The sent message can be a response the the original sender of the message/transaction. In this way, contracts can communicate with other contracts and external users, just like an external user can send transactions.
Each transaction contains a nonce, this nonce needs to correspond with the sender's account current nonce in order to be valid [41]. If the nonce is not valid, the transaction is also not valid, and will therefore not be send. This check prevents double-spending.
If an attacker tries to spend his coins in a second transaction, the nonce needs to be the same as well. Otherwise it would just be another new transaction with a new nonce. Suppose the attackers account nonce is x. The transaction to Bob of 5 ether will have the transaction nonce x, assuming

everything is still okay for now. After this transaction is accepted, the attacker wants to double-spend these 5 coins. He sends a transaction with transaction nonce x, but his account nonce is now $x+1$, and thus the nonce is invalid and the transaction will be invalid and not accepted.

4.4.1 Transaction fees

Transaction fees are added into the system to prevent attacks. It makes a potential attacker pay for resources that he used. To compute the transaction fee, the values StartGas and GasPrice are used [20]. Almost every computational step costs 1 gas and the maximum amount of steps is indicated by the value StartGas. There is also an additional fee of 5 gas for every byte in the transaction data. The transaction fee is computed in the following steps:

1. Transaction fee = StartGas $\times$ GasPrice.
2. This Transaction fee is subtracted from the sender's account.
3. Gas = StartGas minus the gas fee per byte in the transaction data.
4. The amount of ether that is sent over, is subtracted from the sender and added to the account of the receiver.
5. The remaining gas gets added back to the sender's account.
6. The fees that paid for the gas that is consumed, are sent to the miner of the block.

When the sender does not have enough ether on their account to subtract the initial fee, an error will be returned. And thus the message will not be sent. When a message or transaction is sent, it is possible that the code takes longer to execute than the limited steps allow. When this happens the computation will be reversed, but the fees will still be paid to the miner. Another cause for failure and thus reversal, can be a insufficient amount of ether on the sender's account to send the amount of ether over that is mentioned in the transaction.

4.5 Contracts

A contract account has a contract that contains code. This code will execute when the contract account receives a message or transaction [20]. The code is written in low-level stack-based bytecode language, named Ethereum virtual machine code (EVM-code). The code consists of a series of operations, with each operation being a byte long. The operations have access to three different types of space where data is stored or can be stored:

- **Stack**, last-in-first-out container with push and pop operations.
- **Memory**, a byte array with no length limit.
- **Storage**, long-term storage in the form of key/value. The storage will not reset when the computation of the code has ended like the stack and memory.

The code also has access to the other fields of the message and block header data.
A contract can be created with a transaction, the receiver of the transaction is empty and the transaction contains an init field [41]. The init field is a unlimited size byte array that assigns the EVM-code for the account. The init field is executed only once at account creation and gets discarded after that.
The contract code is executed by all nodes that download and validate the block which has that transaction or message in it [20]. The execution of the code is part of the state transition function, which is part of the block validating algorithm.

4.6 Blockchain and blocks

The Ethereum blockchain is quite similar to the blockchain of Bitcoin. The blocks do contain a few more and different values. As listed in the yellow paper [41] the different elements of a block are:

- **ParentHash**: The Keccak 256 bit hash of the parent block's header.
- **Address of miner**: The fees will be transferred to this 160 bit address after the block is successfully mined.
- **StateRoot**: This field contains the Keccak 256 bit hash of the root node of the state tree, after all transactions are executed and finalisations applied.
- **TransactionRoot**: This field contains the Keccak 256 bit hash of the root node of the tree structure with transactions of this block.
- **Difficulty** of the block. The value can be calculated from the previous block's difficulty level and the timestamp.
- **Number**: A number indicating the number of ancestors of the block and thus the height of the block. The first block has a number of zero.
- **GasLimit**: Indicates the current limit of gas that can be spent per block.

- **GasUsed**: Indicates the amount of total gas that is used in all transactions in this block.
- **Timestamp**: A value that shows the time in seconds since unix epoch.
- **ExtraData**: A 32 byte array that contains relevant data for this block. Can be empty.
- **MixHash**: A 256 bit hash that proves, combined with the nonce, that a sufficient amount of computational work has been done.
- **Nonce**: A 64 bit hash that proves, combined with the MixHash , that a sufficient amount of computational work has been done.

There are a few other fields included in a block, but these elements are not necessary for understanding how a block works in the blockchain. The biggest difference is that the blockchain of Ethereum contains a copy of the transaction list and one of the most recent states of the blockchain [20]. This may seem inefficient because the whole state needs to be stored. The state is stored in a tree. This tree is similar to the (Merkle-)tree used in Bitcoin, but it is a different version of a Merkle Tree, namely Merkle Patricia Tree (See 4.6.1 and 2.1.2). The 'normal' Merkle tree is useful for the transaction tree in the block. Once a transaction is added, it is not needed to change it and thus no editing of the tree is necessary. But for the state tree, updates are necessary. Balances, storage and code are able to change in all accounts. In order to efficiently update and create new accounts, the structure of the tree should allow for fast adding, editing and delete operations. The Merkle Patricia Tree is what comes closest to this [21].
To confirm that a block is valid, one needs to check all values of the block [20]. Does the previous block exist and is this block valid? Are the timestamp, block number and other values valid? Is there a Proof-of-Work and is this correct? When all of these values are valid, the new state can be set. The transaction list can be set if no application returns an error and the GasLimit is not surpassed by the total amount of gas consumed. The final state can be set when the fees are paid to the miner. When the Merkle tree root of the final state is equal to the final state provided by the block header, the block is valid. Otherwise, the block is not valid.

4.6.1 Merkle Patricia Tree

In Ethereum, a modified Patricia tree is used called Merkle Patricia tree. In this modified tree the keys are compared in hexadecimal form instead of binary form. There are three different kind of nodes implemented according to the yellow paper [41]:

- **Branch node**: The branch node is used when the keys have a different value. The branch nodes have a list of length 17. 16 elements for

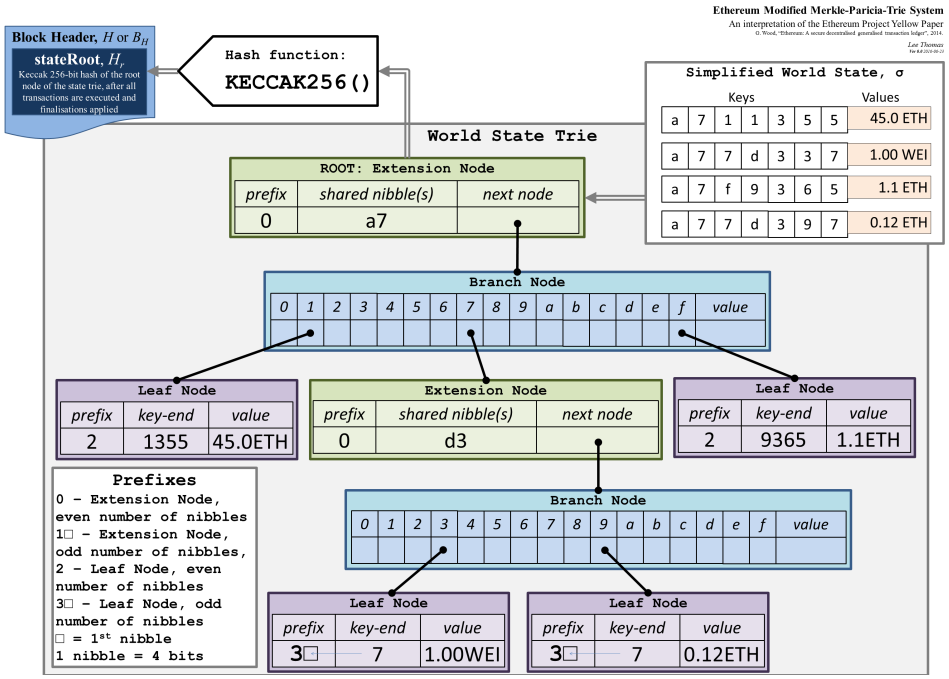

Figure 4.1: Modified Patricia tree used in Ethereum as interpreted of the Ethereum project Yellow Paper by Lee Thomas [11]

.

possible hex characters and the final element can hold a value if there is key that ends at that branch node. Thus a parent node is able to have 17 children instead of two in a normal Patricia tree.

- **Extension node**: This node is used when more than one key contains a shared part (called a nibble). The node keeps 2 values. The first value contains the part of a key that is shared among at least two different keys. The other value directs to the next node.

- **Leaf node**: A leaf node is used when there is only one key left that follows this path. The node keeps 2 values, one value for the last part of the key, which has not already been accounted for in previous branch nodes and extension nodes. The second value contains the actual value that is paired with the key.

Both extension and leaf nodes contain a value which indicates the hex-prefix encoding method. See Figure 4.1 for a more graphical explanation of the modified tree. To make the tree cryptographically secure, the hash of a node is used to reference it. This way, the hash of the root node makes a unique identifier of the entire tree. This makes it a *Merkle*-Patricia tree.

4.7 Proof-of-Work to Proof-of-Stake

At the moment of writing, Ethereum is still based on Proof-of-Work mining like Bitcoin. The Ethereum developers want to switch to Proof-of-Stake in the near future. For this to happen there will be a hard fork in the blockchain (See 3.6). And thus it needs to be planned and thought through in many ways for it to work and still be interesting for people.

4.7.1 Proof-of-Work

The used algorithm Ethash (see 4.2.2) is chosen and implemented because it is sequential memory hard and this makes it basically ASIC resistant [41]. Determining the nonce requires the use of a lot of memory and bandwidth and the algorithm works in such a way that only one nonce can be computed at a time. Ethereum developers also wanted that verification can be done by light clients (smaller clients on a laptop or mobile device for example). But the process of running the algorithm should be slower with a light client than with full client.
Mining a block with Ethash takes approximately 12 seconds [7]. This is a lot faster than the mining in Bitcoin, but Ethereum still has possiblities to be even faster. For consensus time Ethereum does not have a specific number of blocks to wait, like 6 blocks in Bitcoin. The longer a user waits after the block is accepted, the more likely it is the block will be in the actual chain. It is common to wait at least 12 blocks (about 2 to 3 minutes) [12]. Waiting longer gives a higher level of security.

Mining reward

The miner who mines the 'winning' block receives multiple rewards [7]:

- A static reward, consists of 3.0 ether. The reward was 5 ether before the Byzantium update [36]. In this update of 16 October 2017, there was a hard fork and multiple changes were applied. An example of these changes are the reduction of the mining reward, faster processing for transactions and improvements for the smart contracts.

- All of the gas spent within the block. This means all of the gas that is consumed by the execution of all the transactions in the winning block. This consumed gas is compensated for by the senders of the transactions. Also see 4.4.1. The gas-cost is sent over to the miners account as part of the consensus protocol.

- There is an extra reward if the miner has included Uncles as part of the block. The reward is an extra 1/32 per Uncle that is included. An Uncle is a stale block with parents that are ancestors of the including block and go back a maximum of six blocks in the chain.

4.7.2 Proof-of-Stake

Ethereum developers plan to switch from Proof-of-Work to Proof-of-Stake for multiple reasons and benefits [9]:

- There is a lot less electricity consumption. In 2014 Bitcoin used as much electricy as Ireland in a year [33] and this year it is said that "one Bitcoin transaction now uses as much energy as your house in a week" [30]. Ethereum is using more and more electricity as well. Ethereum is using almost as much as Zambia in a year [15]. See Figure 4.2. With Proof-of-Stake the energy consumption will be lowered a major amount, because there are no heavy computations being done. Thus the amount of electricity needed is noticeably less.
- Because there is less electricity needed and mining costs less, it is possible to produce fewer coins. The mining reward is primarily given to compromise for the costs made with mining. If these costs are reduced, the reward can be lower and people will still be willing to participate in the network.
- In Proof-of-Stake there are more possibilities to discourage centralized parties being formed. If such a party does get formed it is easier to prevent them from acting in a mischievous way.
- There is a reduced risk for centralization. All organizations are getting the same proportional gains as others. If a company invests 1000 coins, it will give them 10 times less in return than a company which have invested 10.000 coins.
- There is a possibility for penalties. These penalties make various forms of 51% attacks immensely more expensive to execute. Even more expensive than Proof-of-Work.
- There is a possibility that the block time can be even faster. If this block time is reduced, it is possible to process more transactions in a smaller time period. At the moment, the developers estimate that a block time of 4 seconds would be possible [22]. With Proof-of-Work the block time is 12 seconds.

Mining reward

The validator that provides the block that gets accepted in the chain is rewarded. This reward consists of the total amount of ether that is in the current active validator set *times* a reward coefficient *times* the set block time [22].
$Reward = total\_ether\_active\_set \times reward\_coefficient \times set\_block\_time$

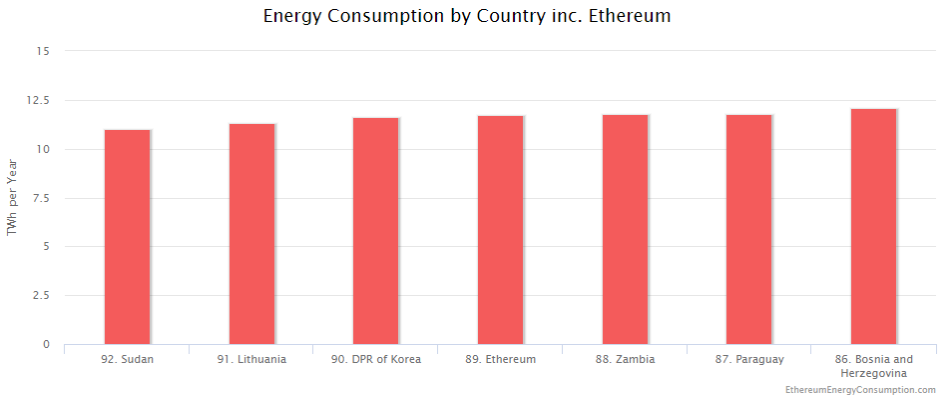

Figure 4.2: Energy consumption of different countries and Ethereum in 2017 [15].

The validator will also receive the transaction fees from that block as a reward.

4.8 Attacks

In this section we will discuss two possible attacks on Ethereum. The double spent attack is also possible but is the same as in Bitcoin. See 3.5.

4.8.1 51% attack

A 51% attack could also happen in Ethereum. Since Proof-of-Work is still used, the attack and consequences are the same as the 51% attack in Bitcoin. See 3.5. When Ethereum changes to the Proof-of-Stake algorithm Casper, the 51% attack is slightly different. If an attacker would happen to have deposited 51% or more of the stake in the current set, the attacker will very likely be chosen to propose the next block. He can choose to ignore previous blocks that are not yet finalized or hinder certain transactions, but every validator still needs to vote on the block he proposes. If the block is not accepted in the chain, the validator gets a penalty and could lose his whole deposit. The chance a 51% attack is profitable is very low, because of the penalties and therefore losses the attacker will face [9]. The attacker must have invested a lot of money to gain 51% of the total amount. If he becomes malicious, it will affect him probably more than a user who has invested only a small amount. When the attack is successful, the value of the tokens will probably lower and the attacker will also suffer from this. Another consequence can be that the confidence in Ethereum will be lower and less people will use it, and therefore its value will drop even more.

4.8.2 Denial-of-Service attack

The second attack we will discuss is a Denial of Service attack. The transactions and messages that are sent over contain executable code which can contain loops [20]. An attacker could create a code with an infinite loop. If a user executes this code, he would get stuck in this infinite loop and is not able to do or send anything else. This attack is prevented by the StartGas value, which indicates the maximum amount of steps that the code is allowed to execute. The transaction fee also makes the sender, in this case the attacker, pay for the code that is executed. This fee makes it less attractive for an attacker to execute such an attack. The chance a Denial of Service attack will succeed is almost zero, and in case it is launched successfully it would be very expensive for the attacker.

Chapter 5

Comparison

In this chapter we will compare the two cryptocurrencies discussed in the previous chapters.
To make everything clear, the most important aspects that will be compared are put in table 5.1 for a quick overview. We will discuss these points separately first in different categories.

5.1 Application

Bitcoin is only used as an online currency. Ethereum is also used as an online currency, but its blockchain is also used for other decentralized applications like voting systems, file storage, (domain-)name registration and even games. This makes Ethereum very versatile and useful on multiple levels besides being a online currency.

5.2 Algorithms

Bitcoin and Ethereum both use the same curve and signature scheme, namely *secp256k1* and *ECDSA*. The difference in the algorithms of the blockchain is the hash function used. In Bitcoin SHA-256 is used and in Ethereum Keccak-256 or Keccak-512 are used. Keccak is a variant of SHA3. The difference between these two functions is the class of algorithms they fall under. SHA-256 is a Merkle–Damgård construction[13] and Keccak falls under the Sponge functions[14]. Both of these functions are still secure and give both a security level of 128 bits.
Because both algorithms use ECDSA for signatures and use the same curve, the keys are similar and have the same length. Bitcoin is different in the way that it sometimes uses a compressed form of the public key. Another difference is the way an address is computed. Ethereum has a specific way to compute an address and Bitcoin uses the public key as an address and has the possibility to use another hash function to create a smaller hash of

160 bits. This length of 160 bits is the same size as an address in Ethereum. The signatures are almost the same size, 512 and 520 bits, but the Ethereum signature is a bit longer due to an extra value that is included in the signature.
Ethereum messages contain a field for executable code. This field makes a big difference in the messages. Bitcoin transactions only have very limited space for extra data. The transaction mostly contains information about the transaction itself.

5.3 Duration

The duration of the processes of mining and transactions is quite different. Ethereum is noticeably faster than Bitcoin. Ethereum can process 2 times more transactions per second than Bitcoin is able to do. When Ethereum switches to Proof-of-Stake, the block time will likely drop even more: from 12 seconds to 4 seconds to create a block. Bitcoin has a block time of 10 minutes. This also influences the consensus time. When can a user be sure that his transaction is included in the blockchain? In Bitcoin this takes about an hour. With an Ethereum transaction you should wait about 3 minutes to be sure that the block is included. Therefore, we can see that Ethereum is a lot faster in processing.

5.4 Mining

Right now the mining process is quite similar between the two cryptocurrencies. Both make use of Proof-of-Work. They do use a different puzzle to let users prove they have done some work. When Ethereum switches to Proof-of-Stake, the mining process will be completely different. The miners do no longer prove they solved a challenge, but provide a stake which proves the user has Ethereum. Then the miner or validator can be chosen to propose a block.
In both cryptocurrencies, the miners get a reward when their proposed block gets accepted in the blockchain and in both currencies the reward is the only way new tokens are created. The amount of the mining reward the miners get is different in Bitcoin and Ethereum. The reward for a mined block in Bitcoin started with 50 Bitcoins per block, this is now 12,5 Bitcoins. Every 210.000 blocks this reward is halved. Because the reward is halved roughly every four years and the mining reward is the only way new tokens are created, the total amount of Bitcoins is limited to $21 * 10^{14}$ bitcoins. Ethereums mining reward is variable, because it consists of various parts. It does have a static part of 3 ether per block. The reward also covers the gas that is consumed in all transactions in the block. The static reward means there is no limit to the total amount of Ether, but the creators of Ethereum have

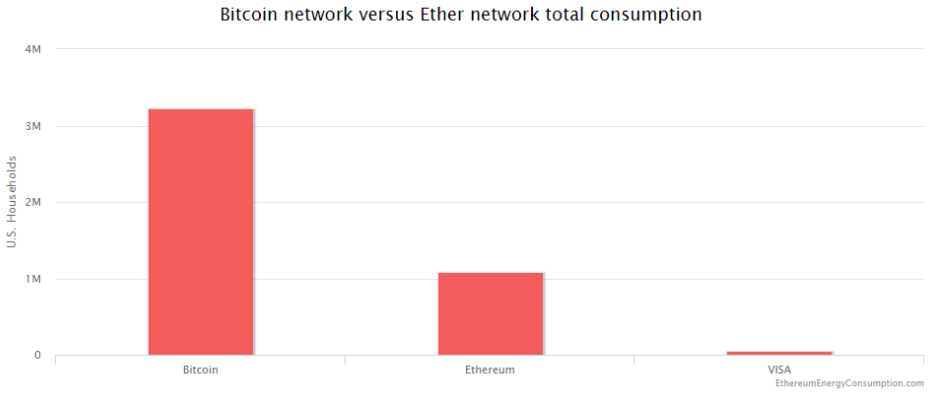

Figure 5.1: Energy consumption of Bitcoin, Ethereum and VISA [15].

limited the amount of new tokens that are created in a year to 18 million ether. It is possible for tokens to get lost, by loss of keys and codes, death of holders or misuse. In Proof-of-Stake, users can lose their whole deposit if they misbehave. And thus in Ethereum these losses will be compensated for by not having a limit on the total amount of ether that will be created.

5.5 Energy

Bitcoin uses a lot more energy than Ethereum. See also figure 5.1. Ethereum is also using more and more energy, but Ethereum will probably drop a lot in energy consumption once it has switched to Proof-of-Stake. The mining takes a lot of computational power and thus uses a lot of energy. In Proof-of-Stake these computations are not necessary and the energy consumption will be noticeably less.

5.6 Overall conclusions

Bitcoin is the first cryptocurrency to grow this big and is the best known all around the world. Ethereum is growing fast and becoming more known to the world and might surpass Bitcoin in size since it is faster and uses less energy now and will even use less energy in the future. The ability to use Ethereum for more than just an online currency also makes it more versatile and useful for people. The step from Proof-of-Work to Proof-of-Stake might give Ethereum the push it needs to grow even bigger.

5.7 Comparison table

Comparables	Bitcoin	Ethereum
Application	Online currency	Online currency and other various applications (e.g. File storage, games, voting, (domain-)name registration, etc.)
Key lengths	Public key = 512 bits (compressed: 257 bits) Private key = 256 bits	Public key = 512 bits Private key = 256 bits Address = 160 bits
Hash function	SHA-256	Keccak-256 or Keccak-512
Signature size	512 bits (64 bytes)	520 bits (65 bytes)
Used curve	secp256k1	secp256k1
Signature scheme	ECDSA	ECDSA
Security	128 bits	128 bits
Block time	10 minutes	12 seconds
Consensus time	$\approx$ 1 hour	$\approx$ 3 minutes
Transactions per second	7 per second	15 per second
Mining	Proof-of-Work	Proof-of-Work to Proof-of-Stake
Mining reward	12,5 Bitcoin per block	3 Ether plus some other rewards
Maximum of tokens	$21 * 10^{14}$	18 million per year
Energy consumption per year	$\approx 3,2$ million households	$\approx 1,1$ million households

Table 5.1: Comparison table

Chapter 6

Related Work

There are no real detailed comparisons made yet to really look at the differences in Bitcoin and Ethereum. Some small parts are compared in different articles or talks and Ethereum was started because of an interest in Bitcoin. At the start of the Ethereum whitepaper [20], Vitalik Buterin talks and explains about various concepts of Bitcoin and later how they are used in Ethereum.
Bitcoin is very well explained in the book "Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction" [32]. In a late chapter they discuss multiple altcoins and explain Ethereum in a bigger section. They do show some differences between Ethereum and Bitcoin but a good technical comparison is not made.
There are multiple articles that (mostly) compare the application of both currencies and the blockchain technology.
Want.nl [29] has an article that compares Ethereum to Bitcoin and talks about the things that are different in Ethereum. The blockchain is used in a different way and the possibility for smart contracts is there and is the main function for the blockchain. These contracts and the blockchain make decentralisation possible on more levels than only a decentralized online currency like Bitcoin. The article does state that fluctuation and scalability is something both currencies have to cope with.
Another article from *Investopedia* [16] gives a small introduction of both currencies Bitcoin and Ethereum, and of the technology of the blockchain. The article states that there are technical differences like the block time and used algorithms, but does not explain more about this and why these differences exist. The article talks mostly about the difference in the purpose of the two cryptocurrencies. Bitcoin is only used as an online currency in which the bitcoins replace physical money like dollars or euros. Ethereum can be used for a lot more different applicications and uses its currency ether to pay for the transactions and the fees to run these applications.
Businessinsider.com [26] describes multiple cryptocurrencies and displays

an info-graphic with the most important attributes of the currencies, which also indicates the differences between them. It also shows a part with the differences in the values of the cryptocurrencies. The info-graphic gives a neat overview of the different currencies, but there is no real explanation of the separate parts.
Learn.onemonth.com [23] gives a bit more detailed comparison. First they give a basic explanation of Bitcoin and Ethereum and what the purpose of the currencies are. Then a table is given in which some more technical details are compared. The table mentions the inventors, supply of coins, units, block time, issuance, purpose and price. Although there are some technical details in there, these are not explained or just very briefly.
In the talk of Bart Preneel at 4 September 2017 [34] a better technical comparison is made with a better explanation of the currencies. The talk covers many technical details of how Bitcoin works and Ethereum is also explained, although much shorter and less technical. Then a list is given of the differences in the technical details. This list is mostly numbers that are compared like block time, transactions per block and reward, but also contains other comparisons like what algorithm is used for mining and plans with proof-of-work.
There are also other comparisons regarding cryptocurrencies. In "On the Complexity and Behaviour of Cryptocurrencies Compared to Other Markets" [40] by Daniel Wilson-Nunn and Hector Zenil, Bitcoin and Litecoin are compared. They are not compared on a technical level, but how their behaviour compares with other currencies on the stock and metal markets.
We wanted to merge and expand these findings into one big explanatory and comparing research to bring all of these findings together and explain more about the cryptocurrencies itself. In this way one does not have to look at so many different articles to have an idea how these cryptocurrencies work and what the differences between them are.
In later research more cryptocurrencies could be added and compared.

Bibliography

[1] Bitcoin chart, coindesk. https://www.coindesk.com/price/. Accessed: 2017-12-12.

[2] Blockchain. https://en.wikipedia.org/wiki/Blockchain. Accessed: 2017-12-06.

[3] Ethash. https://github.com/ethereum/wiki/wiki/Ethash. Accessed: 2017-11-29.

[4] Ethash design rationale. https://github.com/ethereum/wiki/wiki/Ethash-Design-Rationale. Accessed: 2017-12-12.

[5] Ether. https://www.ethereum.org/ether. Accessed: 2017-12-12.

[6] Ethereum. https://www.ethereum.org/. Accessed: 2017-12-30.

[7] Mining. https://github.com/ethereum/wiki/wiki/Mining. Accessed: 2017-12-12.

[8] Patricia tree. https://github.com/ethereum/wiki/wiki/Patricia-Tree. Accessed: 2018-01-03.

[9] Proof of stake faq. https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ. Accessed: 2017-12-19.

[10] Protocol documentation. https://en.bitcoin.it/wiki/Protocol_documentation#Hashes. Accessed: 2017-11-10.

[11] Radix tree. https://en.wikipedia.org/wiki/Radix_tree. Accessed: 2018-01-03.

[12] Reflux-tx. https://github.com/ConsenSys/reflux-tx. Accessed: 2017-12-19.

[13] Sha-2. https://en.wikipedia.org/wiki/SHA-2. Accessed: 2017-12-20.

[14] Sha-3. https://en.wikipedia.org/wiki/SHA-3. Accessed: 2017-12-20.

[15] Ethereum energy consumption index. https://digiconomist.net/ethereum-energy-consumption, 2017. Accessed: 2017-12-19.

[16] Prableen Bajpai. Bitcoin vs ethereum: Driven by different purposes. https://www.investopedia.com/articles/investing/031416/bitcoin-vs-ethereum-driven-different-purposes.asp, November 2017. Accessed: 2018-01-10.

[17] Bitcoinmining.com. Bitcoin mining hardware guide. https://www.bitcoinmining.com/bitcoin-mining-hardware/. Accessed: 2017-11-10.

[18] Blockgeeks. What is ethereum? a step-by-step beginners guide. https://blockgeeks.com/guides/what-is-ethereum/. Accessed: 2017-11-22.

[19] Boxmining. Ethereum proof-of-stake. http://boxmining.com/ethereum-proof-of-stake/. Accessed: 2017-11-10.

[20] Vitalik Buterin. A next generation smart contract and decentralized application platform, 2013.

[21] Vitalik Buterin. Merkling in ethereum. https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/, November 2015.

[22] Vitalik Buterin. Ethereum 2.0 mauve paper. https://cdn.hackaday.io/files/10879465447136/Mauve%20Paper%20Vitalik.pdf, 2016.

[23] Chris Castiglione. Bitcoin vs. ethereum. , December 2017.

[24] CodeTract. Inside an ethereum transaction. https://medium.com/@codetractio/inside-an-ethereum-transaction-fa94ffca912f, February 2017. Accessed: 2017-12-13.

[25] Michael Crosby, Nachiappan, Pradhan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. Blockchain technology, 2015.

[26] Jeff Desjardins. How bitcoin, ethereum, and the other major cryptocurrencies compare to one another. http://www.businessinsider.com/bitcoin-price-etherum-and-other-cryptocurrencies-compare-2017-9?international=true&r=US&IR=T, September 2017.

[27] Glenn Fowler, Landon Noll, Kiem-Phong Vo, Donald Eastlake, and Tony Hansen. The fnv non-cryptographic hash algorithm. Internet-Draft draft-eastlake-fnv-14, IETF Secretariat, December 2017. http://www.ietf.org/internet-drafts/draft-eastlake-fnv-14.txt.

[28] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1), 2001.

[29] Noah Korevaar. 9 dingen die je moet weten over ethereum vs bitcoin: beste investering? `https://www.want.nl/ethereum-eth-ether-bitcoin-cryptocoin/`, December 2017.

[30] Cristopher Malmo. One bitcoin transaction now uses as much energy as your house in a week. `https://motherboard.vice.com/en_us/article/ywbbpm/bitcoin-mining-electricity-consumption-ethereum-energy-climate-change`, November 2017.

[31] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf.

[32] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, Princeton, NJ, USA, 2016.

[33] Karl J O'Dwyer and David Malone. Bitcoin mining and its energy footprint. 2014.

[34] Bart Preneel. A perspective on cryptocurrencies, September 2017. Slides.

[35] Certicom Research. Standards for efficient cryptography, SEC 2: Recommended elliptic curve domain parameters, 2010. Version 2.0.

[36] Ethereum team. Byzantium update announcement. `https://blog.ethereum.org/2017/10/12/byzantium-hf-announcement/`, October 2017.

[37] Søren Steffen Thomsen and Lars Ramkilde Knudsen. *Cryptographic hash functions*. PhD thesis, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Applied Mathematics and Computer ScienceInstitut for Matematik og Computer Science, 2005.

[38] Bitcoin wiki. Elliptic curve digital signature algorithm. `https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm`. Accessed: 2017-11-10.

[39] Bitcoin wiki. Proof-of-work. `https://en.bitcoin.it/wiki/Proof_of_work`. Accessed: 2017-11-10.

[40] Daniel Wilson-Nunn and Hector Zenil. On the complexity and behaviour of cryptocurrencies compared to other markets. *arXiv preprint arXiv:1411.1924*, 2014.

[41] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[42] Work2heat. Understanding the ethereum trie. `https://easythereentropy.wordpress.com/2014/06/04/understanding-the-ethereum-trie/`, June 2014.

[43] Vlad Zamfir. Introducing casper "the friendly ghost". `https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/`, 2015.