NTNU
Norwegian University of
Science and Technology

# Long-term Bitcoin Scalability

**Erlend Solberg Thorsrud**

|  |  |
|---|---|
| Master of Science in Communication Technology | |
| Submission date: | June 2018 |
| Supervisor: | Colin Alexander Boyd, IIK |
| Co-supervisor: | Chris Carr, IIK |

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

|  |  |
|---|---|
| **Title:** | Long-term Bitcoin Scalability |
| **Student:** | Erlend Solberg Thorsrud |

**Problem description:**

Since its launch as the first modern cryptocurrency in 2009, Bitcoin has increased enormously in the number of users, transactions and market capitalization. Bitcoin is the biggest cryptocurrency and has dominated the market of cryptocurrencies for a long time. During the peak in transactions in December 2017, the challenges with Bitcoin revealed themselves. The scalability problems Bitcoin is facing became reality, and the users experienced how long a transaction could take to get published onto the blockchain, and the increased transaction fee as a consequence of the increased number of transactions. These among other scalability problems will be analyzed in this thesis. Statistical data from the live blockchain will be used if applicable. In order to understand the scalability problems, there will be a critical evaluation of the Bitcoin system.

There are some suggested solutions to the scalability problems Bitcoin is facing being developed. Accordingly, the thesis will give a explore various possible solutions and give a more in-depth analysis of the lightning network as a proposed solution. Historical data will be used to explore if one can make a future prediction about the system. Furthermore, the thesis will consider future scenarios based on assumptions considering the lightning network.

|  |  |
|---|---|
| **Responsible professor:** | Colin Alexander Boyd, IIK |
| **Supervisor:** | Chris Carr, IIK |

# Abstract

The interest has grown around Bitcoin as a trustless and decentralized digital payment service. Along with the increase in users and transactions, Bitcoin suffers from challenges regarding scalability. Many solutions have been proposed over the years, but Bitcoin is still struggling with finding the solution that suits everyone. *Lightning Network* as one of the most promising solutions, will be in the scope of this thesis, along with why such a solution is needed.

By investigating Bitcoin, scalability challenges and proposed solutions, we evaluated Bitcoin's future. We explore challenges such as propagation time, block size and energy consumption. Increasing the block size, changing the consensus algorithm and off-chain payments are discussed. The Lightning Network as an off-chain solution is investigated further in-depth. Challenges and consequences of implementing the Lightning Network are discussed. Furthermore, this thesis discusses whether the Lightning Network is the solution Bitcoin needs or not.

# Sammendrag

Interessen har vokst rundt Bitcoin som en tillitsminimalt og desentralisert digital betalingsløsning. Med økt antall transaksjoner og brukere, opplever Bitcoin utfordringer med tanke på skalering. Mange løsninger har blitt foreslått gjennom årene som har vært, men Bitcoin har ennå ikke funnet en løsning som passer alle. *Lightning Network* er en av de bedre løsningene som blir utviklet i dag. Denne oppgaven vil undersøke Lightning Network og hvorfor en slik løsning trengs.

Ved å undersøke Bitcoin, skalerings problemene og løsninger vil denne oppgaven evaluere Bitcoins fremtid. Utfordringer som forplantning av blokker, størrelsen av en blokk og bruk av energi vil bli undersøkt. Løsninger som økelse av blokkstørrelsen, endring av konsensus algoritmen og utenom-kjeden betalinger vil bli diskutert. Lightning Network som en utenom-kjeden løsning vil bli undersøkt i en dypere sammenheng der utfordringer og konsekvenser av implementasjon vil bli diskutert. Deretter vil denne oppgaven reflektere rundt Lightning Network, og om det er en god løsning for Bitcoin.

# Preface

This thesis is submitted at the Department of Information Security and Communication Technology at Norwegian University of Science and Technology (NTNU). The thesis constitutes the final project for the MSc program in Communication Technology with specialization in Information Security. The duration of the study has been 20 weeks, performed in the Spring semester of 2018.

I would like to thank my supervisor Chris Carr and responsible professor Colin Boyd for nice remarks and feedback.

Trondheim, 11th of June 2018. Erlend Solberg Thorsrud

# Contents

**List of Figures** **ix**

**List of Tables** **xi**

**1 Introduction** **1**

1.1 Motivation . . . 1

1.2 Research objectives . . . 2

1.3 Outline . . . 3

**2 Bitcoin** **5**

2.1 Hash . . . 6

2.2 Digital Signatures . . . 7

2.3 Transaction . . . 8

2.3.1 Contracts And Time Locks . . . 9

2.3.2 Multisignature transactions . . . 11

2.3.3 Transaction malleability . . . 13

2.4 Blockchain . . . 14

2.4.1 Blocks . . . 14

2.4.2 Proof-of-Work . . . 16

2.4.3 Fork . . . 17

2.5 Network . . . 18

**3 Scalability Problems and Possible Solutions** **19**

3.1 Block size . . . 19

3.2 Block propagation . . . 22

3.3 Energy Consumption . . . 22

3.3.1 Proof-of-Stake . . . 25

3.3.2 Decentralization . . . 28

3.4 Off the blockchain . . . 29

**4 Lightning Network** **33**

4.1 Requirements . . . 34

4.2 Transactions . . . 35

4.3 Hashed Time-Locked Contract . . . . . . . . . . . . . . . . . . . . . . 38
4.4 Evolution of the Lightning Network . . . . . . . . . . . . . . . . . . . . . 40
4.4.1 Channel opening . . . . . . . . . . . . . . . . . . . . . . . . . 40
4.4.2 Channel closing . . . . . . . . . . . . . . . . . . . . . . . . . . 42
4.4.3 Normal operation . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.4.4 Transactions and HTLC . . . . . . . . . . . . . . . . . . . . . 45
4.5 Challenges . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
4.6 Network Topology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
4.6.1 Flare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
4.6.2 Onion Routing . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
4.7 Collusion Attack . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

**5 Discussion** **53**
5.1 Bitcoin survival . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
5.2 Lightning Network evaluation . . . . . . . . . . . . . . . . . . . . . . . 56
5.3 Scenario 1 - When all bitcoins are mined . . . . . . . . . . . . . . . . . 58
5.4 Scenario 2 - The operational Lighting Network . . . . . . . . . . . . . 59
5.5 Centralization vs Decentralization . . . . . . . . . . . . . . . . . . . . . 61
5.6 GDPR . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62

**6 Conclusion** **65**
6.1 Limitations and future research . . . . . . . . . . . . . . . . . . . . . . 66

**References** **67**

# List of Figures

2.1 Bitcoin transactions . . . 9
2.2 2-of-2 multisignature . . . 11
2.3 Basic micropayment . . . 12
2.4 Blockchain . . . 14
2.5 Bitcoin block . . . 15
2.6 Average number of transaction in a block . . . 16
2.7 Fork . . . 17

3.1 Block propagation . . . 23
3.2 Hash rate distribution . . . 28
3.3 Invalidation tree . . . 30
3.4 Dupelx channel . . . 31
3.5 Bidirectional channel . . . 31

4.1 Lightning Network channel . . . 37
4.2 HTLC . . . 38
4.3 Fully revoked commitment transaction . . . 41
4.4 Channel establishment . . . 42
4.5 Closing channel . . . 43
4.6 Normal operations . . . 44
4.7 Lightning Network transactions . . . 45
4.8 Lightning Network test network . . . 47
4.9 Organic routing . . . 48
4.10 Global beacons . . . 49
4.11 Local beacons . . . 50
4.12 Collusion attack . . . 52

5.1 Fee and unconfirmed transactions graph . . . 54
5.2 Correlation graph . . . 55
5.3 Bitcoin ecosystem . . . 56

# List of Tables

3.1 The evolution of mining . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

# Chapter 1
# Introduction

In this chapter, we will explain the motivation behind thesis. We state our goal and research question based on the motivation. There will be an outline of how we are planning to answer the research question and reach our goal.

Originally, Satoshi Nakamoto designed Bitcoin as an answer to the unstable banking system after the collapse in 2008. When converting the hex values in the coinbase value of the first block generated, the sentence “The Times 03/jan/2009 Chancellor on brink of second bailout for banks” appears. Bitcoin was and is aiming to replace the banking system in a trustless manner. However, to accomplish consensus on a trustless peer-to-peer network is a task involving serious knowledge about cryptography and networking. Satoshi Nakamoto’s identity is still unknown, and the Bitcoin community has taken over the development of the largest cryptocurrency. Bitcoin is an open source development project, for everyone to read and contribute to the source code.

## 1.1 Motivation

Bitcoin experienced a tremendous increase in the price during 2017. Notably, the interest of Bitcoin grew as the price continued to grow throughout the year. During the peak of 2017, in December, users experienced long confirmation time for publishing a transaction additionally to significant transaction fees. The capacity of Bitcoin was reached. A congested network resulted in high transaction fees and long confirmation time for a single transaction. This research was motivated to find out why this was happening and if there were some solutions to the problems users of Bitcoin was experiencing. The information about the Lightning Network was found mainly in the white paper and the Request for Comments (RFC). Understanding the essentials of Bitcoin is necessary to understand the challenges Bitcoin is facing, along with the suggested solutions. Why are people willing to use a slow and expensive service like Bitcoin?

In the beginning of Bitcoin, Satoshi Nakamoto was the leading figure. When he vanished, there was a lack of leadership in the Bitcoin community. Whether this is a perk or a downside can be discussed. However, radical changes must be discussed and decided upon by the community, resulting in Bitcoin Improvement Proposal (BIP). A BIP is a proposal reviewed by the Bitcoin developers. It is their decision whether it should be implemented or not. In the case of a dispute, the conservative option will be preferred. The lack of governance leads to a slow process where reaching consensus among the developers could be difficult. As a result of this process, serious problems will only be discussed, instead of discussing a specific solution to the problem. The Lightning Network is exciting, as it is one of the few comprehensive solutions being currently worked on. The Lightning Network aims to solve the scalability problem along with reducing the high transaction fee experienced in December 2017.

## 1.2 Research objectives

Based on the motivation, a goal for this thesis is defined as **G1: Evaluate the long-term scalability of Bitcoin.** By setting this goal, the thesis will provide an analysis of the problems in Bitcoin and the respective solutions. Accordingly, the main research question addressing **G1** will be:

**RQ1: Can Bitcoin survive in its original form?**

The way this will be approached is to divide the main research question (**RQ1**) into smaller and more specific sub-questions (**SQ**)s. First of all the thesis will look at Bitcoin in its original form in Chapter 2, answering the following sub question:

**SQ1: What is the original form of Bitcoin?**

After exploring the original form of Bitcoin, the problems of Bitcoin will be identified. This thesis focuses primarily on the scalability and technical issues of Bitcoin, resulting in the following sub-questions:

**SQ2: What are the scalability problems of Bitcoin?**

Chapter 3 will answer this question. After identifying the bottlenecks of scalability, such as block size, energy consumption, and block propagation, the thesis will discuss solutions that can be implemented in the Bitcoin protocol. However, making radical changes in the Bitcoin protocol is complicated as mentioned earlier in this Introduction.

Therefore this thesis will explore external solutions building a second layer above Bitcoin. Two solutions will be explored, before diving into one of the solutions, the Lightning Network, leading to the last sub-questions:

> **SQ3: What is the Lightning Network, and how does it address the scalability problems?**

Chapter 4 explains the Lightning Network in detail and how it addresses the scalability problems. Challenges will be discussed. After acquiring knowledge about the Lightning Network, this thesis will discuss the consequences of using the Lightning Network on Bitcoin. To enrich the evaluation, futuristic scenarios will be described based on assumptions.

An analysis of how the blockchain reacted under massive stress in December 2017 is performed. The results will be used to discuss the scalability problems of Bitcoin and the respective solutions. From the discussion, some trade-offs will be extracted. The Lightning Network will be explored in detail, and the knowledge found is used to evaluate the impact on Bitcoin. The results from the findings will be used to evaluate the survival of Bitcoin.

## 1.3 Outline

In order to understand the challenges of Bitcoin, it is important to understand the system itself. However, to cover every detail about Bitcoin will be too large of a scope. Accordingly, mostly the components and fundamentals of Bitcoin-related scalability problems is mentioned.

When knowledge about Bitcoin is satisfied, scalability problems will be explored and discussed leading to a specific solution, the Lightning Network. Mainly, the Lightning Network will be explored in depth to evaluate Bitcoin's future.

Lastly, the thesis will summarize the results and knowledge found, giving scenarios about the future based on acquired knowledge.

**Chapter 2: Bitcoin.** A critical analysis of the Bitcoin system. Including the essential pieces of the Bitcoin system to understand the causes of the scalability problems. This chapter will give a description of Bitcoin in its original form.

**Chapter 3: Scalability and Solutions.** Exploration of the scalability problems Bitcoin is facing, along with some suggested solutions.

**Chapter 4: Lightning Network.** An in-depth description of the Lightning Network and its challenges.

**Chapter 5: Discussion.** Summarize and an analysis of the results found in Chapter 3 and Chapter 4.

**Chapter 6: Conclusion.** The last chapter contains a conclusion that answers to the research objectives given in this introduction chapter. Future research will also be proposed.

# Chapter 2 Bitcoin

Bitcoin is the most successful cryptocurrency ever made. However, what is Bitcoin? In this chapter, the building blocks of Bitcoin and how they fit together is explained. The goal of this chapter is to describe Bitcoin in its original form.

Bitcoin is a protocol that describes a public ledger, where every transaction is broadcast to every node in the network. A node is a computer connected to the Bitcoin network. The ledger keeps track of who sent what to who. When Alice wants to send 5 Bitcoins as a currency, also bitcoins (BTC) to Bob, she announces to the network that she sent 5 bitcoins to Bob. Bitcoin will be used as terminology for the system itself, while bitcoins or BTC will be used as the currency, similar to dollar and USD. Now everyone knows that the transaction took place, and it is written down in the ledger. How can we be sure that Alice completed the transaction? A transaction is only broadcast if she puts a receiving address in the transaction output, and thereby executed. We will explain this further in Section 2.3.

Now, because Bitcoin is working over a worldwide network, there is no universal time measure. So how can the network know at what time Alice broadcast that she spent her bitcoins? The Bitcoin *blockchain* keeps the order of transactions, see Section 2.4. Thus the blockchain functions as a timeline. The blockchain consists of blocks, explained in Subsection 2.4.1, where a block contains data about, e.g., transactions. The blocks are linked together by a hashing function, by including the hash of the previous block. Similarly, *hash chain* is chain where hash values are used to create a link between the elements. Haber and Stornetta [HS91] proposed using a hash chain for time-stamping digital documents [HS91].

The timeline, blockchain, is secured by an algorithm called *proof-of-work*, which ensures consensus on the blockchain among the nodes. In order to add a transaction on the timeline, it must be added to a block, and a computational puzzle needs to be solved, which is a part of the proof-of-work algorithm. This concept was explored by Dwork and Naor [DN92] in 1992 to combat junk mail. When a node solves the computational puzzle, the transaction gets published onto the timeline. Now the transaction is confirmed and can be seen on the timeline of transactions. This explanation is a simplified explanation of Bitcoin.

Instead of a single transaction getting published on the timeline, the blockchain is composed of blocks. One block contains numerous transactions, taken from a pool of unconfirmed transactions. For a node to publish one block, it has to solve the puzzle, because of the proof-of-work algorithm. When the puzzle is solved, the block gets broadcast to the network and appended on the blockchain. Now the transaction will be confirmed. The process where the nodes are finding new blocks by solving the computational puzzle is called *mining*.

The explanation above is still a brief explanation of Bitcoin, now let us dive into the technical aspect of Bitcoin. To understand Bitcoin, understanding the cryptographic building blocks that make the blockchain is necessary.

## 2.1 Hash

A hash function is a function that takes a value of variable length in and produces a hash value of fixed length, $h = \mathrm{H}(m)$. The output should be pseudorandom so that the string will be unreadable for humans. The Secure Hash Algorithm (SHA) family of cryptographic hash functions differs from regular hash functions because of certain characteristics.

- It is computationally efficient, easy to compute.
- **Preimage**[1] **resistance** provided. This means it should be computationally infeasible to find a given hash value $h$ by running $y$ through the hash function.
- **Second preimage resistant** requirement met. Given a message $x$ with hash value $h$, it is infeasible to find a alternative message $y$ that if put into the hash function gives the same hash value $h$, so that $\mathrm{H}(y) = \mathrm{H}(x)$.
- **Collision resistant**. This means it is impossible or infeasible to find a pair of messages *(x,y)* that gives the same hash value h, $\mathrm{H}(x) = \mathrm{H}(y)$.
- The output of cryptographic hash function H is **pseudorandom**.

Bitcoin uses SHA-256 as the underlying cryptographic hash function for most of the components where a hashing algorithm is required. RIPEMD160 is also used as a hashing function when verifying the public key, as described in Subsection 2.3.1. SHA-256 gives an output of 256-bits length and is most commonly used.

[1] A preimage is the value you put into the hash function, the $x$ in $h = \mathrm{H}(x)$.

When hashing a hash value, a hash chain is created. If $x$ represents a block, Equation 2.1 shows that if a value is changed, so that $x$ is different, then the total value will change. This is a consequence of the last pseudorandom property of a hashing algorithm as explained above. Chaining the block gives the property that changing a transaction in a block will change the hash value of all the block pointing to that block.

$$h(h(h(h(h(x))))) = h^5(x) \tag{2.1}$$

## 2.2 Digital Signatures

Consider the scenario where Alice wants to create a transaction for sending bitcoins to Bob. How can Bob be sure that it is Alice who sends him the funds? How can the rest of the network verify this transaction? There must be countermeasures against the following possibilities to make a secure solution [Sta14]:

- Bob forging a transaction and claiming it came from Alice using the authentication code shared between Alice and Bob.
- Alice denying the transaction took place since it is possible for Bob to forge the transaction.

The solution is digital signatures, and these are the challenges digital signatures must solve.

The described scenarios set some requirements for a digital signature. It is crucial that the signature contains some information about the sender. The information should be sufficient to prevent forgery and denial as mentioned above. A digital signature should also be easy to produce and verify. However, it should be infeasible for attackers to forge a digital signature. An attacker can forge a digital signature by either changing the message of an existing signature or forging the digital signature of a given message.

Normally a digital signature is composed of three components:

- Key generation function
- Signature function
- Verification function

Both ends of the transaction should generate a key pair, consisting of a public key and a private key. The public key should be known to everyone. Simplified, Alice must know the public key of Bob to send him bitcoins. To prove that it is, in fact, Alice who sent the bitcoins, she must use her private key in a signature function to produce a digital signature, which will be sent with the transaction. Bob can then use the public key of Alice to verify the signature. The verification function does this, and if it accepts the signature, Bob can indeed be sure that it was Alice who sent the bitcoins. Bitcoin uses Elliptic Curve Digital Signature Algorithm (ECDSA) with a *secp256k1* curve for creating the signatures. *secp256k1* is not a National Institute of Standards and Technology (NIST) standard.

To make a Bitcoin address, a key-pair must be generated. After creating, the key pair, the public key is used. A flag compression byte is added to the key, then the key is hashed, both with SHA-256 and RIPEMD160. After the key is hashed, a network version byte is added, and then the checksum is appended. Resulting in a hash value called Bitcoin address. The Bitcoin address can be compared to a public key certificate. Both are used for proving ownership of a public key in a way. While the public key certificate is issued by a Certificate Authority (CA), no central authority issues Bitcoin addresses. Everyone is free to make a Bitcoin address.

## 2.3 Transaction

When a Bitcoin transaction is broadcast, it arrives in a pool of unconfirmed transactions at the miner nodes. Subsequently, the transaction is added to a block that a miner will work on to solve the puzzle. If the miner succeeds, the block is then broadcast to the network. The transaction is now confirmed. The *txid* is the identifier of a signed Bitcoin transaction, which is a hash value of the serialized representation [DW14].

There are three components in a Bitcoin transaction, see Figure 2.1 for a simplified version. Transaction metadata, input(s) and output(s). There can be several inputs and outputs in one transaction. The transaction metadata is containing several fields, including the size of the whole transaction and the transaction identification. As mentioned there can be several inputs and outputs, so in the metadata, there are two fields that each contain the number of inputs and outputs. Along the mentioned fields, there is also a field called *nLockTime* which set a time for when the transaction will be published. See Subsection 2.3.1 for time locks. All the fields can be seen in Figure 2.5.

Each input contains a pointer to the hash value of a previous transaction, which links the transactions together. Along with the hash pointer to the previous transaction, the input also contains a value that specifies the index of the output in the previous transaction. All inputs contains has a standard claiming script, called *scriptSig* [DW14]. Note that from BIP141 [LJW15], this is called the *witness stack*.

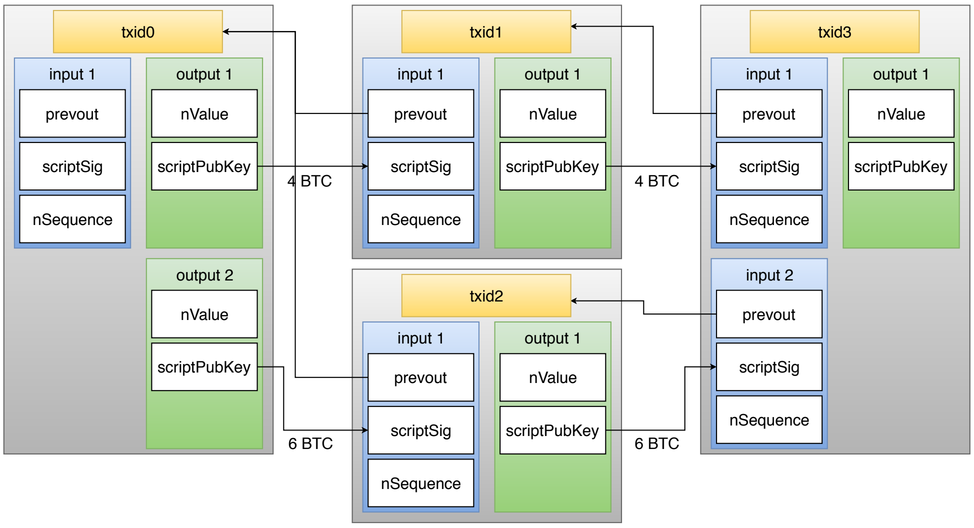

**Figure 2.1:** Simplified figure showing transactions with several inputs and outputs. Txid is the hash value of all the data, included *nVersion* and *nLockTime* which is not included in this figure. For txid3, output 1 is worth 10 BTC.

This script verifies that the spender is the owner of the bitcoins, and uses digital signatures to do so. A *coinbase transaction* is the transaction creating a new coin. This transaction does not have a hash pointer to a previous transaction, but there is an output which the miner is free to modify to his own or others address. Every miner includes a coinbase transaction in the block they are currently working on. This transaction can be to anyone, but usually, the miner sends this transaction to one of his addresses. The coinbase transaction is how new coins get minted in Bitcoin.

There is two subfields in each output, *nValue* and standard claiming condition, usually a script called *scriptPubkey*. The *nValue* specifies the value of the output in *satoshi*. One bitcoin is $10^8$ satoshi. The second subfield is the recipient address, the receiver's Bitcoin address along with a script and some commands. Outputs that can be spent by an input of another transaction are called Unspent Transaction Output (UTXO). Bitcoin wallets show a balance, but this balance is in fact just UTXOs, transactions that are waiting to be spent.

### 2.3.1 Contracts And Time Locks

As mentioned the inputs and outputs can contain various scripts. A *contract* is usually enforced by scripts in both input(s) and output(s). Pay-To-PublicKey-Hash

(P2PKH) is a contract enforced by *scriptPubKey* in the output and *scriptSig* in the input. Bitcoin uses a language close to the Forth stack [BBSU12]. For example, when Bitcoin is checking a signature, it adds the data onto the stack, before it uses some operations to check if the signature is valid, *CHECKSIG*.

```
DUP HASH160 0afabk18ddf EQUALVERIFY CHECKSIG
```

In the example above, Alice wants to make a simple standard P2PKH transaction to Bob. In the output of the transaction, Alice put this script, where *DUP* is duplicating the hashed value of the hash value of Bob's public key, which is already on the stack. *HASH160* hashes the top element of the stack, which is the public key of Bob. Note that in Bitcoin, a public key is 160-bit long hash value in the third field, but for this example, it is simplified. Then *EQUALVERIFY* is checking if the hash value of *HASH160* is the same as the hash value of the public key hash of Bob. *CHECKSIG* is verifying an ECDSA signature compared to the public key and leaves true if it is valid, otherwise false.

### Time locks

Time locks are a feature that forces a transaction to wait in the pool of unconfirmed transactions, by adding a time-based constraint. A transaction is invalid when this time constraint is active. If a transaction with an active time lock is included in a block, the transaction is invalid and therefore also the block. Thus, miners will avoid including transactions with activated time locks in their blocks. There are two types of time locks as of now in Bitcoin, *absolute time lock* and *relative time lock* according to McCorry et al. [MMSH16].

An *absolute time lock* is a time lock defined in the header by the field *nLockTime*. An *absolute time lock* is a time lock preventing a transaction from getting published onto the blockchain for a given period of time. The *nLockTime* defines how long a transaction must wait to be included in the blockchain. More specifically, it defines either the height of blocks or the amount of time the transaction must wait before getting accepted onto the blockchain. The *absolute time lock* is defined as OP_CLTV in the stack language of Bitcoin.

A *relative time lock* is the second type of time lock. The *nSequence* variable defines the *relative time lock*. Based on the confirmation of the parent transaction to the blockchain, the time lock is activated. BIP68 [FBDk15] and BIP112 [FBL15] is the proposals of this functionality. *nSequence* often defines a block height or some seconds with a unit of 512 for the transaction to wait. However, the *nSequence* field could also be a regular sequence number and not a time lock. The *relative time lock* is defined as OP_CSV in the stack language of Bitcoin.

An example: At block 1, Alice sets a *nSequence* to be 40 blocks for tx2. The parent transaction, tx1, is not included before in block 10. Since the *nSequence* is 40 for tx2, it will first be included at block 50. While, if Alice used an *absolute time lock* by defining *nLockTime* to be 40 blocks, her transaction is included at block 41.

### 2.3.2 Multisignature transactions

A multisignature is when several people can sign the same message [OO91], where in Bitcoin the message is a transaction. Two or more signatures, depending on the type of multisignature, is needed to spend the from the output of the multisignature transaction. In other words, two or more signatures are required in the input of the next transaction by the claiming script.

There are different variants of a multisignature, but for our scope, the 2-of-2 multisignature is the most relevant. A 2-of-2 multisignature means that both signatures of the public keys are needed to spend form the output. Note that there can be one to several inputs in a multisignature transaction. Consequently, both signatures are needed to publish a transaction spending from the 2-of-2 multisignature transaction output(s). If only one signature is provided, the transaction spending from the multisignature transaction is invalid.

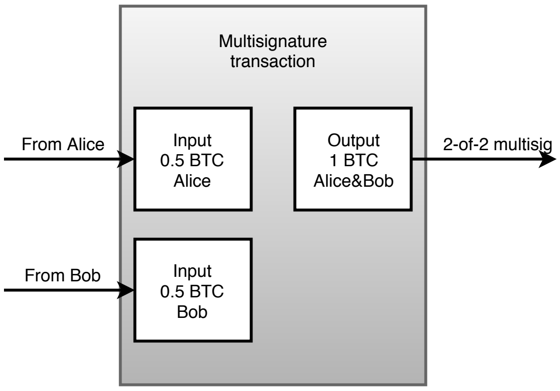

**Figure 2.2:** 2-of-2 multisignature. Alice and Bob puts 0.5 BTC each. In order to spend, both signatures are needed.

Imagine Alice is visiting a bar, and she wants to open a running tab. Instead of paying each beer with a Bitcoin transaction, she can update the balance, and close the tab with one single Bitcoin transaction when she leaves the bar. Accordingly, reducing the number of transactions can be done with a multisignature transaction, or for simplicity, called a multisignature address. Alice plans to spend at maximum 1 BTC this evening, and one beer is costing 0.2 BTC. Alice sends 1 BTC to the

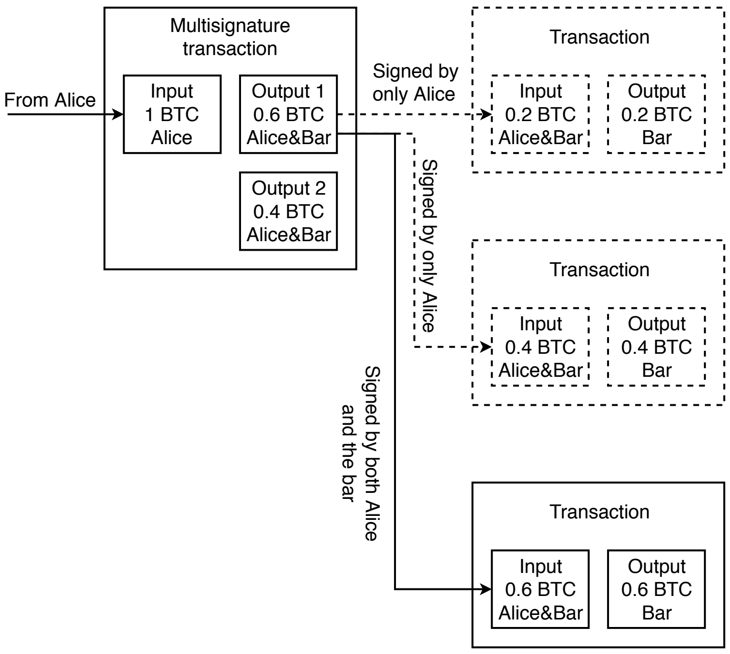

**Figure 2.3:** Multisignature transaction can enable off-chain payments. Only the 0.6 transaction is published. Transactions where Alice are refunded the change is not included. Only the last output of the multisignature transaction is shown.

multisignature address. For each time she is buying a beer, she will add 0.2 BTC to the first transaction made from this address.

In the first transaction, Alice sends 0.2 BTC from the multisignature address with her signature to the bar. Note that she will also send the change, 0.8 BTC back to her Bitcoin address in a separate transaction. The next transaction will be a value of 0.4 BTC from the multisignature address and 0.6 BTC back to her, signed by Alice. As mentioned, one of two signatures is not a valid transaction, because it is a 2-of-2 multisignature address. Both signatures are required to make both the transactions, to the bar and the change back to Alice, valid. Therefore, when Alice leaves the bar, the bar wants to sign the most recent transactions made from this address, because the one going to the bar contains most bitcoins. When the bar signs the most recent transactions, they will become valid, since Alice already signed them. See Figure 2.3, for the scenario. The bar will also need to sign the change transaction back to Alice, if there is some. What if the bar does not sign the change transaction? A time lock can be put on the multisignature address to avoid funds being locked up by an uncooperative party, see Subsection 2.3.1 for time locks.

### 2.3.3 Transaction malleability

Double-spending is one of the core problems which Bitcoin solves. Commonly, double-spend is associated with an attacker publishing a transaction, then try to spend from the same output to invalidate the first transaction he broadcast. An example is if Alice pays Bob with tx1 and broadcast it. In the same time, she makes another transaction back to herself, tx2, which is spending from the same output as tx1. By the consensus rules, only one transaction can spend from this output, so one of them will be invalidated. Now, Alice is successful if tx2 reaches more miners than tx1, so that tx2 is included in a block, and therefore tx1 is invalidated.

A variant of double-spend is exploiting transaction malleability [DW14]. Multisignature transactions and normal transactions can suffer from *transaction malleability*. As mentioned in Section 2.3, a transaction is identified by a hash value. Wuille [Wui14] described nine possible way for an attacker to change the transaction identification without invalidating the transaction in BIP62 [Wui14].

The attacker is the recipient of a transaction, and the victim is the sender. When the attacker gets the broadcast transaction from the victim, he will modify the transaction so that the identification is changed, but without invalidating the transaction. Subsequently, the attacker will broadcast the modified transaction, which still pays the attacker from the victim. According to the consensus rules, only one of the transaction will be confirmed. The attacker is successful if the modified transaction is confirmed [DW14]. The sender can be a victim if he uses transaction identification to update the balance. By using transaction identification to track the transaction, the sender will see that the transaction was not confirmed. He might issue a new transaction. However, if the sender used a UTXO list, he will see that the output that the unconfirmed transaction spent from, is now spent. This is because the modified transaction is still spending from the same output.

Segregated Witness (SegWit) is some functionality added to Bitcoin to create a separate data structure called *witness* [LJW15]. By moving the data used for validating a transaction, such as signatures, the sources of malleability listed by Wuille [Wui14], are prevented. The validation data is moved to the witness structure. The witness structure is a tree which is nested into the Merkle root via the coinbase transactions [LJW15].

The SegWit protocol allows an increase of the block size. However, the increase is possible in a soft fork because the term block size is redefined to block weight, see Equation 2.2. The *Base size* is the original block size, without the witness data. The *Total size* is the block size when including the base data and the witness data [LJW15]. The block weight is defined as maximum 4 MB, so in addition to a standard block of 1 MB, there can be additionally 3 MB of witness data.

$$\text{Block weight} = \text{Base size} \cdot 3 + \text{Total size} \tag{2.2}$$

A hard fork was not necessary because SegWit can adjust the block size for old nodes [LJW15] See Subsection 2.4.3 for forks. In addition to *nVersion*, *txins*, *txouts*, *nLockTime*, a *marker*, *flag* and *witness* field is added in a transaction [LJW15]. There will be two transaction identifications, one for old nodes, and one for new nodes, *txid* and *wtxid* respectively. Data will be added or stripped depending on the software the receiving node is running. A signature is taking considerable amount of space in the transaction[2]. By moving the signature to the new data structure, a witness transaction can be significantly smaller. If a block only contains witness transactions, the number of transaction per block can be increased.

## 2.4 Blockchain

The blockchain is the essential technology behind Bitcoin. Blocks form the blockchain. Blocks are consisting of transactions, and is linked together by a hash pointer, see Figure 2.4. The hash pointer is a pointer to the previous block, and also the hash value of that block. By hashing the previous block, one can achieve some security regarding the integrity of that block. If some values in the previous block changed, all the calculated hash value after it would also change. Recall Section 2.1 that by including the previous block in the hash, a chain is made. When modifying one of the blocks in this chain, the hash value will change for all of the descendant blocks.

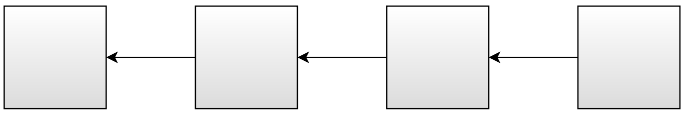

**Figure 2.4:** Block linked together by a hash pointer creating the a chain of blocks.

### 2.4.1 Blocks

Blocks are the foundation of the blockchain. A block consists of two things, the block header and the list of transactions in the form of a Merkle tree [Mer87], see Figure 2.5. The header contains, for instance, data about the computational puzzle the miner solved. This data enables the rest of the network to verify the block. Among the data, there is a nonce, that the miner changes to solve the puzzle, a time stamp and some bits that indicate how difficult it was to solve the puzzle. By

[2]https://www.youtube.com/watch?v=DzBAG2Jp4bg accessed 06.06.2018

defining the difficulty of each block, Bitcoin can adjust the difficulty of the next puzzle. Consequently, the average block creation is every 10th minute. Adjusting of the difficulty takes place every 2016 blocks. A *genesis block* is the first block created in a blockchain.

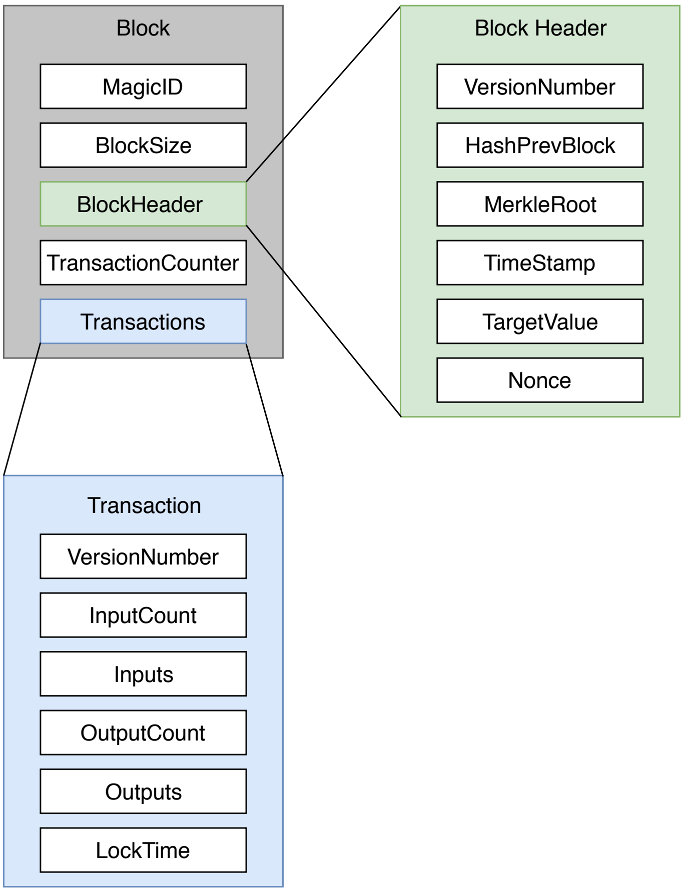

**Figure 2.5:** The components of a block.

Block size is a debated topic in the Bitcoin community. In the original Bitcoin system, the block size is 1 MB. The block size will be discussed in Section 3.1. One transaction is at least 250 bytes [NBF+16], which means a block can maximum contain 4000 transactions. $1 \cdot 10^6/250 = 4000$. Note that there is no restriction for the minimum transaction size, but for having one input and two outputs, 250 bytes is a minimum. Figure 2.6 shows that during December 2017, the average number of transactions included in a block was 2225 transactions. Based on the estimate in Section 3.1, the average transaction size during December is approximately 480 bytes. Increasing the block size will allow inclusion of more transactions in a block, and therefore increase the throughput.

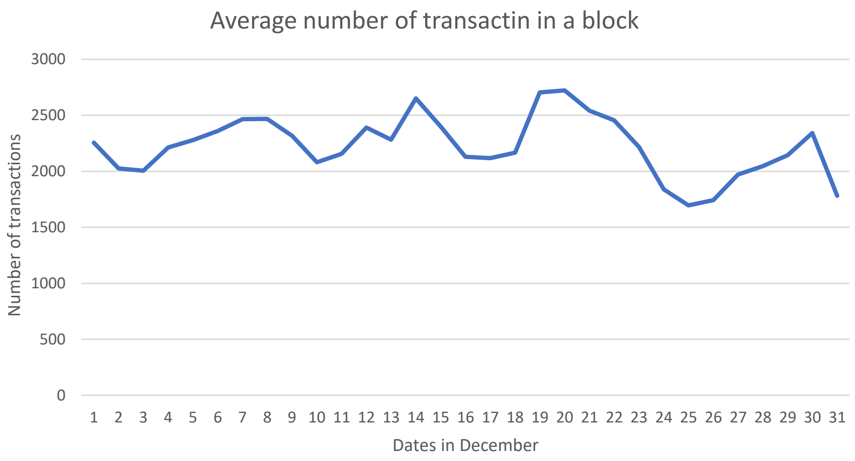

**Figure 2.6:** Chart describing the average transactions in a block during December 2017. Data taken from blockchain.info[3].

### 2.4.2 Proof-of-Work

Proof-of-Work (PoW) is the algorithm used in Bitcoin to achieve consensus throughout the network. PoW was introduced by Dwork and Naor [DN92]. By solving a computational puzzle, the entity shows that it went through an effort to get the service. By restricting the right to broadcast by solving a puzzle, flooding and DoS-attacks will be avoided to a certain degree. Solving the puzzle should require significant computational power. In Section 2.1 we defined that a given hash value should be hard to find, but easy to verify. Twice, the miner runs a string of values through a SHA256 function to find the solution to the puzzle. $H(B) = \text{SHA256}(\text{SHA256}(B))$.

If the hash value the miner produces is in a targeted space, the puzzle is solved. The miner will broadcast the block. If not, the miner changes the nonce in the string of values and runs it through the hash function again. Repeatedly, the process is performed until the miner gets a hash value within the targeted space. If the miner runs through all possible nonces, he can replace a transaction and start over with the new block.

Initially, PoW was working as a lottery. One Central Processing Unit (CPU) would give one ticket. Because of the SHA-256 algorithm, the output is a 256-bit value. Accordingly, the target value is also 256 bit. Setting the target is done by defining a space, usually, by defining the number of starting zeros in the hash value.

A target value of $2^2$ starting zeros is effortless to reach compared to a target value of $2^{80}$ starting zeros. As a result of the evolution of mining hardware, as explained in Section 3.3, the difficulty increases. Intentionally, the increase of difficulty is corresponding to the total hash rate of the network to maintain approximately 10 minutes between each block.

### 2.4.3 Fork

As mentioned in the introduction to this chapter, a temporary fork is when there are two valid blockchains on the Bitcoin network, illustrated in Figure 2.7. When it comes to changing the rules of Bitcoin, changes are resulting in either a *soft fork* or a *hard fork*.

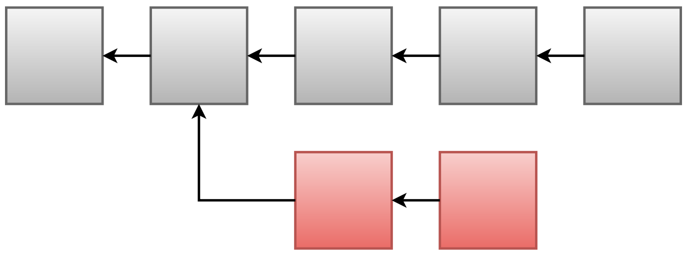

**Figure 2.7:** Temporary fork where the longest chain wins. The other fork is discarded, shown in red.

**Hard forks** are necessary when a feature is added to the Bitcoin protocol, which makes blocks invalid by the old rules implemented in the software [NBF$^+$16]. Nodes that updates the software in a hard fork will be able to accept the new blocks. However, the nodes using the old software, *legacy nodes*, will reject the blocks created by the new set of rules, as it is contradicting with the rules they are using. Legacy nodes will start working on a blockchain only consisting of valid blocks by the old protocol. Consequently, a fork will happen where contributions are depending on which software a node is running. Legacy nodes will only append blocks on the fork considered as the old blockchain. The two forks can never merge. Therefore the legacy nodes will be cut out of the network. An example of a change that will result in a hard fork is increasing the block size that is hard coded in the Bitcoin protocol, as discussed in Section 3.1.

A **soft fork** is happening when the change is restricting the validation rules [NBF$^+$16]. Legacy nodes will validate everything, while the new nodes running the updated software will reject blocks contradicting with the new set of rules. Thus, avoiding a hard fork as a result of the split between normal and legacy nodes. If the majority of nodes is running the new software, they will be enforcing the stricter set of validation rules. Stricter rules may lead to legacy nodes getting their new blocks rejected. Eventually, the operator of a legacy node will observe that its blocks are getting rejected, and then upgrade the software. An example of a change causing a soft fork is SegWit, see Subsection 2.3.3.

## 2.5 Network

Bitcoin is peer-to-peer network based, where every node is equal. It runs over TCP, and new nodes can join at any time. If a node wants to be a full node, it has to download the entire blockchain once, which is 152.2 GB checked at 12.04.2018. A Simplified Payment Validation (SPV) client is a node that will only check the blocks essential for the verifying transactions. Thus, SPV clients are only downloading fractions of the blockchain. For leaving the network, a node has to be inactive for 3 hours [NBF+16].

For a newly connected node to find peers, it will ask a known seed node for the known peers. Repeatedly, to extend the network, the newly connected node can ask the new peers for their peers, until it has enough connections. When a node generates a new block or makes a new transaction, the node broadcasts it every neighboring node. Subsequently, the neighboring nodes will broadcast the block or transaction to their neighboring nodes. This process is called a *gossip* protocol.

Since the network is a peer-to-peer based network, there will be some latency because of the distances between the nodes. Some nodes will receive the transaction or a block before other nodes. When a node broadcasts a new block, the other nodes will append the block to the blockchain after validating it. Subsequently, the nodes will start mining on top of the block they just received. Because of the latency through the network, there will be a scenario where two nodes find a new block at the same time and broadcast it to the network. Some nodes will receive one of the blocks first, and thereby discard the other block. However, other nodes can start to mine on the block that the other discarded, causing a temporary fork, see Subsection 2.4.3.

# Chapter 3 Scalability Problems and Possible Solutions

In the previous chapter, the fundamentals as blocks, proof-of-work consensus algorithm, and mining were explored. In this chapter, these elements will be investigated and analyzed as possible bottlenecks for scaling Bitcoin. Along with the problems, the applicable solutions will be presented. An *off-chain* solution will be presented to introduce payment channels.

Even though the popularity of Bitcoin has increased enormously, its technology struggles with keeping up with the interest. During the peak period of December 2017, users could experience long confirmation time of a transaction. At the 8th of December, the unconfirmed pool was on average at 177,729 transactions, with a peak of 181,908 unconfirmed transactions in the pool that day. By collecting and analyzing data from blockchain.info [S.A18a], we calculated the average unconfirmed pool size during December. We found that the average transactions in the unconfirmed transaction pool were 99056 transactions. Consequently, the transactions fee will rise, as miners prefer to include transactions with a high fee. A large pool of unconfirmed transactions, the *mempool*, also means that the network is congested. There are several elements for this congestion, examining those will one of the goals of this chapter.

## 3.1 Block size

In December 2017, the average number of unconfirmed transactions in the mempool were 99056 transactions, as described in the introduction to this chapter. Blockchain.info took 2-3 samples each day, one sample in the period from midnight to morning. Then one sample in the period from midday to afternoon. Lastly, sampling was also performed in the evening. By finding an average on each of these periods, we found that the evening period was the busiest with an average mempool size of 101298 unconfirmed transactions. However, this was the one with least samples. The average of the three periods gave an average mempool size of 99056 unconfirmed transactions.

How long a transaction stays in the unconfirmed pool, depends on how much transaction fee that is included in the transaction. For a fast payment, the transaction fee has to be high. A high fee gives the miners incentive to include that transaction before the other transactions with less fee included. If the user is unwilling to pay a high transaction fee, he might never get his transaction confirmed. This is because of a continuous stream of new unconfirmed transactions to the pool.

In Bitcoin, the block size is usually 1 MB, but it may vary due to SegWit, see Subsection 2.3.3. In December 2017 to January 2018, the average block size varied around 1.05 MB block size[1]. As mentioned in Subsection 2.4.1, one block can maximum hold 4000 transactions if the transaction is 250 bytes. Since average creation time of a block is every 10th minutes, the average throughput will be approximately seven transactions per second, see Equation 3.1.

$$\frac{\text{max Trans Per Block}}{\text{time Between Each Block in seconds}} = \frac{(4000 \cdot 1)}{(10 \cdot 60)} = 6.67 \tag{3.1}$$

In reality, transaction size is bigger, and thus transactions included in a block is fewer. In December 2017, the average transaction per block was 2225 transactions, based on data analysis of blockchain.info[2]. Using Equation 3.1, the throughput is 3.7 transaction per second. Combining the transaction throughput with the average unconfirmed pool size during December, a user would have to wait approximately 7 hours and 24 minutes. However, this calculation suggests a First In First Out (FIFO) queue. As mentioned, the transaction fee gives a priority. Thus the confirmation time could vary. Miners will rather include transactions with a high transaction fee, so with a low transaction fee, one could wait even longer to get the transaction confirmed. With 3.7 transactions per second, Bitcoin is incapable of competing with Visa at an average of 2000 transactions per second [CDE+16], and that is on a regular basis. During peaks, like Bitcoin experienced in December, Visa can manage 50,000 transactions per second.

Interestingly, as mentioned the transaction size may vary. The reason of this could be many. SegWit data may be included, as described in Subsection 2.3.3. Most importantly, the transaction size may vary because of input(s) and output(s). By investigating the block at block height 522,188, we discover that the block only contains four transactions. The transaction with *txid* $H$[3] contains 4 inputs and a total of 81 outputs. Accordingly, this transaction size is 3,261 bytes. The related

[1]https://blockchain.info/charts/avg-block-size?timespan=180days accessed 20.04.2018
[2]https://blockchain.info/charts/n-transactions-per-block accessed 07.05.2018
[3]$H$ = f3a954ad35f78a89aae0028c3f9df67974f0f231f510a89e0146016fee432f5f

Bitcoin address is probably belonging to an exchange service because of the large output number. By looking up the Bitcoin address, one can see that there are several large transactions done in and out from this address. The average transaction size during the previous 30 days was 611 bytes[4].

Whether to increase the block size or not is debated in the Bitcoin community. In fact, a hard fork, see Section 2.4.3, of the Bitcoin blockchain, took place on the 1st of August 2017. The reason for the hard fork was to increase the block size to 8 MB. As a result, another cryptocurrency named Bitcoin Cash was created[5]. By investigation Bitcoin Cash, one can see that the maximum transaction per second has increased to 53 transactions per second, see Equation 3.2. Why is still this such a splitting topic in the Bitcoin community?

$$\frac{(250/8 \cdot 10^6)}{(10 \cdot 60)} = 53.33 \tag{3.2}$$

One reason is that by increasing the total block size, and keep the average time of 10 minutes between each block, the amount of data contained by the blockchain increases dramatically. In some sense, it will be too big for full nodes to download the whole blockchain when installing the software. Another reason is that hard forks are avoided, see Subsection 2.4.3.

Assumed that a full block is created every 10th minute, the blockchain should grow by 52,560 MB per year, see Equation 3.3. If we look at 2017, the blockchain was 96,345 MB at the 1st of January. At 31st of December, the blockchain was 149,114 MB which gives us a difference of 52,769 MB, that is in range of the estimate. A reason for the estimate being lower than in reality could be intermediate time between blocks. The intermediate time could be lower as a consequence of the increased hash rate of the mining hardware. Another uncertainty about the estimate is that it is assuming full blocks. After one year, the Bitcoin Cash blockchain, see Equation 3.4, will be over twice as big as the blockchain of Bitcoin today. Bitcoin Cash will increase approximately 420,480 MB each year, assumed all the blocks are filled. Increased block size and required storing capacity can be challenging for the miners that have to store the entire blockchain.

$$1 \text{ MB} \cdot 6 \text{ times/hour} \cdot 24 \text{ hours} \cdot 365 \text{ days} = 52,560 \text{ MB/year} \tag{3.3}$$

$$8 \text{ MB} \cdot 6 \text{ times/hour} \cdot 24 \text{ hours} \cdot 365 \text{ days} = 420,480 \text{ MB/year} \tag{3.4}$$

[4]http://statoshi.info/dashboard/db/transactions accessed 06.06.2018
[5]https://www.bitcoincash.org/ accessed 10.05.2018

According to Decker and Wattenhofer [DW15], if Bitcoin is scaled up to 500 transactions per second, the miner must store 10 TB data on disk each year. For small miners and private users, this could be too demanding. Increasing the block size could potentially reduce the number of miners. This reduction of miners will again affect the security of Bitcoin, which will be discussed in Section 5.1. Increased block size could lead to Bitcoin being more centralized. Only serious actors would be able to provide the amount of storage needed. However, a 10 TB disk on Amazon costs 350 dollars[6]. Buying such a hard drive should be manageable for dedicated miners, but not for only running a full node.

## 3.2 Block propagation

Increasing the block size will increase the block propagation latency through the network. Originally, the creation of block propagation delay happens at each hop. How messages are exchanged between each node is shown in Figure 3.1. Firstly, a node receives a block. After the node verifies the block, it will broadcast that it has a new block to the neighboring nodes with an *inv* message. If a neighboring node accepts the block, it responds with *getdata*. Finally, the block is sent to the neighboring node, *block*. Decker and Wattenhofer [DW13] explored the Bitcoin network in 2013 to see how propagation delay causes Bitcoin forks. As a result of the research, blocks larger than 20kB will introduce an 80 ms delay for each kilobyte over 20 kB. This delay will increase until the majority of nodes knows the block. This finding gave a fork rate at 1.69% during a period of 10,000 blocks [DW13].

However, this research dates to 2013, and there are some new BIPs that is improving the block propagation time. BIP 152 suggests compact block relay in two ways, for peers with high bandwidth and low bandwidth. Originally, compact block relay was not designed for reducing block relay time, but to remove the overhead from block relay [Cor16]. As a result, the compact block relay protocol also reduced the propagation time. There are also some other interesting work on this field [Dev17][Riz17].

## 3.3 Energy Consumption

Bitcoin's energy consumption is another widely discussed topic. There are several estimates on the power consumption of Bitcoin [OM14][Vra17][Dig]. All the mentioned sources conclude with different estimates. Before discussing the different estimates, some terminology is needed.

[6]https://goo.gl/q7fVfq accessed 10.05.2018

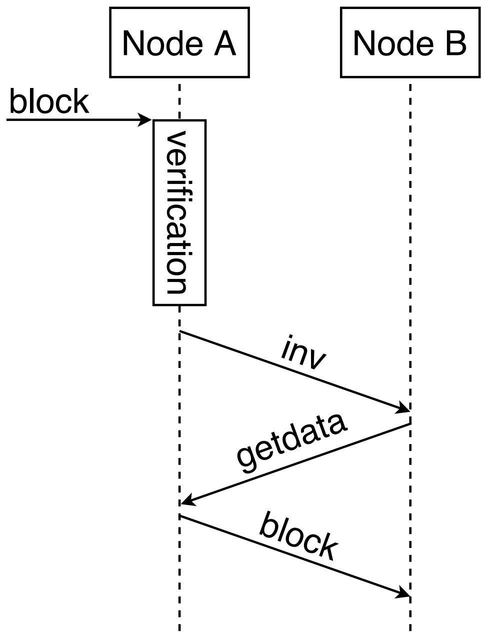

**Figure 3.1:** The process when a block arrives to a node, hop, and gets broadcast to the next node, hop. Adapted from [DW13]

- **Hash rate**. Hash rate is the number of hashes a miner can compute per second. This rate has increased during technological evolution. Usually measured in a mega hashes per second, (Mhash/s)
- **kWh**. kWh is a measure of how much energy used during one hour. 1 kWh = 3600 Watt = 3,600,000 joule. MWh is 1000 kWh. The average electricity consumed per capita in Norway during 2016 was 23.73 MWh, while the average in International Energy Agency (IEA) member countries[7] was 8.69 MWh [IEA].
- **Energy efficiency**. Energy efficiency means that a miner increases the number of hashes calculated with the same amount of power as before. Measured in hashes per Joule [Vra17].

In the beginning, Bitcoin miners used CPU for mining bitcoins. As miners achieved higher hash rates, the more incentive was given to the miner because he would succeed making more blocks. Therefore an arms race developed, and the CPU was soon outdated in terms of hash rate and efficiency. By increasing the efficiency, a miner can use less power to compute the same amount of hashes as before. After

[7]https://www.iea.org/countries/membercountries/ accessed 20.04.2018

the CPU, miners turned to the Graphics Processing Unit (GPU), a component in the graphics card of the computer. The GPU offers better efficiency and hash rate because it is used for calculating complex graphics. The GPU increases the hash rate because of utilizing a higher degree of parallelism [Vra17].

Miners used the third generation for a brief time. Field Programmable Gate Arrays (FPGA) could be customized by the Bitcoin miners to further improve the efficiency and hash rate compared to the GPU. Shortly after the introduction of FPGA, miners started using the fourth generation, the Application-Specific Integration Circuit (ASIC). An ASIC is a dedicated piece of hardware for mining. See comparison the evolution of mining in Table 3.1. If the ASIC is programmed to mine Bitcoin, it is only capable of mining Bitcoin, or other cryptocurrencies using SHA256 puzzles. ASICs is the most recent generation of Bitcoin mining hardware.

**Table 3.1:** The evolution of mining. Taken from [Vra17].

| **Hardware** | **Introduction** | **Hash rate (h/s)** | **Energy efficiency** |
|---|---|---|---|
| CPU | 2009 | $10^5$-$10^8$ | $10^4$-$10^5$ |
| GPU | Late 2010 | $10^6$-$10^9$ | $10^5$-$10^6$ |
| FPGA | Mid 2011 | $10^8$-$10^{10}$ | $10^7$ |
| ASIC | Early 2013 | $10^{10}$-$10^{13}$ | $10^8$-$10^{10}$ |

Surprisingly, even the development of each generation of ASIC is significant considering the hash rate and efficiency. AntMiner S1, that is the first SHA256 ASIC in the S-series from Bitmain[8], had an advertised rate of 180,000 Mhash/s. AntMiner S9, which is the latest version can offer a rate of 14,000,000 Mhash/s. Comparing the two ASICs, the newer generation improved significantly regarding efficiency. Even if the hardware gets more efficient, the energy consumption is still high.

More powerful hardware means that the difficulty of the puzzle will increase. Miners using outdated hardware will not be able to compete with miners with the newest hardware. When more miners get the newest hardware, it will lead to an increase in the difficulty. This is a phenomena called *tragedy-of-common* [WL15]. Relating this to the example of Bitcoin, miners are buying the most recent and efficient hardware. This is resulting in an increased difficulty. Because of the increased difficulty, one would need many ASICs to create a block. Whereas, in the early days of Bitcoin, people could mine on their personal computer. However, the security of the Bitcoin blockchain is increased, as discussed in Section 5.1.

While O'Dwyer and Malone [OM14] estimates that Bitcoin consumes as much energy as Ireland 3 GW, more correctly, they found that the consumption ranges

[8]https://en.bitcoin.it/wiki/Mining_hardware_comparison accessed 25.05.2018

from 0.1 GW to 10 GW. Vranken [Vra17] suggest that Bitcoin consumes far less than O'Dwyer's and Malone's estimate, it consumes closer to 100 $MW$. O'Dwyer and Malone are calculating for both CPUs and ASICs. Upper and lower limits are calculated because the Bitcoin network is composed of various equipment.

Another approach to estimating the power consumed by Bitcoin is to look at the total estimated hash rate in the network which is 30,345,703 TH/s according to blockchain.info[9]. If we divide the total hash rate by the hash rate of an AntMiner S9, 14,000,000 MH/s, we will see that it takes approximately 2,167,550 devices to achieve the total hash rate. Note that the network is composed of various mining equipment, so this will bring some uncertainty to our estimate.

By taking one of the most efficient ASICs, we will get a lower bound. It is also arguable that the miners will run efficient hardware, or else they will lose money mining because of the high electrical bill and low reward for mining. The AntMiner S9 is consuming 1375 W. Multiplied with all the devices, we will get a total consumption of approximately 3 GW if all the devices used in the network are the AntMiner S9. Our estimate is close to the estimate of O'Dwyer and Malone. The AntMiner S9 has a fan already installed, so some of the consumption regarding cooling is taking into consideration. Nonetheless, if many AntMiner S9s composes the mining rig, an external cooling system would be needed. An external cooling system is not included in this estimate. In addition, some networking gear is required.

If Bitcoin is already consuming 3 GW, it will not scale well, unless decreasing the power consumption. On the other hand, the banking system is consuming 2,340,00 GW [Vra17]. However, the transaction volume cannot be compared, as well as the other services offered by the banks.

As mentioned, the hash rate contributes to the high energy consumption. The hash rate is necessary for computing the puzzle and providing the security of Bitcoin. Proof-of-work is the fundamental component of Bitcoin that makes mining unavoidable, see Subsection 2.4.2. There are some suggestion to replace the proof-of-work algorithm with something less power needing. Proof-of-stake is a consensus algorithm that is suggested for solving the power consumption problem.

### 3.3.1 Proof-of-Stake

Proof-of-Stake (PoS) is a concept of *virtual mining* with various types of implementations. By virtual mining, there is no specialized hardware needed, like there is in PoW. Generally, the idea is that the security of the blockchain should be maintained by those who are financially invested in the blockchain rather than solving difficult

[9]https://blockchain.info/charts/hash-rate accessed 25.05.2018

computational puzzles. Subsequently resulting in a reduction of the amount of electricity spent mining. Various implementations differ when determining who should be elected to mine or forge a new block.

Mainly, PoS is different from PoW in the process of who should be allowed to append the next block. Recall Subsection 2.4.2 that to append a block to the blockchain, a miner must solve a computational puzzle. He is competing against the rest of the network, and the more computer power and hash rate he can achieve, the more likely it is for him to find a solution to the puzzle first. PoS on the other hand, is electing a small group of miners, called validators, to include transactions in a block and find its hash, then publishing it. One could say that in PoW computational power is necessary to be able to vote, while in PoS, it is the *stake.* The stake could refer to the asset or funds that a node locks up in a *bond transaction*, or basically, the amount of currency one is holding.

In some cases, there could be only one validator elected, as in Nxt [com14]. A validator is elected by its stake. If the validator publishes a block, but it gets rejected because of an invalid transaction, the validator will lose the funds locked up, the stake. The mentioned scenario is what motivates the validators to be honest. If the block is accepted, the validator gets the transaction fees from the transactions included in the block.

PeerCoin [KN12] have implemented a hybrid between PoW and PoS [NBF$^+$16]. For the PoS algorithm, *coin age* is used instead for stake. Coin age is the age of the coin, in blocks, starting from the most recent transfer. If Alice gives Bob 1 BTC at block 50, the coin age will be 400 at block 450, only if it is unspent. When a validator successfully uses the coin age for mining a new block, the coin age reset. As mentioned, Peercoin uses PoW too, so the old SHA256 puzzle must still be solved. What is different in Peercoin, is that the coin age is used to set the difficulty of the puzzle. If a miner has a considerable amount of mining power, he can still compete with the others, even though the coin age is low. However, this scheme favors a high coin age, because the computational puzzle will be significantly easier. Resetting the coin age will in some instances prevent the rich from getting richer. This could be a problem in a PoS where the stake is the amount of currency. In Nxt [com14], the funds have to be stationary on an account for 1440 blocks before they can be used as a stake to mine/validate a block.

Recall from Section 2.4 that the blockchain bases its security on the hard puzzle a miner must compute. Chaining each of those puzzles makes it problematic to replace blocks in the blockchain. Intentionally, the difficulty attacking a block, increases with the height of the block and blocks appended after it. How is PoS achieving the same level of security if the puzzle is not so hard to compute? Nxt has implemented a fixed checkpoint [com14]. After 720 blocks, one cannot change the blockchain. Even if an attacker has high stakes, he will not be able to change the blockchain or make a temporary fork longer than 720 blocks.

To perform a 51% attack on the Bitcoin blockchain, the adversary needs to obtain 51% of the mining power. Recall from Subsection 2.4.3 temporary forks may appear. The *Nakamoto conensus* decides that the longest chain will live. By controlling the majority of the total hash rate, the attacker can create a fork of the original blockchain. In this fork, he can include double spend transactions, see Subsection 2.3.3, and censor transactions. Because of holding 51% of the hash rate, in the long run, this fork will be longer than the main blockchain. When the attacker publishes this fork on the Bitcoin blockchain, nodes will drop the original chain. The fork is now the main chain, and the double spend transactions are validated. This attack is specific to the PoW blockchain.

For an adversary to perform the same attack on a proof-of-stake blockchain, he has to obtain 51% of the currency. If Bitcoin were using the proof-of-stake algorithm, the attacker would need to buy approximately 3,335,807 BTC[10]. A total cost of 33,000,000,000 USD worth 30,885,831 AntMiner S9 with a total hash rate of 401,000,000 TH/s. In the Bitcoin network, the total hash rate is around 27,591,393 TH/s[11]. By investing this kind of money in the attack, it is way more profitable to attack a proof-of-work blockchain, based on the numbers of Bitcoin. The results of the estimate will vary of the coins in circulation and how much each coin is worth.

On the one hand, acquiring a significant portion of the coins in a cryptocurrency will drive its price up, for example, the first coin will be cheaper than coin number 10,000 because the demand of the currency will increase. On the other hand, if the sellers sell the coins for the same price, it will be cheaper for an attacker to acquire the coins needed for an attack. Most likely, the price of each coin will increase as the attacker must buy a large quantum for his attack.

Finally, PoS may solve the electrical power consumption problems Bitcoin is facing, but is it solving the transaction throughput? In Nxt, a block can contain up to 255 transactions. With an average block creation time of 80 seconds [com14], this gives three transactions per second, which close to Bitcoins throughput. However, by reducing the intermediate time between block, the throughput increases. However, decreasing the intermediate block time could introduce more temporary forks, see Section 3.2.

[10]https://coinmarketcap.com/ accessed 05.05.2018
[11]https://blockchain.info/charts/hash-rate accessed 05.05.2018

### 3.3.2 Decentralization

Consequently, when mining gets optimized, all the mining operation would take place where the energy is cheapest. Two of the biggest mining pools are owned by the Chinese company Bitmain, who is also producing popular mining equipment such as the AntMiner series. BTC.com, 25.7%, and AntPool, 17.6%, produce accordingly 43.3% of all blocks produced, see Figure 3.2. This estimate is over four days.

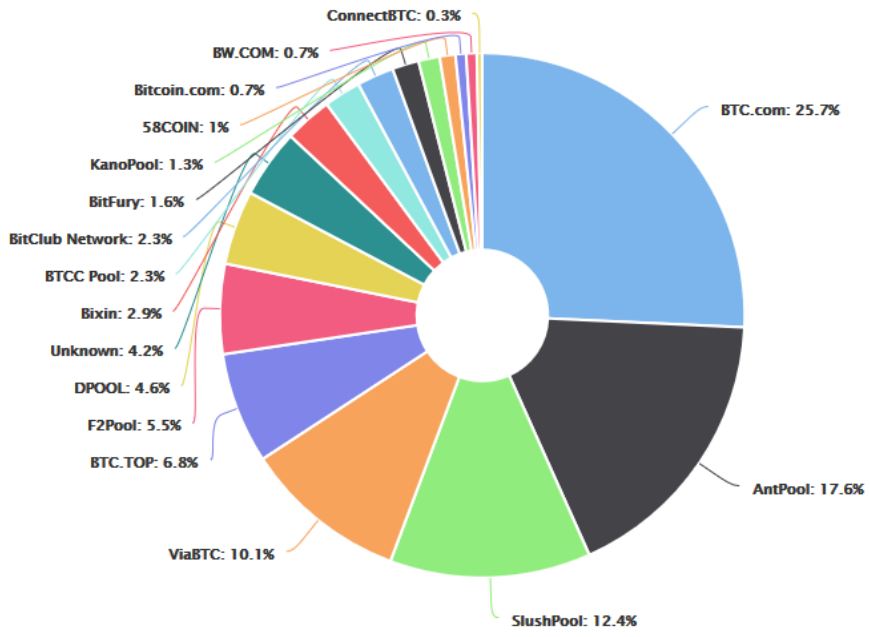

**Figure 3.2:** A snapshot of the estimated hash rate distribution of largest mining pools over four days. Taken from blockchain.info [S.A18b]

Bitcoin that once was a decentralized cryptocurrency tends to be more centralized regarding mining. One could argue that the currency will be in the hands of the Chinese government. If the government put regulations for mining a cryptocurrency, the miners have to relocate, and that could impact the cryptocurrency itself.

Although Bitmain owns the two biggest pools, the miners themselves are spread out through the world, where the biggest amount of miners are located in USA and

Germany[12]. Even though the miners are spread around the world, a mining pool can be a single point of an attacker who wants to control a significant portion of the mining power. In addition to that, the mining pool managers can prioritize and reject transactions, to censor Bitcoin addresses. When a miner joins a mining pool, the pool manager can decide upon which blocks the miners should be working on.

## 3.4 Off the blockchain

*Off-chain* transactions is one of the most promising solutions to the Bitcoin scalability problems, addressing both instant payments and throughput. Off-chain aims to move transaction off the blockchain. That means avoiding publishing every transaction. Thus, the throughput could be scaled. However, a solution to the double-spend problem, see Section 2.3.3, is required. Necessarily, a public dispute solver is needed in cases of disputes, which will be the task of the Bitcoin blockchain. Off-chain solutions primarily want to make *microtransactions* off-chain. A microtransaction is a transaction of small value, for example, the price of a coffee cup or less. The security around microtransactions does not necessarily have to be as good as normal transactions on the blockchain, because losing the money worth a cup of coffee is acceptable. Essential to all off-chain models is the multisignature transaction, see Subsection 2.3.2. There are different implementations and ideas how off chain transactions should be implemented.

#### Duplex Micropayment Channels

Duplex Micropayment Channels (DMC) is used as an example of a different way of implementing off-chain payments. However, this solution is no longer under development.

Decker and Wattenhofer [DW15] want to enable off-blockchain transactions through channels. A channel, or a payment channel, is a connection between to nodes used for transferring microtransactions. A Payment Service Provider (PSP) establishes channels with other PSPs, similar to the autonomous systems and the Internet Service Provider (ISP)s in the Internet today. By routing transfers between end users, they aim to achieve end-to-end security. Also, by making the transfers final would result in instant payments. The Bitcoin blockchain will only be used to set up and closing channels. Using multisignatures, contracts and time locks, see Subsection 2.3.1, DMC aims to enable off-chain payments.

A channel is established by a *funding transaction* where both parties place funds in a 2-of-2 multisignature. To spend from this output, both parties must agree and sign the transaction spending form the output of the multisignature transaction. By

[12]https://bitnodes.earn.com/ accessed 07.05.2018

combining two unidirectional channels, DMC enables payments to go both ways. An *invalidation tree* is used to update the current balance in both channels. The invalidation tree is a structure of time locked transactions [MMSH16]. One problem with a unidirectional channel from Alice to Bob is that if Alice uses all her funds, she will not be able to send Bob anymore, even if he made a transfer after the exhaustion through another unidirectional channel. The invalidation tree resets both channels, so in the scenario where Alice used all her funds, but Bob paid her through another unidirectional channel, she will be able to spend it in her channel. Both parties need to agree on a new branch in the invalidation tree to reset the balance.

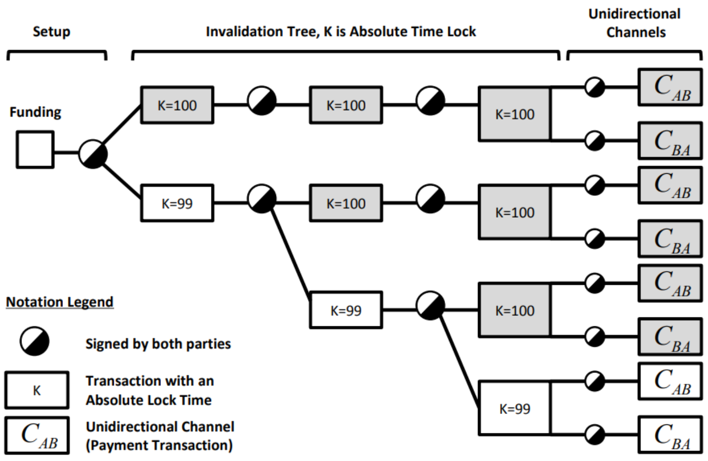
Setup
Invalidation Tree, K is Absolute Time Lock
Unidirectional Channels
Funding
K=100
K=99
$C_{AB}$
$C_{BA}$
Notation Legend
Signed by both parties
K
Transaction with an Absolute Lock Time
Unidirectional Channel (Payment Transaction)

**Figure 3.3:** Invalidation tree. When a unidirectional channel is exhausted, a new branch on the invalidation tree is created to reset the balance. A time lock is then incremented. Taken from [MMSH16]

Closing the channel is done by both parties signing a transaction without any lock time. Then the transaction is published on the blockchain. In the scenario where cooperation is impossible, either party should broadcast the active branch in the invalidation tree. However, this transaction is set by a time lock, so either party must wait for the time lock to run out.

### Unidirectional versus Bidirectional channels

There are two kinds of channels when considering off-chain payment solutions. A unidirectional channel is a channel where funds can be sent in one direction only. Unidirectional channels is essential for the DMC to work, see Figure 3.4, which will be explained. One problem with DMC is that resetting the channels after one direction is exhausted is necessary. What if Bob could send his funds back to Alice in the same channel, with funds he recently got from her? DMC is creating a bidirectional channel by combining two unidirectional channels, and by storing the transactions with the incremented time locks in the invalidation tree. However, when a unidirectional channel is exhausted, both channels will be reset to update the balance. A bidirectional channel is a channel where a payment can be sent in both directions in the same channel, see Figure 3.5. The *Lightning Network* is using one bidirectional channel, where the balance is updated without resetting the channel. The *Lightning Network* will be introduced in the next chapter, Chapter 4.

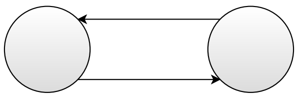

**Figure 3.4:** Two unidirectional channels enables both parties to send funds to each other.

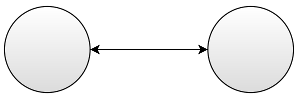

**Figure 3.5:** One bidirectional channel enables both parties to sends funds to each other in the same channel.

Chapter 4

# Lightning Network

In the previous chapter, bottleneck regarding the scalability of Bitcoin was explored. By proving that the network was congested in December 2017, we demonstrated that a solution avoiding a hard fork is necessary. We defined off-chain solutions and looked into one particular solution. In this chapter, we will investigate the *Lightning Network*, which is currently being under development. After exploring the semantics of the Lightning Network, the challenges with this solution will be discussed.

When the transaction fees increase because of a congested network, it will be unbearable to pay the coffee with bitcoins. Another problem is that the merchant will not receive the payment instantly, and with a congested Bitcoin network, it can take hours for the transaction to be confirmed, as discovered in Section 3.1. Poon and Dryja came up with the Lightning Network that aims to solve the scalability problems of Bitcoin, primarily, the throughput of transactions per second.

To explain what the Lightning Network is, it is necessary to do some simplifications. The Lightning Network can be described as a stack with four layers according to Osuntokun[1]. Bottom-up, we start with the lowest layer, which is the Bitcoin layer. See Chapter 2 where the blockchain of Bitcoin is described. The second layer is the channel link layer, which is the actual establishment of a channel and the transactions happening in it. Since the Lightning Network is functioning as a network, there is a routing protocol needed. In the case of the Lightning Network, it is the Hashed Time-Locked Contract (HTLC) which provides the binding of the transactions over the network, and it can be divided into its layer, layer 3. HTLC is described in Section 4.3. The last layer is the application layer, which can be the users using their phones when paying for the coffee or building exchanges on top of the Lightning Network. Usually, Lightning Network is described as a second layer network.

The Lightning Network is an off-chain solution that uses bidirectional payment channels, as described in Section 3.4, to transfer funds between two parties. Purpose-

[1]https://www.youtube.com/watch?v=V3f4yYVCxpk&t=62s accessed 20.05.2018

fully, the payment channel is used for microtransactions that are not published on the blockchain until specific criteria are fulfilled. Recall the multisignature transaction from Subsection 2.3.2, where both parties had to sign the transaction before it got broadcast to the Bitcoin network. Lightning Network uses the 2-of-2 multisignature transaction between two parties as a building block. By spending from the multisignature address, microtransactions can be executed off-chain, until both parties signs and broadcast the last transaction spending the multisignature output. Time locks are used as countermeasures against funds being locked up in a channel by a party refusing to sign, see Section 2.3.1.

## 4.1 Requirements

Necessarily, there should be some requirements to the Lightning Network, and Prihodko et al. [PZS$^+$16] defined the following requirements:

- Peer-to-peer network. All nodes should be able to be senders and receivers of transactions and funds both.
- Source routing scheme. Lightning Network should provide user privacy, censorship resistance, fee predictability and time-lock predictability
- Trustlessness. It should work like in Bitcoin where there is no need for a third party involvement.
- Fast payment processing. In order to improve the confirmation time compared to Bitcoin, there should be a way that a payment can be instant, maximum minutes.
- Security. Payment should reach the receiver if not, the sender should be able to roll back the payment. Should not impose a security threat to the blockchain.
- Probabilistic route discovery. Finding the route between two nodes should be probabilistically high because it is possible to establish new channels if none is found.
- Moderate operational resources. Intentionally, users should be able to use their phones when making their payment. Therefore, Lightning Network should be able to support SPV wallets. Smartphones also have restricted computer power and storage compared to a desktop. It is important that this will not compromise security.
- Scalability. To scale Bitcoin and other cryptocurrencies are the main reason for Lightning Network creation in the first place. Lightning Network should be able to scale up to thousands of transactions per second and supporting millions of users as Bitcoin will be adopted worldwide.

– Automation. Lightning Network should be user-friendly, so most of the process should be automated in the sense that the user should decide some options, e.g., maximum transaction fee. On the other hand, it should also be possible for more advanced users to decide the route themselves.

## 4.2 Transactions

To initiate a payment channel, Alice and Bob put some bitcoins into the 2-of-2 multisignature transaction, called the *funding transaction.* To spend the output, before the funding transaction itself is on the blockchain, Poon and Dryja propose a SIGHASH_NOINPUT script. In addition to the funding transaction, there are in four more underlying transactions in a Lightning Network channel [MMSH16]:

– Commitment Transaction

– Revocable Delivery Transaction

– Delivery transaction

– Breach Remedy Transaction

As a transaction in the Lightning Network is a Bitcoin transaction, there are at least one input and one output field in each transaction, described in Section 2.3. A **commitment transaction** is spending from the funding transaction output. Since the funding transaction is a 2-of-2 multisignature transaction, the commitment transaction requires signatures from both Alice and Bob to be broadcast. Generating a new key pair for each commitment transaction is necessary. To avoid publishing every commitment transaction to the blockchain, Alice and Bob exchanges signatures beforehand [PD16]. Then both Alice and Bob create their version of the commitment transaction. Because Alice and Bob exchanged the signature beforehand, either party can close the channel. This is possible because the commitment transaction will include the counterparty's signature. Closing the channel is done by signing and broadcasting the most recent commitment transaction, which already is signed by the counterparty.

The commitment transaction is the transaction that keeps track of the balance on a channel. It has two outputs [PD16]. One output for immediate redemption for the receiver, and one for delayed redemption for the sender forced by a contract called Revocable Sequence Maturity Contract (RSMC). A RSMC is a contract used to delay the payment in the output that the RSMC is affecting [PD16]. Intentionally, the commitment transaction works like a balance transaction where both Alice and Bob can show what the most recent balance was in case of a dispute. Updating the balance is done by creating a new commitment transaction. Imagine the scenario where there are two commitment transactions on the channel at the same time. How does the Lightning Network prevent an old commitment transaction with an outdated balance to be broadcast?

A **breach remedy transaction** is used to invalidate and punish the party that broadcast an old and invalid commitment transaction. A breach remedy transaction is spending from the RSMC output from the commitment transaction. If Alice broadcasts an old and invalid commitment transaction, Bob can broadcast a breach remedy transaction and take all the funds in the channel as a punishment. Note that the breach remedy transaction is spending from the same output as the revocable delivery transaction, that is the reason why the counterparty can steal all funds from the channel.

The breach remedy transaction is signed by both parties when they decide to invalidate an old commitment transaction. After Alice publishes the old commitment transaction, her funds are locked in the first output by the relative time lock, see Subsection 2.3.1. Because of the time lock, Bob can scan the blockchain and discover that the transaction published by Alice is old. Bob will then have the time to broadcast a breach remedy transaction to collect the bitcoins locked by the relative time lock. See Figure 4.3. Therefore, Alice has no incentive in broadcasting an old and invalid commitment transaction after signing a breach remedy transaction.

A **revocable delivery transaction** is spending from the first output of the commitment transaction. Recall that the breach remedy transaction is spending from the same output, hence only one can be confirmed. A revocable delivery transaction spends from the script output, consisting of a *relative timelock*. Firstly, the commitment transaction is published. Secondly, after a given number of blocks, the revocable delivery transaction can spend the output from the commitment transaction. RSMC is the contract describing what will happen with the funds of the party creating the commitment transaction. The purpose of the revocable delivery transaction is to send funds back to the broadcaster of the commitment transaction after a given amount of time.

The **delivery transaction** is spending the second output of the commitment transaction. The delivery transaction is sending funds to the counterparty of the broadcaster. If Alice wants to close the channel, she broadcasts the commitment transaction, C1a. Bob can then immediately redeem the funds by signing the delivery transaction, D1, see Figure 4.3.

Figure 4.1 describes how a channel will look like with two commitment transactions. In the figure, a party successfully broadcast the second commitment transaction and closed the channel. All the white transaction was only inside the channel, and therefore invisible to the blockchain. Once a party publishes the commitment transaction, it will be impossible to spend from the same output of the funding transaction, and therefore a new channel needs to be established between Alice and Bob for future transactions.

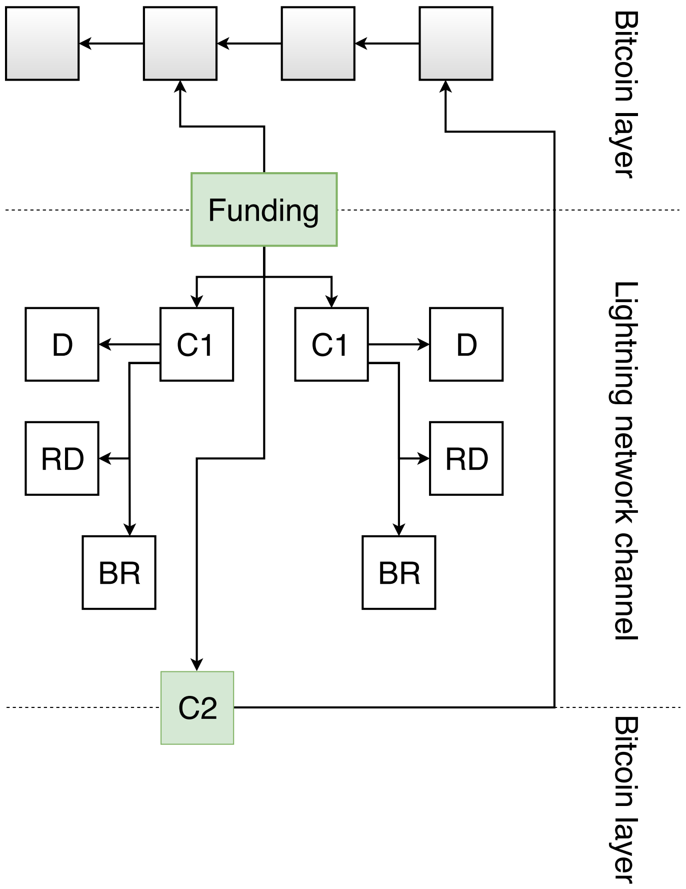

**Figure 4.1:** Figure showing how a Lightning Network channel works. C1 is a commitment transaction, D is the delivery transaction, RD is the revocable delivery transaction, and BR is the breach remedy transaction. Adapted from [PD16].

In the Lightning Network, the pre-generated keys are stored in a Merkle tree [PD16]. Russell [Rus15] suggests using a hash function as an authentication method, where the owner of the commitment transaction includes a contract with a hash value. This will replace the pre-generated keys. To invalidate a commitment transaction, the owner of the transaction reveals the preimage to the hash value. By using hash value and corresponding preimage for authenticating, key storage for all the transactions is no longer necessary. Avoiding key storage will also improve the security of an attack against either Alice and Bob. However, the Lightning Network implementations are implemented with public key cryptography and digital signatures to validate and invalidate transactions.

## 4.3 Hashed Time-Locked Contract

Hashed Time-Locked Contract (HTLC) makes it possible to construct a route of payment channels between Alice and the coffee shop, without having any direct payment channel with it. In other words, the HTLC is the mechanism that makes it possible with secure transfers over the Lightning Network with multiple hops.

Alice wants to pay for her coffee, but she has no direct channel with the coffee shop. Instead, the Lightning Network routes the payment through Bob and then the coffee shop. So, to secure this route, the coffee shop sends Alice a hash value $H(R)$ based on preimage $R$ to construct an HTLC. Transmission of the hash value is sent over the Internet, in the same way as the establishment of the payment channel. Alice then makes a transaction where the output is an HTLC, see Figure 4.2 for a simplified version. Together with this transaction, she sends Bob the hash value, $H(R)$. There is also an *absolute lock time*, $K$, that decrements at each hop included in the transaction. The time lock helps intermediary nodes to redeem the money from the previous hop in a time span after the funds sent from the intermediate node are redeemed. In our case, it could be $K = 2$ for the first hop, Alice to Bob, $K = 1$ for the final hop, Bob to the coffee shop. However, how do we know that the final receiver is, in fact, the rightful node to redeem the funds?

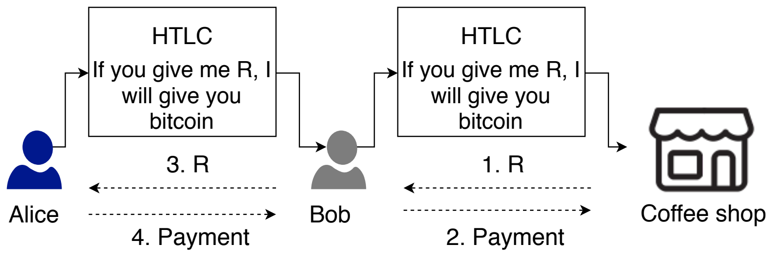
HTLC
If you give me R, I will give you bitcoin
HTLC
If you give me R, I will give you bitcoin
3. R
1. R
Alice
Bob
Coffee shop
4. Payment
2. Payment

**Figure 4.2:** HTLC contract used for routing. Showing the transaction order for the payment. Establishment of the HTLC is already done. Adapted from [PD16].

When the coffee shop gets the funds, it will reveal the preimage $R$ to Bob to redeem the funds from the HTLC output. Bob will send the preimage he got from the coffee shop to Alice to redeem his funds from Alice. When redeeming the funds, the party redeeming the funds signs a *HTLC Execution Delivery.* How the party redeems the funds is shown in the *OP_IF* below. If $R$ is not provided before the time lock $K$ expires, the sender can reclaim the funds by providing his/her signature.

Therefore it is essential that a receiver will not reveal the preimage after the time lock runs out. If the time lock runs out, the HTLC will refund all the parties where the next hop did not provide the preimage $R$. This refund is described in *OP_ELSE*.

```
OP_IF

    OP_HASH160 <Hash160 (R)> OP_EQUALVERIFY

    2 <Alice2> <Bob2> OP_CHECKMULTISIG

OP_ELSE

    2 <Alice1> <Bob1> OP_CHECKMULTISIG

OP_ENDIF
```

In the Lightning Network, there will be a specific HTLC output in the commitment transaction in addition to the other outputs described in Section 4.2. For Bob to redeem the bitcoins, he must broadcast the *Delivery Transaction*. For a valid *Delivery Transaction*, $R$ needs to be included. There are three ways to redeem a commitment transaction with a HTLC output [MMSH16].

1. If the receiver can provide a signature and the preimage $R$, the receiver can redeem by broadcasting a *HTLC Execution Delivery* transaction.
2. If the time lock has expired and the sender can provide his/her signature, the sender can redeem the funds by broadcasting a *HTLC Timeout* transaction.
3. If the transaction is revoked, but later broadcast, the counterparty can redeem it by its signature and preimage $R$. *HTLC Timeout Breach Remedy* and *HTLC Breach Remedy* is the transactions invalidating the HTLC.

If Bob can not provide the preimage to Alice, she can broadcast a *HTLC Timeout* transaction. Consequently, broadcasting this transaction would lead to closing the channel, and committing the commitment transaction to the blockchain. To keep the transactions off-chain, Alice and Bob should agree on creating a new commitment transaction where Alice gets refunded, and avoid broadcasting the HTLC Timeout transaction. However, if Bob is uncooperative, Alice must broadcast this transaction, which will close the channel.

After creating a new commitment transaction that is updating the balance after a time lock expires, both parties should invalidate the HTLC Timeout transaction. The sender signs a *HTLC Timeout Breach Remedy* to invalidate the HTLC Timeout transaction. If the belonging commitment transaction, which is invalidated, is broadcast, the receiver can immediately redeem the funds from this transaction.

Recall that Bob can redeem a HTLC Execution Delivery transaction, to enable off-chain transactions, this transaction should also be invalidated. *HTLC Breach Remedy* transaction enables Bob to steal all the funds if Alice broadcasts this old commitment transaction. See Figure 4.3 for a fully revoked commitment transaction and terminated HTLC [PD16]. When fully revoking a commitment transaction, a party who publish the commitment transaction will be punished and lose all funds in the channel.

## 4.4 Evolution of the Lightning Network

Poon and Dryja published the Lightning Network white paper in 2015. Three companies are currently working on the Lightning Network, Blockstream, Lightning Labs and ACINQ. Respectively, *c-lightning*, *lnd* and *eclair* are the projects worked on the the three companies. As these projects are under constant development, the Lightning Network changed from when Poon and Dryja first introduced it in 2015. This section will explore the changes according to the RFC defined by all three companies [RAM+18]. Documents called Basis of Lightning Technology (BOLT) describes the Lightning Network. So far there are eleven different BOLTs describing different functions and protocols.

The most significant difference is that the transaction set described in Section 4.2, is changed. Only the funding and the commitment transactions are kept. However, having the transactions from the original paper in mind could help to understand how Lightning Network works. Necessarily, the funding transaction is used to establish a channel.

### 4.4.1 Channel opening

When two nodes have found each other, they can open a channel. See Figure 4.4 for which messages exchanged to establish a funding transaction:

1. *open_channel*: Alice sends an open_channel message to Bob, containing 20 fields of data. The fields describe for instance which blockchain Alice wants to use, how much funds she wants to put in the transaction and maximum HTLCs the channel accepts [RPM+18a].

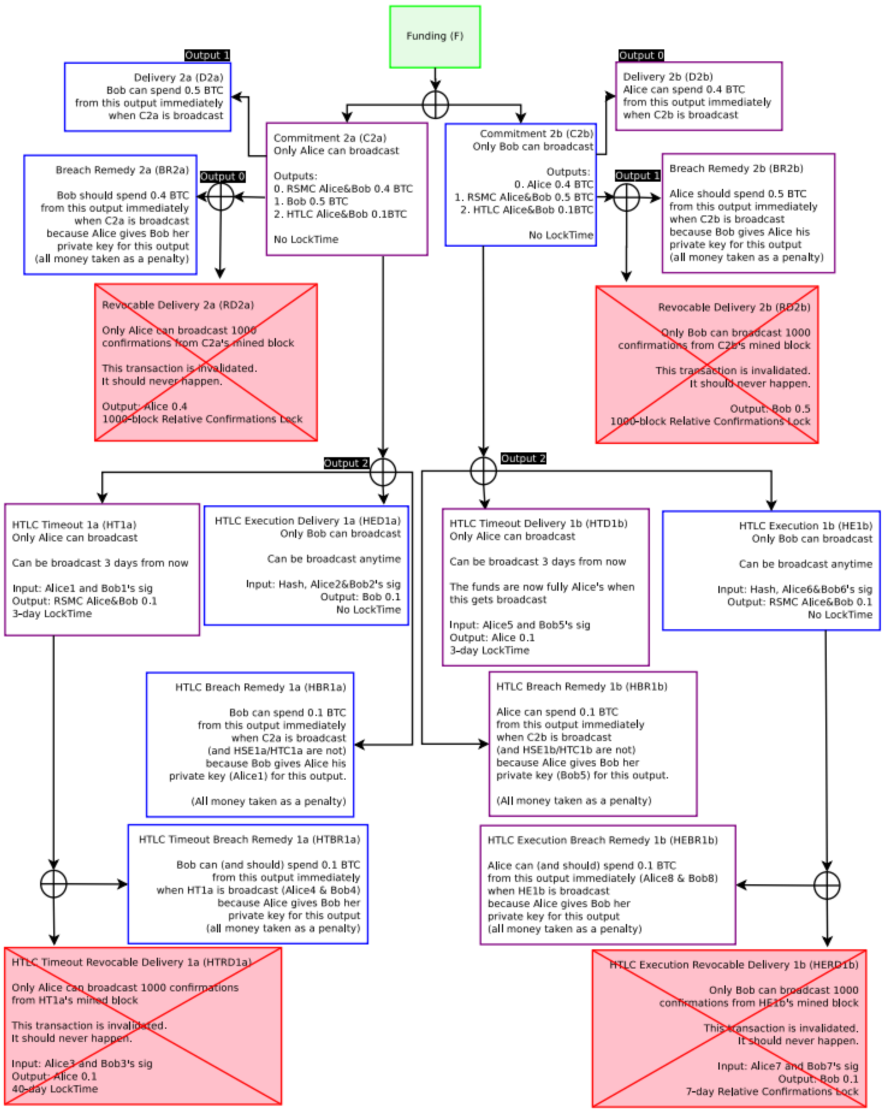
Funding (F)
Output 1
Delivery 2a (D2a) Bob can spend 0.5 BTC from this output immediately when C2a is broadcast
Output 0
Delivery 2b (D2b) Alice can spend 0.4 BTC from this output immediately when C2b is broadcast
Commitment 2a (C2a) Only Alice can broadcast
Outputs: 0. RSMC Alice&Bob 0.4 BTC 1. Bob 0.5 BTC 2. HTLC Alice&Bob 0.1BTC
No LockTime
Commitment 2b (C2b) Only Bob can broadcast
Outputs: 0. Alice 0.4 BTC 1. RSMC Alice&Bob 0.5 BTC 2. HTLC Alice&Bob 0.1BTC
No LockTime
Breach Remedy 2a (BR2a) Bob should spend 0.4 BTC from this output immediately when C2a is broadcast because Alice gives Bob her private key for this output (all money taken as a penalty)
Breach Remedy 2b (BR2b) Alice should spend 0.5 BTC from this output immediately when C2b is broadcast because Bob gives Alice his private key for this output (all money taken as a penalty)
Revocable Delivery 2a (RD2a) Only Alice can broadcast 1000 confirmations from C2a's mined block. This transaction is invalidated. It should never happen. Output: Alice 0.4 1000-block Relative Confirmations Lock
Revocable Delivery 2b (RD2b) Only Bob can broadcast 1000 confirmations from C2b's mined block. This transaction is invalidated. It should never happen. Output: Bob 0.5 1000-block Relative Confirmations Lock
Output 2
HTLC Timeout 1a (HT1a) Only Alice can broadcast. Can be broadcast 3 days from now. Input: Alice1 and Bob1's sig Output: RSMC Alice&Bob 0.1 3-day LockTime
HTLC Execution Delivery 1a (HED1a) Only Bob can broadcast. Can be broadcast anytime. Input: Hash, Alice2&Bob2's sig Output: Bob 0.1 No LockTime
HTLC Timeout Delivery 1b (HTD1b) Only Alice can broadcast. Can be broadcast 3 days from now. The funds are now fully Alice's when this gets broadcast. Input: Alice5 and Bob5's sig Output: Alice 0.1 3-day LockTime
HTLC Execution 1b (HE1b) Only Bob can broadcast. Can be broadcast anytime. Input: Hash, Alice6&Bob6's sig Output: RSMC Alice&Bob 0.1 No LockTime
HTLC Breach Remedy 1a (HBR1a) Bob can spend 0.1 BTC from this output immediately when C2a is broadcast (and HSE1a/HTC1a are not) because Bob gives Alice his private key (Alice1) for this output. (All money taken as a penalty)
HTLC Breach Remedy 1b (HBR1b) Alice can spend 0.1 BTC from this output immediately when C2b is broadcast (and HSE1b/HTC1b are not) because Alice gives Bob her private key (Bob5) for this output. (All money taken as a penalty)
HTLC Timeout Breach Remedy 1a (HTBR1a) Bob can (and should) spend 0.1 BTC from this output immediately when HT1a is broadcast (Alice4 & Bob4) because Alice gives Bob her private key for this output (all money taken as a penalty)
HTLC Execution Breach Remedy 1b (HEBR1b) Alice can (and should) spend 0.1 BTC from this output immediately (Alice8 & Bob8) when HE1b is broadcast because Alice gives Bob her private key for this output (all money taken as a penalty)
HTLC Timeout Revocable Delivery 1a (HTRD1a) Only Alice can broadcast 1000 confirmations from HT1a's mined block. This transaction is invalidated. It should never happen. Input: Alice3 and Bob3's sig Output: Alice 0.1 40-day LockTime
HTLC Execution Revocable Delivery 1b (HERD1b) Only Bob can broadcast 1000 confirmations from HE1b's mined block. This transaction is invalidated. It should never happen. Input: Alice7 and Bob7's sig Output: Bob 0.1 7-day Relative Confirmations Lock

**Figure 4.3:** Fully revoked commitment transaction where also the HTLC is terminated. Taken from [PD16].

2. *open_channel*: Bob checks if the data given is reasonable before he accepts the channel by sending an open_channel message back to Alice.

3. *funding_created*: Alice creates the funding transaction by the parameters agreed upon, signs it herself and sends it to Bob for his signature. All inputs should be SegWit inputs to prevent malleability [RPM+18a], see Subsection

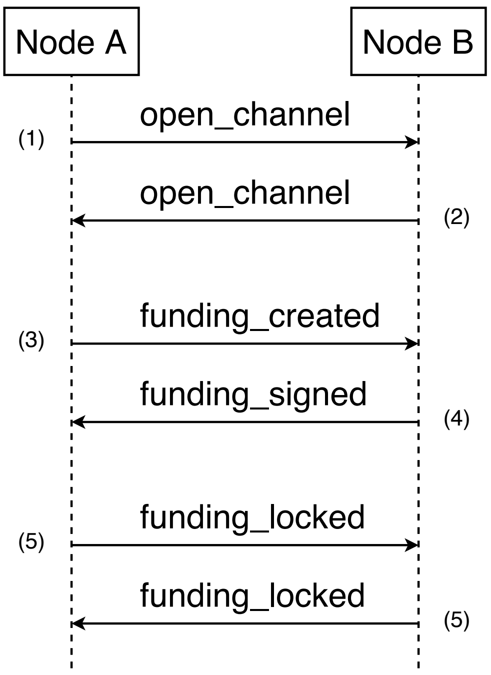

**Figure 4.4:** Establishment of a channel and funding transaction. Adapted from [RPM+18a].

2.3.3.

4. *funding_signed*: Bob signs the funding transaction and sends it back to Alice. Alice now broadcasts it after she receives the funding_signed message.

5. *funding_locked*: Sent by both Alice and Bob. The funding transaction is on the blockchain, and after this message, the channel can go into normal operation mode [RPM+18a].

Note that by specifying the hash of the genesis block, see Subsection 2.4.1, of the desired blockchain, Lightning Network can function as an off-chain payment layer for several cryptocurrencies. Interestingly, the maximum amount possible to put into a channel is 0.16777216 BTC [RPM+18a]. This amount is a temporary limit.

### 4.4.2 Channel closing

Before closing a channel, no new HTLCs should be accepted, and the already existing HTLCs should be executed [RPM+18a]. If a party disconnects during the shutdown, the negotiation of fees will be renegotiated after the disconnected party reconnects. Figure 4.5 shows the messages exchanged during the closing of a channel.

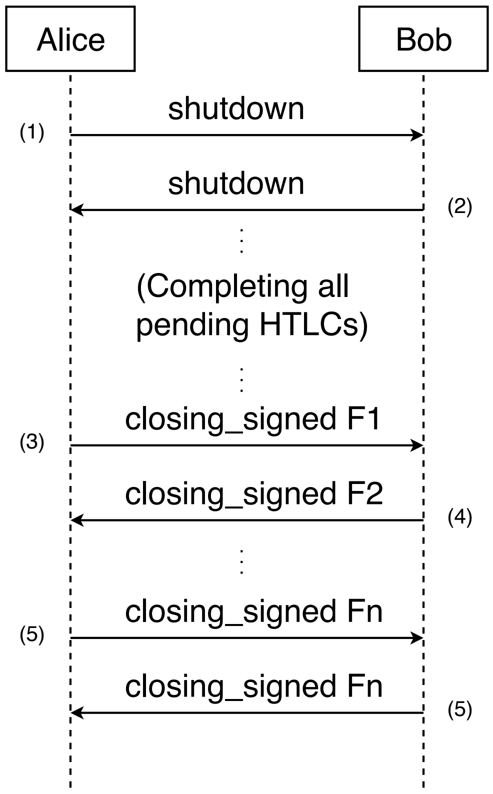

**Figure 4.5:** Closing a channel. Adapted from [RPM+18a].

1. *shutdown*: This message initiates the closing of a channel. If the channel is clean, without any pending HTLCs or changes, the current commitment transaction is broadcast. The *scriptPubKey* script is included in this message to enable the payment to the node.

2. *shutdown*: Bob answers with a shutdown if the channel is clear and ready to be settled.

3. *closing_signed F1*: Alice sends a closing_signed F1 once the channel is clean to negotiate the transaction fee, signed by the *scriptPubKey* from the *shutdown* message.

4. *closing_signed F2*: If Bob disagrees on the transaction fee, he will respond with an updated transaction fee and his *scriptPubKey* used in the *shutdown* message.

5. *closing_signed Fn*: The negotiation continues to either both of them agree or if one of the nodes disconnects. When they agree, the commitment transaction will be published using the *scriptPubKey* form the shutdown messages.

### 4.4.3 Normal operation

For normal operations, updates are sent in batches, as shown in Figure 4.6 where (1), (2) and (3) is one batch. After the updates, Alice signs a new commitment transaction that belongs to Bob, with the updates for Bob, and sends it as shown in (4). Firstly, Bob replies with a *revoke_and_ack* that he agrees on the changes in his commitment transaction. The *revoke_and_ack* is revoking the previous commitment transaction published by Bob and acknowledging that he indeed applied the changes. Then Bob sends Alice her version of the commitment transaction (6). Alice sends back an acknowledgment that she also applied the changes and revoked her previous commitment transaction (7).

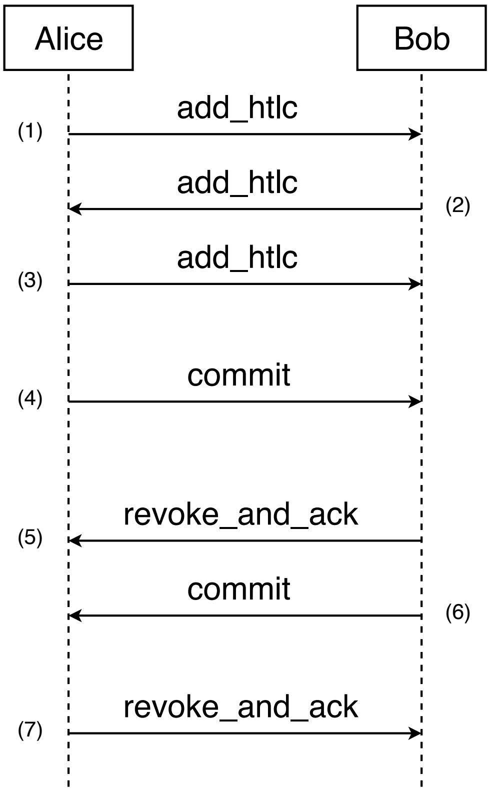

**Figure 4.6:** Normal operations of a channel. Adapted from [RPM$^+$18a].

### 4.4.4 Transactions and HTLC

There are five transactions in total in the evolved Lightning Network. Funding, Commitment, HTLC-Timeout, HTLC-Success and Closing transaction are the transactions described by the RFC [RPM+18b]. The Delivery, Revocable Delivery, and Breach Remedy transactions are no longer defined.

In Figure 4.7 we can see how three of the transactions are working together. Note that after SegWit was implemented, the witness stack is used to check the conditions of a claim instead of a *scriptSig*, as explained in Subsection 2.3.1. Of all the transactions shown in the figure, the funding transaction is the only one that is on the blockchain. In order to close the channel, all pending HTLCs will be executed. Then a closing transaction will be published onto the blockchain.

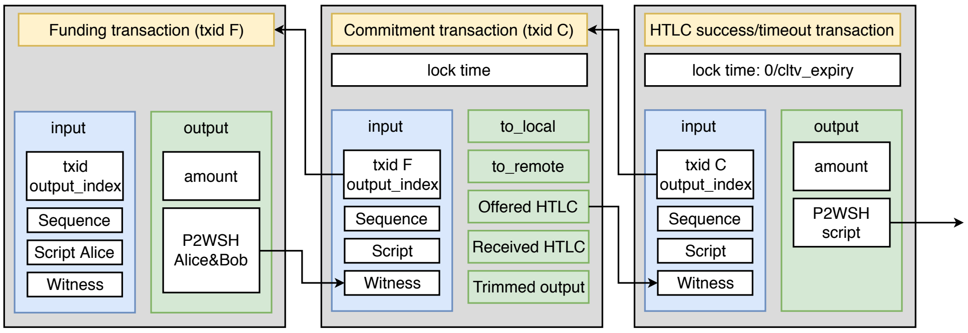

**Figure 4.7:** A sequence where a channel has been established by the funding transaction. An update is done, by the commitment transaction. The channel have forwarded an HTLC, which is not redeemed yet.

## 4.5 Challenges

One drawback with Lightning Network is when one node is not cooperative. As mentioned in Subsection 4.2, Lightning Network uses time locks to prevent funds from being locked up in a channel. Even with the time locks, with an uncooperative node, some funds will be locked up for some time, until the time lock expires. An uncooperative node is not the biggest problem, but if malicious nodes collude against an honest node, they can prevent it from transferring funds. This attack can be carried out by locking up all the channels, initially by offering channels with a long time lock. Therefore it is recommended only to accept reasonable time locks.

For an honest node to punish a dishonest node, the honest node must monitor the blockchain. Once the dishonest node publishes a revoked commitment transaction on the blockchain, the honest node must broadcast a breach remedy transaction. Recall that there is a time lock for the party who broadcast the commitment transaction, see Section 4.2. The honest node must broadcast the breach remedy transaction before this time lock runs out to punish the dishonest node.

Monitoring the blockchain could be a bit demanding for lightweight nodes, SPV. A solution could be to use a third party watching for a revoked transaction on the blockchain. In this case, they will alert the victim, and charge a fee. Since the punishment is to take all funds in the channel, the honest node will most likely profit from having a third party monitoring for it.

Today, Lightning Network is only suitable for micropayments. As mentioned in Subsection 4.4.1, temporarily, the maximum funds possible to put in a channel is 0.16777216 BTC. Another factor is that the Lightning Network is under development still, so it is not safe to send large amounts of Bitcoin over the Lightning Network. If a user wants to execute a transaction of significant value, Bitcoin provides better security.

Another drawback with Lightning Network is that both the sending and receiving node must be online. As shown in Subsection 4.4.3, both parties are exchanging messages. This exchange would not be possible if either Alice or Bob were offline. However, by a good user interface at the SPVs, it should be possible to notify a user to activate the channel. Whereas Lightning Network is more challenging in countries where there is a limited Internet connection, a Bitcoin transaction only requires an address and the sending party to be online.

## 4.6 Network Topology

Network topology in Lightning Network is a controversial subject. The predicted network topology in Lightning Network is the hub-and-spoke topology, see Figure 4.8[2]. Subsequently, because of the funds required to become a hub is significant, only a few nodes can become a hub. Imagining creating a hub where each of the channels is maximized, 0.16 BTC. The hub makes hundreds of those channels. In order to obtain 1.6 BTC, the node must be financially strong. In this case, the hubs will be few and centralized, because not many actors can acquire the required funds.

As shown in Figure 4.8, which is a picture of the test network, some hubs connect many nodes to the rest of the network through channels. Accordingly, to make a payment through the Lightning Network, it is most efficient to connect to a hub when connecting to the Lightning Network. Connecting to a hub will increase the range of payment, and possibly this hub is connected to the receiving node. Having a topology with hubs could work in a decentralized manner if the hubs were many

[2]https://explorer.acinq.co/ Accessed 14.05.2018

but small. Unfortunately, as shown in Figure 4.8, it tends to be fewer, but more prominent hubs.

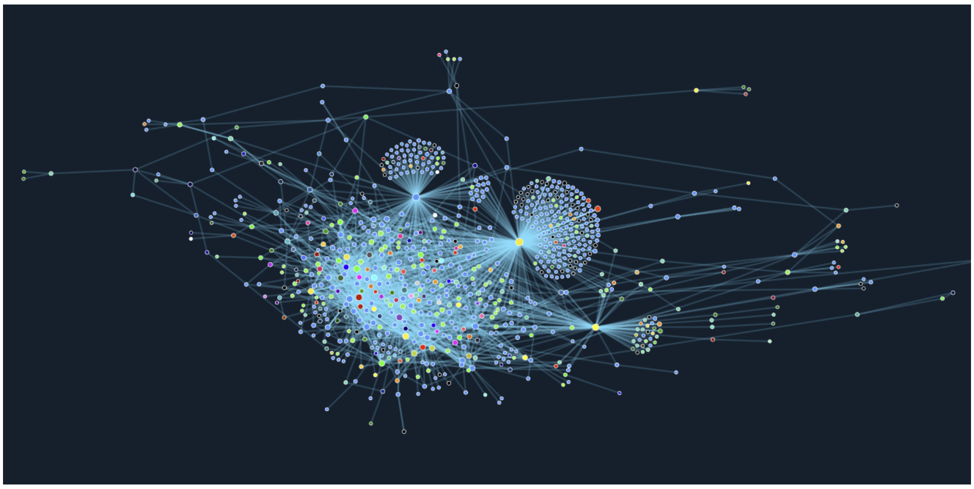

**Figure 4.8:** This picture show the topology of the lightning test network. In the picture, there are some clear hubs, with many spokes connected to it. Snapshot from [ACI18], accessed 14.05.2018

A vulnerability of the hub-and-spoke network with few hubs is that there is a single point of failure. If one of the hubs goes offline, or crashes, some nodes in the network will be without channels to other nodes, and in this case, all of the spokes will have to establish new channels, which will be expensive. As mentioned, Lightning Network requires that the end nodes to be online and to find a path, the hub must also be online. Conditionally, the availability is vital in the Lightning Network with a hub-and-spoke topology. On the contrary, Bitcoin itself is not dependent on online recipients to transfer money. In this case, Lightning Network restricts the availability of Bitcoin. When it is a single point, it is easier for hackers to attack the target, compared to if they want to attack Bitcoin, they have to gain control over most of the network or a big pool.

Another possible topology in Lightning Network is achieved by organic routing [PAN17]. Organic routing will result in a dynamical topology because channels are created on demand. Recall the problem with the barrier to start a hub in the hub-and-spoke topology. Organic routing reduces the barrier of funds needed to run a hub. Additionally, it will be more dynamically, with fewer single points of failure, because channels are made on demand. If Alice wants to pay for her coffee, and she has no connections that share a channel with the coffee shop, she will create a channel herself. In this scenario, Alice will be able to send payments to nodes that have established channels with the coffee shop, see Figure 4.9.

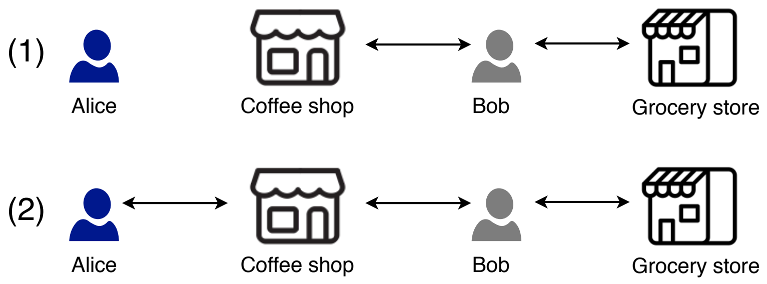

**Figure 4.9:** Here, if Alice establish a channel with the coffee shop, she can also access the grocery store using the channels between the coffee shop, Bob and the grocery store

Organic routing is not without problems either. One problem is that the Lightning Network requires nodes to be active to perform a channel payment. If one of the nodes is offline in the route, the node must set up a direct channel with the recipient instead. Unreliable routes can lead to unnecessary channel creation, resulting in more transactions on the blockchain. Another problem is to find a route where each hop has enough funds in the channel to cover the payment. On the other side, if a node fails the connection, fewer nodes will be affected compared to if a node failure in the hub-and-spoke topology.

### 4.6.1 Flare

There are several other propositions on routing in the Lightning Network. *Flare* [PZS+16] is a routing proposal that uses *global* and *local* beacons. A beacon is a chosen node that will be known for other nodes in the network. The difference between global and local beacons is that global beacons are a set of nodes that are known for all other nodes, see Figure 4.10. For local beacons, the set of beacons may vary between nodes in the network. However, there might be one or more beacons that are linked to another beacon inside the other sets and will function as the connection point. See Figure 4.11.

By having only global beacons, a threat imposes where the attacker may want to create a false node to increase the chance of becoming a beacon. The incentive to be a beacon is to collect fees, or controlling the network. As a solution to that, a dynamical set of beacons is often used. After a given period of time, the beacons will be replaced. On the contrary, establishing new channels with every new beacon is not scalable if the beacons change every day.

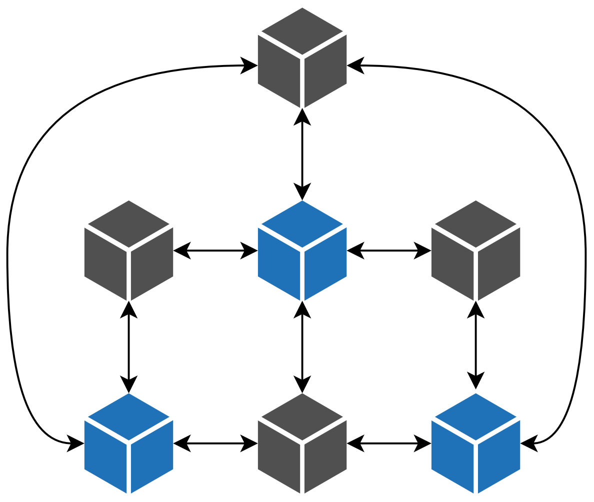

**Figure 4.10:** The beacons, in blue, is connected to each node in the network. The beacons can be selected randomly. Each node in the network will ask where the beacon is, and establish a connection to it.

Flare aims to find possible routes quickly and uses a hybrid between a proactive and reactive collection of information. It proposes a proactive collection of long-term states of the Lightning Network, such as payment channels between nodes. By proactively finding open payment channels, the routing will know which channels to ask for the next step. For short-term and more dynamical information, such as the distribution of funds in a channel and fees, Flare is proposing to use a reactive collection of such information. However, due to the vulnerability of beacons, Flare is not used in the implementation of Lightning Network.

### 4.6.2 Onion Routing

In Section 4.1, privacy was mentioned as one of the requirements for the Lightning Network. If the Lightning Network goes towards the hub-and-spoke topology, big hubs will be able to collect information about transactions, senders, and receivers. Collection of such data will impact the privacy. A suggestion to that problem is to implement onion routing. Camenisch and Lysyanskaya [CL05] have written a paper defining secure onion routing [CL05].

Purposefully, onion routing is used for private communication over a public network. Onion routing encrypts the messages sent between the client and the server multiple times, depending on how many hops on the route there are. An onion router

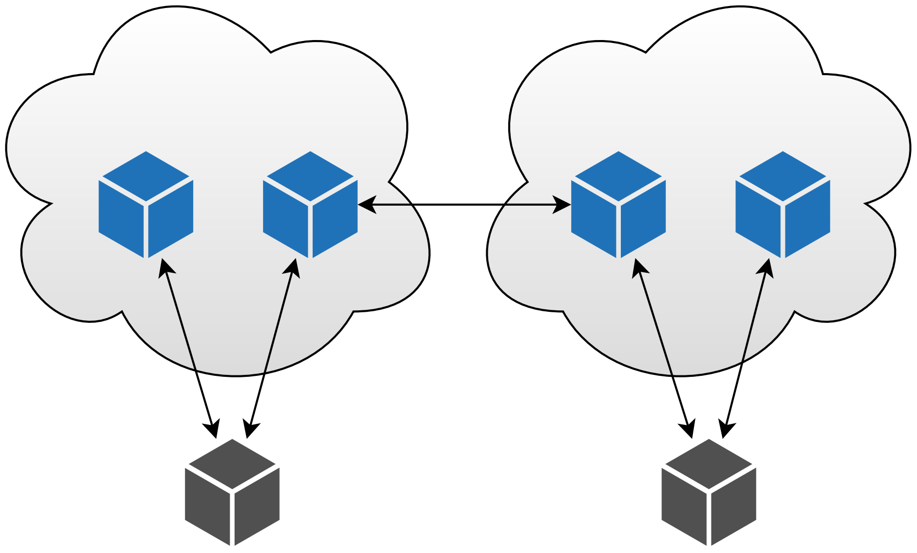

**Figure 4.11:** Two nodes having independent sets of beacons, where one of the beacons in each set is linked to a beacon in the other set. Several nodes can share the same set, but there are several different sets like this in the network.

is a router that removes a layer of encryption before it forwards it to the next onion router. Consequently, an onion router only knows the hop before and the hop after itself. In order for $A$ to communicate with the server $S$, she will route the payments through onion router $B$ and $C$. Public key cryptography is used, so $A$ knows the public key of both $B$ and $C$. Therefore she can encrypt the packet with first $B$'s public key and then $C$'s public key. When the packet arrives at onion router $B$, it will remove one encryption layer. The packet is still encrypted by the public key of $C$, so $B$ is not able to read it. $B$ forwards the packet to $C$, where $C$ is removing the last layer. The packet could be a simple GET request, so $C$ forward the message to the server. For packets from the server to $A$, each onion router will encrypt the packet before forwarding it. This scenario achieves *sender anonymity*. The sender remains anonymous, while the destination is public.

However, simple onion routing like mentioned above can be a bit slow, so to lower the latency, the Lightning Network proposes High-speed Onion Routing at the NETwork layer (HORNET) [CAB$^+$15]. In addition to increase the speed, HORNET provides *sender to receiver anonymity* as well [CAB$^+$15]. *Sender to receiver anonymity* means that both the source and destination is kept anonymous. HORNET uses Sphinx [DG09] packets in order to set up the onion route. Sphinx is a protocol for provably-secure mix [DG09]. HORNET moves the session state from the intermediate nodes, like in ordinary onion networks like The Onion Router (ToR), to the end hosts. The session states are included in the packets so that the node can extract its part when the packet arrives at that node before being forwarded again [CAB$^+$15]. Each node keeps a local secret to encrypt the state per-session data. A header Anonymous Header (AHDR) is created to enable each hop to retrieve their session state without leaking any information about the route and end points. The setup phase uses Diffie-Hellman key exchange for processing [CAB$^+$15]. During one session, nodes will use symmetric keys to decrypt or encrypt the payload, extract their state from the message and process the AHDR.

Note that the Lightning Network aims to implement HORNET over time, but at the moment Lightning Network is based on Sphinx construction [RPM$^+$18c].

## 4.7 Collusion Attack

One weakness that Lightning Network should handle is the collusion attack discovered by Piatkivskyi et al. [PAN17]. Lightning Network assumes that every node along the route is honest, and keeping the preimage $R$ secret until pulling funds. The attack assumes that Lightning Network uses a form of onion routing to provide anonymity. Originally, in the paper of Poon and Dryja [PD16], acknowledging that a transaction is successful is "knowing $R$ is proof of funds sent." [PD16]. However, depending on which end, the transaction is successful either when the HTLC is executed, or when the funds are received. This different interpretation of a successful transaction can be exploited. What is happening if two nodes in the Lightning Network decides to collude against the sender, Alice?

In the scenario where Alice wants to pay Dave, Dave can collude with anyone on the route. If Dave colludes with Bob, he will send the preimage $R$ to Bob after he sends the hash value used for the HTLC. After Alice set up her HTLC with Bob, Bob can execute the contract and redeem his funds, because he already got the preimage $R$ from Dave. Dave can then claim that the transaction failed, and then refuse to send the goods Alice paid for to her. This attack would only work for colluding against the sender, and the Lightning Network does not have any good measure against such attack. Since the acknowledgment of a transaction is ambiguous, Alice will understand this as a successful transaction. While Dave, the cheater, will see this as an unsuccessful transaction and can demand Alice to pay him again.

After a discussion with the Decker [Dec18], Dave will be responsible for this transaction after all. Then, Dave has no incentive to perform such an attack. An executed HTLC means that Dave revealed the preimage $R$. A simplified illustration of the collusion scenario is described in Figure 4.12. In the figure, there are only three nodes, but the principles are the same, that the middle node will get all the payment without forwarding it. Considering this new information, there is no attack on the regular user. However, Piatkivskyi et al. stated that it could impact money laundering activities [PAN17].

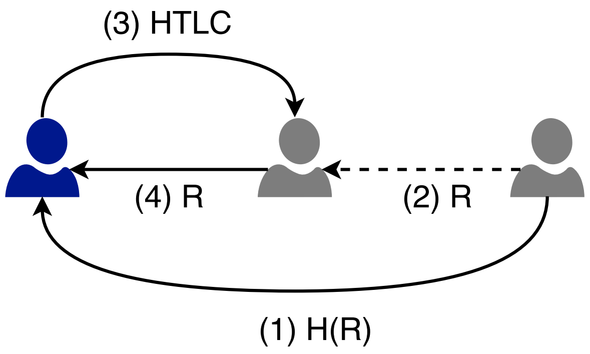

**Figure 4.12:** Collusion attack where the grey nodes colludes against the blue. Taken from [PAN17]

Chapter 5

# Discussion

In the previous chapters, we have investigated the Bitcoin system, bottlenecks for scalability, and the Lightning Network. Will Bitcoin survive in its original form, or must fundamental changes happen before it can be used by millions worldwide? Is the Lightning Network the solution Bitcoin needs, without changing the values of Bitcoin? In this chapter, we will discuss the finding in the previous chapter in a broader perspective. Two future scenarios will be predicted, based on assumptions and data gathered from previous chapters.

## 5.1 Bitcoin survival

**Transaction fee.** By looking at the data when Bitcoin was peaking regarding confirmed transactions per day in December 2017, we showed that both the transaction fee grew. Additionally, the waiting time to confirm a transaction took over 7 hours, see Section 3.1. Is the transaction fee correlated to how big the pool of unconfirmed transaction is?

For this estimate, we chose to sample December 2017, because the block size was around 1 MB, see Section 3.1. By doing so, we want to explore how a congested network is affecting the transaction fee, see Figure 5.1. Using Pearson correlation coefficient, Equation 5.1, estimated with Equation 5.2, the resulting correlation coefficient is 0.58. Figure 5.2 is describing this relation. Thus, the graphs are moderate correlated.

One reason for the moderate correlation could be that the graphs are different in two ways. Firstly, there are two peaks, on the 8th of December on the 22nd of December. It is true for both graphs are at a peak at those dates; however, the peak is varying in size. Secondly, from the 14th of December until the 18th, the graphs are contradicting. Otherwise, the two graphs seem to be correlating through December. In conclusion, if Bitcoin maintains the same throughput, the transaction fee will probably increase in the same manners as transactions broadcast on the network. Thus, Bitcoin should increase the throughput in order to survive.

$$\rho = \frac{\operatorname{cov}(X, Y)}{\sigma_x \sigma_y} \tag{5.1}$$

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{5.2}$$

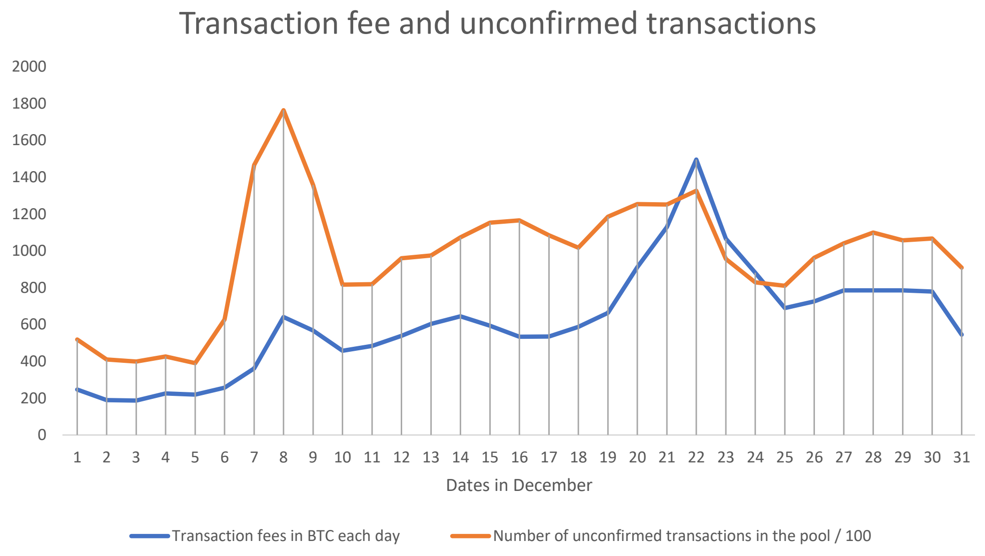

**Figure 5.1:** A chart describing two different graphs. The blue graph is the transaction fee in BTC. The orange graph is the number of transactions in the pool of unconfirmed transactions divided by 100.

**Block size.** Is Bitcoin capable of handling more users, or did the idea outgrow the technology? Chapter 3 explored the solution to this matter. To increase the throughput, increasing the block size is suggested. According to Decker et al. [CDE$^{+}$16], the optimal block size could be increased, but not exceed 4 MB because of the effective throughput of the network. Increasing the block size to 4 MB will allow a throughput of 27 transactions per second at most. A throughput of 27 transactions per second is still low if Bitcoin wants to scale to billions of users. However, it could give Bitcoin the time it needs to find a more mature and scalable solution.

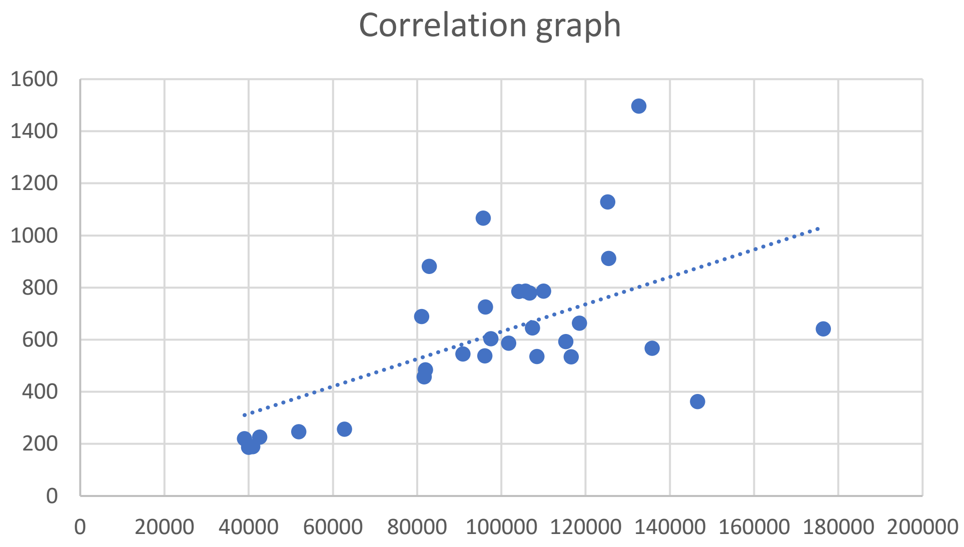

**Figure 5.2:** Correlation graph where $r = 0.58$.

Interestingly, Bitcoin Cash was an attempt at increasing the block size. However, considering their actual block size. According to BitInfoCharts[1], the block size is smaller than the block size of Bitcoin. One reason for this could be that Bitcoin Cash does not have enough users to be utilizing the maximum capacity of a block.

Consequently, increasing the block size will result in a bigger latency through the network than the small blocks, which can cause an increased creation of temporary forks, see Subsection 3.2. Big latency impacts the fairness of mining. By increasing the block size, miners should be connected to a high-speed bandwidth to receive the block faster. Oppositely, increased block propagation time affects the centralization. Big clusters of miners in a geographical area, with low bandwidth, will experience an advantage because they will probably have low latency between each other. BIP 152, compact block relay, as mentioned in Subsection 3.1 is reducing the block latency.

**Energy consumption.** Considering the energy consumption of Bitcoin, *Proof-of-Stake* was suggested as a solution. PoS is a consensus algorithm used by other cryptocurrencies [KRDO17][Lar13][Vas14][Aca18] today. Even though it is getting more acknowledged, PoS is experiencing resistance in the Bitcoin community. A hard fork is needed to change the consensus algorithm. Replacing the consensus algorithm results in a change of the software, and the set of rules nodes are following.

[1]https://bitinfocharts.com/comparison/size-btc-bch.html accessed 25.05.2018

Thus, nodes running the old software will reject blocks built with the new set of rules. It could result in a loss of many active nodes. History shows that such a fork can split the community, leaving with one active and one idle solution, e.g., Bitcoin and Bitcoin Cash, see Section 3.1.

Mining is the primary cause of high power consumption, so to combat the consumption problem, the mining process of Bitcoin should be reviewed. However, having a wealthy mining ecosystem secures the security of the blockchain. Correcting the mining ecosystem could result in a decreased security level of Bitcoin, which could lead to the demise of Bitcoin.

Figure 5.3 shows how the security of Bitcoin impacts the value of the currency. If the security of Bitcoin is deprecated, there is a possibility that the value of the coin itself will decrease. Miners are providing the security of Bitcoin, but once the price of Bitcoin is falling, miners could be affected. Possibly, the energy consumption will make it unprofitable to mine for some miners. Unprofitable mining will, as mentioned, affect the security of Bitcoin. Interestingly, the price of electricity will affect the localization of miners in such a scenario, leading to more centralization in a geographic area.

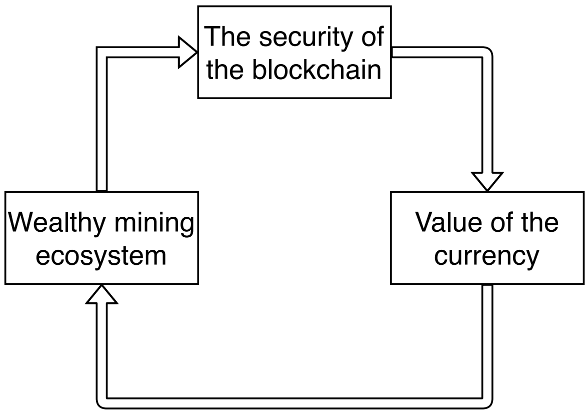

**Figure 5.3:** This figure illustrates the security of Bitcoin. A collapse in the price of Bitcoin will eventually impact on the miners, which will again impact the security of Bitcoin. Inspired by Narayanan et al. [NBF+16].

## 5.2 Lightning Network evaluation

Initially, with off-chain micropayment solutions, there is no immediate need to increase the block size itself. Micropayments channels can increase the total throughput by several thousand transactions per second. The Lightning Network can even provide instant payment. What are the consequences of implementing Lightning Network onto Bitcoin?

How the Lightning Network is going to impact Bitcoin depends on several issues. Firstly, what kind of routing used in the Lightning Network impacts the decentralization. In Section 4.6 we discovered the topology of the Lightning Network, along with some routing protocols. The onion routing in the Lightning Network uses Sphinx [DG09] for the message format. Because of decentralization is an essential value of Bitcoin, the developers of the Lightning Network is trying to avoid centralization, even if it is less efficient. A gossip protocol is used for the awareness of the topology. Routing is done by the sending node, based on the local view of the network surrounding it. Flare is not used, because beacons, as mentioned in Section 4.6, can be a single point of failure, and they also contribute to centralization [Dec18].

Is the Lightning Network solving the energy consumption problem of Bitcoin? Two ways to reduce the power consumption is:

1. Mining hardware must increase their efficiency significantly.
2. By a reduction of the total hash rate of the network.

A reduction of the total hash rate means reducing the number of miners, which will affect the security of the blockchain. Even though, the Lightning Network is not capable of doing either of those, unless it makes the number of transaction off-chain large enough for miners to be mining empty blocks. Empty blocks could lead to unprofitable mining in the future, similar to Scenario 1 in Section 5.3. However, the miners are currently receiving the coinbase transaction. Most likely, the power consumption of Bitcoin will be unaffected by successful deployment of the Lightning Network.

From the beginning, one of the primary goals of Bitcoin was to be a decentralized ledger. What is the purpose of being decentralized? No government can intervene, one is no longer dependent on central actors like a central bank, which is less trusted after the financial crisis of 2007-2008 on a world basis [vEPvR17]

If the Lightning Network topology goes towards a hub-and-spoke topology, it will affect the decentralization of Bitcoin. As mentioned in Subsection 4.6, the test network of Lightning Network contain a mixed network also consisting of few big hubs. However, this is just a test network, and it will be easier for the user to connect to one of the already known hubs. By having big hubs, there could be central entities that can affect the prioritization of the payments, the availability of the network, censorship and even collection of data breaking anonymity. The latter statement is only valid if there is no onion routing. One of the main components of Bitcoin is that it is a decentralized platform. Once it gets centralized, there will be other services that can provide more efficient and suitable solutions, like the Visa network for payments.

## 5.3 Scenario 1 - When all bitcoins are mined

In this scenario, we assume that all the coins are mined. When all coins are mined, there will no longer be a coinbase transaction reward for the miner. Satoshi Nakamoto suggested that the transaction fee will be the new incentive to mine, validate and create new blocks. How will this affect the mining ecosystem?

In December 2017, the average number of transaction per block was 2225, see Section 3.1. At this time, the coinbase transaction is worth 12.5 BTC. In order to give a simplified estimate of the future transaction fee, it is possible to divide the coinbase transaction reward by the average transactions included in a block. Hence, we will find out how much a single transaction must compensate for the lost coinbase transaction reward.

$$\frac{\text{coinbase reward}}{\text{average transactions in a block}} \cdot 1 \text{ BTC} = (\frac{12.5}{2225}) \cdot 1 = 0.005618\text{BTC} \qquad (5.3)$$

As a result of the estimate, each transaction will have to include a transaction fee worth 0.005618 BTC. This estimate is only covering the coinbase transaction reward, and not the additional transaction fee. When 1 BTC was worth 7547.94 USD[2], the transaction fee calculated in Equation 5.3 would be 42.4 USD. This is quite high for one single transaction. Consequently, there is a possibility that miners will stop mining when the coinbase transaction fades out. A reason for this could be that mining will become unprofitable because of electrical expenses can surpass the reward. An other reason could be that the transaction fee will be too high for people to do a transfer. Thus, people will stop using Bitcoin, and miners will not have any incentive to mine.

The evolution of mining hardware resulted in an efficient outcome, the ASIC. Because of Moore's law, the future could bring even more efficient mining hardware. This evolution will contribute to an increase in the total hash rate. Consequently, the difficulty of solving a puzzle will increase. It is possible that big companies will take over the mining of Bitcoin entirely, producing ASICs for using internally in the company. Cautiously, this is a violation of two of Bitcoins core values, being a trustless and decentralized currency. Therefore it can be in the best interest of the company to keep Bitcoin mining decentralized due to the ecosystem described in Section 5.1.

[2]https://coinmarketcap.com/ accessed 26.05.2018 19.44 UTC+2

## 5.4 Scenario 2 - The operational Lighting Network

This scenario is building on the previous scenario. We presume that the Lightning Network is now implemented on top Bitcoin. The topology is organic, and onion routing is being used to send a payment through the network of channels. The Lightning Network still runs on the Bitcoin blockchain. Payments are now instant in time, and the Bitcoin blockchain is no longer the bottleneck.

Transaction fees are now the only incentive miners have to mine new blocks and maintain the security of Bitcoin. How will the Lightning Network impact the transaction fee for publishing a transaction onto the blockchain?

As seen in Scenario 1, see Section 5.3, when the coinbase transaction no longer gives an incentive to mine, transaction fees alone will be the incentive. Because of the increased transaction fees, off-chain payments will be economical for the users. By using the Lightning Network, only two transactions—the funding transaction and the closing transaction—is published on the blockchain, see Section 4.2. Because they are published on-chain, the transactions will include the transaction fee. By having fewer transactions on the blockchain, the pool of unconfirmed transaction will decrease. Thereby the additional transaction fee will decreased as described in Section 5.1. Note that coinbase reward replacing fee will still apply in this scenario.

It is difficult to predict how significant the reduction of the pool will be, but a channel is supposed to last a long time. The Lightning Network can scale Bitcoin up to 35 million users based on an assumption made by Poon and Dryja [PD16]. Poon and Dryja are presuming that the average Lightning Network user will publish three transactions on the blockchain per year. Equation 5.4 shows this estimate based on the assumption made by Poon and Dryja.

$$\frac{\text{transaction per second}}{\text{3 trans per year}} = \frac{3.33}{9.51\text{e}{-8}} = 35 \text{ million} \tag{5.4}$$

To scale Bitcoin even further, Poon and Dryja suggest increasing the block size. If it is increased to 4 MB as Croman et al. suggests [CDE$^+$16], Bitcoin with the Lightning Network will be able to handle 140 million users. This estimate assumes that the transaction size is 500 bytes and that each block includes 2000 transactions. As shown in Section 3.1, during the peak in December 2017, a block included on average 2225 transactions, which increases the total amount of users to 38.9 million, see Equation 5.5. Poon and Dryja further say that to scale up to 7 billion users, the block size must increase to 133 MB if using the Lightning Network [PD16]. However, increasing the block size to 133 MB could be troubling regarding network latency, see Section 5.1. Another consequence is that the blockchain will be growing rapidly in size, see Section 3.1. After a conversation with one of the developers of the Lightning Network, this estimate was concluded to be outdated [Dec18]. It is hard to give an estimation of the scalability because the Lightning Network is still under development. However, the estimate is included to illustrate the scalability of Lightning Network, even though it is based on an outdated number.

Increasing the block size to 133 MB at once could impact the mining ecosystem. Miners will mine many empty blocks. If this happens when the incentive of the coinbase transaction ends, miners will probably shut down their operations, because it will not be profitable. Without filling the blocks with transactions, there will be limited transaction fees. If the block size utilized is under Bitcoin's capacity after the deployment of the Lightning Network, the Lightning Network should not force a significant block size increase. Significant changes can lead to unexpected consequences.

$$\frac{\text{transaction per second}}{\text{3 trans per year}} = \frac{3.70}{9.51\text{e}{-8}} = 38.89 \text{ million} \tag{5.5}$$

Poon and Dryja recommend only to send micropayments through the Lightning Network. For doing more significant transfers, one should still be sending the transaction on the Bitcoin network. The estimates in Equation 5.4 and Equation 5.5 is not considering this fact.

During December 2017 Bitcoin had an average of 935470 unique Bitcoin addresses[3]. Comparing 935470 unique addresses to 38.9 million users shows that the Lightning Network will indeed enable Bitcoin to scale. The total amount of Bitcoin addresses could be used for this comparison as well. However, using active unique addresses gives a number of active users. Active users are required for Lightning Network channels to be operative. Using active unique addresses for estimating users could be faulty as well. Some addresses are just being used to obtain bitcoins for holding onto them. Addresses used only for holding funds could work as a node in the Lightning Network. According to the same site, BitInfoCharts[4], 5,118,631 addresses possess 100 dollar worth of bitcoins. 100 dollar will be enough to create channels and be a part of the Lightning Network, but not all of these addresses are active. Note that one user can create as many Bitcoin addresses as the user desires.

Interestingly, the increase in block size will have to happen through a hard fork to increase the throughput. Gradually increasing the block size will result in numerous hard forks. On the one hand, the hard forks can also be used to implement pending BIPs, which lower the barrier writing a BIP that requires a hard fork. On the

[3]https://bitinfocharts.com/comparison/bitcoin-activeaddresses.html#6m accessed 28.05.2018
[4]https://bitinfocharts.com/bitcoin/ accessed 28.05.2018

other hand, many hard forks can be unfortunate for the mining environment as well, because it requires all nodes to update the software often.

Poon and Dryja are presuming a hub-and-spoke topology [PD16], which will lead to centralization. Consequently, by using organic topology, more transaction will go on-chain. By more on-chain transactions, the estimates done by Poon and Dryja will be insufficient. By increasing the transactions published onto the blockchain to five per year, Bitcoin will scale up to 23 million users. Note that no data were found to support this estimate if using organic routing. Organic topology will be more expensive because more channels will be created, and most likely restrict the scalability possibilities for Bitcoin. On the contrary, organic topology allows Bitcoin to keep its decentralized value. Is it worth the price?

## 5.5 Centralization vs Decentralization

What is the point of an energy consuming, expensive solution that can barely scale, if it is turning centralized in the end?

In Bitcoin, the security is no longer dependent on one entity, i.e., the bank, but rather the system as a whole. For an attacker to attack Bitcoin, he has to gain control of 51% of the total hash rate of the network. Once the Bitcoin network gets more centralized, regarding mining power, it can no longer provide the same security as if the network was completely decentralized. We discovered in Subsection 3.3.2. With 43.3% Bitmain owns a significant portion of the hash rate in the Bitcoin network. It is alarming if Bitmain's pools get more than 50% hash rate. The main concern about centralized mining power is the 51% attack as mentioned in Subsection 3.3.1. When reaching 51% of the total hash rate, the pool manager can decide what blocks the miner should mine on. The pool manager can favor and censor transactions form specific Bitcoin addresses.

A high hash rate distribution is needed to provide strong security. On the contrary, if the mining power is centralized, this will impact the soundness of the Bitcoin blockchain because of the mentioned reasons above.

One of the core values of Bitcoin is to have a trustless and decentralized system, as explored in Chapter 2. Bitcoin is the first of its kind that made a successful currency based on these values. Eventually, Bitcoin has to face the scalability challenges if it wants to be the currency for everyone. Several of the suggested solutions tend to move towards a more centralized system. Even today, without any of those solutions implemented, Bitcoin turns more centralized regarding the mining process.

Is the currency still effective if it gets centralized? Bitcoin will without any doubt be more effective if it was centralized, but the security of Bitcoin decreases for the following reasons:

1. First of all, with centralized mining, Bitcoin can suffer from double-spend attack, see Subsection 2.3.3, as a consequence of the 51% attack.
2. Secondly, if companies owning the largest pools collude, once they gain 51% they can censor targeted transaction by not including them in a block in the mining process.
3. Thirdly, if the mining companies origin from the same country, there is a possibility of governmental interference.

All the mentioned factors are why Bitcoin should be decentralized, even when it comes to mining, or else there are better payment solutions on the market which are centralized.

How will the Lightning Network affect decentralization? As mentioned in Chapter 4, it depends on the topology of the Lightning Network. Hub-and-spoke topology will lead to a centralized Lightning Network, working as a banking system. Poon and Dryja state the following in the original paper: “Eventually, with optimizations, the network will look a lot like the correspondent banking network, or Tier-1 ISPs” [PD16]. However, work on other routing protocols is in progress. After looking at the different ongoing Lightning Network projects, the routing has been solved in a different way than suggested in the original paper. The development is considering the problems with centralization, and therefore moving towards a more decentralized solution.

## 5.6 GDPR

Because Bitcoin is a decentralized distributed system, where the data appended on the blockchain is permanent by nature, how General Data Protection Regulation (GDPR) should handle it is uncertain.

Bitcoin’s blockchain has been used for storing illegal information [MHH+18]. Once published on the blockchain, this data can not be removed. Transactions published on the blockchain defines as private data because of the incomplete anonymity of the users in Bitcoin. The Lightning Network could solve the problem considering the anonymity of users, because onion routing will anonymize the data, keeping it out of the scope of GDPR. However, the same should be done for Bitcoin to be compliant with the new regulation. The Lightning Network itself is compliant to GDPR because all the states stored in a channel will be discarded. It is only the funding and the closing transaction that will be published on the Bitcoin blockchain.

On the other hand, there is a chance that blockchain technology could be an exception. Article 2 (c) [EU16b] describes that household activities are out of the scope of the GDPR. On the one hand, one could argue that running a Bitcoin node, and mining is household activities. Maybe it can even help to regulate big company mining, which will help to keep Bitcoin decentralized. On the other hand, according to Recital 18 [EU16a], the regulation applies to "[...] controllers or processors which provide the means for processing personal data for such personal or household activity." [EU16a]. This statement could be interpreted in the way that running a full node is out of the scope of the regulation, while processing transaction and blocks that are containing personal data, mining, would be affected by the regulation.

However, by anonymizing the data, the Bitcoin blockchain and the Lightning Network can be compliant with GDPR.

Chapter 6

# Conclusion

In the introduction, the research question for this thesis was stated.

**RQ1: Can Bitcoin survive in its original form?**

We have discovered that Bitcoin suffers from scalability problems. The solutions that addresses problems like the block size and the energy consumption will radically change Bitcoin, it will no longer be in its original form. As explained in Subsection 2.4.3, the Bitcoin community is conservative when it comes to changing the protocol radically. Therefore, we explored an exciting solution regarding off-chain payments. The Lightning Network aims to solve the scalability problems of Bitcoin. Initially, the Lightning Network will not change the original form of Bitcoin, but as the user base increases, some changes regarding the block size might be necessary. Without the Lightning Network, Bitcoin will have to change one of its fundamental elements to survive.

To answer **RQ1**, we stated three sub-questions:

**SQ1: What is the original form of Bitcoin?**

**SQ2: What are the scalability problems of Bitcoin?**

**SQ3: What is the Lightning Network, and how does it address the scalability problems?**

In Chapter 2 we explained some of the fundamentals of Bitcoin, answering **SQ1**. This chapter review the block size, blockchain and the underlying transaction used for off-chain solutions. We found that decentralization and trustlessness are two of the essential values of Bitcoin in Chapter 3. In Chapter 3 we explored the known problems of Bitcoin, (**SQ2**), and used December 2017 as a source of data when calculating estimates. We showed that with a congested network, Bitcoin had a throughput of 3.7 transactions per second. This throughput is below the maximum capacity of Bitcoin. Nonetheless, the throughput must increase to scale Bitcoin.

Off-chain solutions is a scaling proposition that avoids affecting the essentials of Bitcoin. The Lightning Network was explored in depth in Chapter 4. We found that the Lightning Network will be able to scale Bitcoin by making payments off-chain, answering the final sub-question, **SQ3**. Eventually, the Lightning Network will need Bitcoin to change the block size if the user base continues to grow. There are particular challenges regarding conserving the decentralization in Bitcoin when implementing the Lightning Network. We also discussed how decentralized Bitcoin is throughout this thesis.

The Lightning Network is online. However, there is still development going on, so users are recommended to try out the test network before putting funds into the real network. As mentioned in Chapter 4 there are three ongoing projects developing the Lightning Network, *lightning-c*, *lnd* and *eclair*.

## 6.1 Limitations and future research

In this thesis, we only covered the theoretical part of Lightning Network because of the limited scope. Further research could consider *eltoo* as an update mechanism for the Lightning Network channels. Implementing the Lightning Network test network would be a natural next step for this thesis. Following the development and deployment of the online Lightning Network would also be interesting, and collecting data continuously during the early stages. To be able to find the right estimates of the scaling regarding the Lightning Network, collecting data on the number of transactions published on the blockchain will be helpful.

Eventually, the Lightning Network will need Bitcoin to change its original form. Exploring the consequences of increasing the block size would be interesting. By surveying the miners, research can predict the outcome of a hard fork, easing the decision of the Bitcoin developer community. Regarding the Bitcoin blockchain, it would be interesting to see if blockchain technology will be affected by the GDPR.

# References

[Aca18] Lisk Academy. What is delegated proof of stake?, 2018.

[ACI18] ACINQ. Test net explorer for lightning network, May 2018. https://explorer.acinq.co/ accessed 14.05.2018.

[BBSU12] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better - how to make bitcoin a better currency. In Angelos D. Keromytis, editor, *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, volume 7397 of *Lecture Notes in Computer Science*, pages 399–414. Springer, 2012.

[CAB$^+$15] Chen Chen, Daniele Enrico Asoni, David Barrera, George Danezis, and Adrian Perrig. HORNET: high-speed onion routing at the network layer. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1441–1454. ACM, 2015.

[CDE$^+$16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

[CL05] Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2005.

[com14] Nxt community. Nxt whitepaper. *Self-published paper, July*, 2014.

[Cor16] Matt Corallo. Bip 152: Compact block relay. GitHub, April 2016. https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki accessed 01.06.2018.

[Dec18] Christian Decker. Personal communication over IRC. #lightning-dev, freenode.net, May 2018. https://botbot.me/freenode/lightning-dev/.

[Dev17] Bitcoin Core Developers. Bitcoin fibre. GitHub, August 2017. https://github.com/bitcoinfibre/bitcoinfibre accessed 11.06.2018.

[DG09] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *30th IEEE Symposium on Security and Privacy (S&P 2009), 17-20 May 2009, Oakland, California, USA*, pages 269–282. IEEE Computer Society, 2009.

[Dig] Digiconomist. Bitcoin energy consumption index. https://digiconomist.net/bitcoin-energy-consumption#assumptions. Accessed: 24.04.2018.

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[DW13] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento, Italy, September 9-11, 2013, Proceedings*, pages 1–10. IEEE, 2013.

[DW14] Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and mtgox. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II*, volume 8713 of *Lecture Notes in Computer Science*, pages 313–326. Springer, 2014.

[DW15] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In Andrzej Pelc and Alexander A. Schwarzmann, editors, *Stabilization, Safety, and Security of Distributed Systems - 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings*, volume 9212 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015.

[EU16a] EU. Recital 18: Not applicable to personal or household activities* (*unofficial description). Recital of the GDPR, April 2016. https://gdpr-info.eu/recitals/no-18/.

[EU16b] EU. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). Legislation from the EU, April 2016. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN accessed 02.06.2018.

[FBDk15] Mark Friedenbach, BtcDrak, Nicolas Dorier, and kinoshitajona. Bip68: Relative lock-time using consensus-enforced sequence numbers. GitHub, May 2015. https://github.com/bitcoin/bips/blob/master/bip-0068.mediawiki accessed 05.06.2018.

[FBL15] Mark Friedenbach, BtcDrak, and Eric Lombrozo. Bip112: Checksequenceverify. GitHub, August 2015. https://github.com/bitcoin/bips/blob/master/bip-0068.mediawiki accessed 05.06.2018.

[HS91] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *J. Cryptology*, 3(2):99–111, 1991.

[IEA] IEA. Norway - energy system overview. https://www.iea.org/media/countries/Norway.pdf. Accessed: 24.04.2018.

[KN12] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. White Paper, August 2012. https://peercoin.net/assets/paper/peercoin-paper.pdf accessed 20.05.2018.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388. Springer, 2017.

[Lar13] Daniel Larimer. Transactions as proof-of-stake, 2013.

[LJW15] Eric Lombrozo, Lau Johnson, and Pieter Wuille. Bip141: Segregated witness. GitHub, December 2015. https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki accessed 02.06.2018.

[Mer87] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.

[MHH$^+$18] Roman Matzutt, Jens Hiller, Martin Henze, Jan Henrik Ziegeldorf, Dirk Müllmann, Oliver Hohlfeld, and Klaus Wehrle. A quantitative analysis of the impact of arbitrary blockchain content on bitcoin. In *Proceedings of the 22nd International Conference on Financial Cryptography and Data Security (FC). Springer*, 2018.

[MMSH16] Patrick McCorry, Malte Möser, Siamak Fayyaz Shahandashti, and Feng Hao. Towards bitcoin payment networks. In Joseph K. Liu and Ron Steinfeld, editors, *Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I*, volume 9722 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2016.

[NBF$^+$16] Arvind Narayanan, Joseph Bonneau, Edward W. Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies - A Comprehensive Introduction.* Princeton University Press, 2016.

[OM14] K. J. O'Dwyer and D. Malone. Bitcoin mining and its energy footprint. In *25th IET Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, pages 280–285, June 2014.

[OO91] Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the fiat-shamir scheme. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739 of *Lecture Notes in Computer Science*, pages 139–148. Springer, 1991.

[PAN17] Dmytro Piatkivskyi, Stefan Axelsson, and Mariusz Nowostawski. Digital forensic implications of collusion attacks on the lightning network. In Gilbert L. Peterson and Sujeet Shenoi, editors, *Advances in Digital Forensics XIII - 13th IFIP WG 11.9 International Conference, Orlando, FL, USA, January 30 - February 1, 2017, Revised Selected Papers*, volume 511 of *IFIP Advances in Information and Communication Technology*, pages 133–147. Springer, 2017.

[PD16] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *Draft Version 0.5.9.2*, January 2016.

[PZS+16] Pavel Pridhodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy, and Olaoluwa Osuntokun. Flare: An approach to routing in lightning network. *Version 1.0*, July 2016.

[RAM+18] Rusty Russel, Shannon Appelcline, Landon Mutch, Christian Decker, Olaoluwa Osuntokun, Christopher Jämthagen, and dlogemann. Lightning network in-progess secifications. GitHub, May 2018. https://github.com/lightningnetwork/lightning-rfc accessed 03.06.2018.

[Riz17] Peter R. Rizun. Towards massive on-chain scaling: Presenting our block propagation results with xthin. Medium, May 2017. https://medium.com/@peter_r/towards-massive-on-chain-scaling-presenting-our-block-propagation-results-with-xthin-da54e55dc0 accessed 11.06.2018.

[RPM+18a] Rusy Russel, Pierre-Marie Padiou, Landon Mutch, Olaoluwa Osuntokun, Christian Decker, practicalswift, Otto Allmendinger, Cristopher Jämthagen, Shannon Appelcline, EmelyanenkoK, ZmnSCPxj, yuntai, and Janus Troelsen. Bolt #2: Peer protocol for channel management. GitHub, May 2018. https://github.com/lightningnetwork/lightning-rfc/blob/master/02-peer-protocol.md accessed 03.06.2018.

[RPM+18b] Rusy Russel, Pierre-Marie Padiou, Landon Mutch, Olaoluwa Osuntokun, Christian Decker, practicalswift, Otto Allmendinger, Cristopher Jämthagen, Shannon Appelcline, Jim Posen, Fabrice Drouin, Matt Corallo, Li Xuanji, nayuta ueno, William Casarin, fivepiece, and Andrew Samokhvalov. Bolt #3: Bitcoin transaction and script formats. GitHub, April 2018. https://github.com/lightningnetwork/lightning-rfc/blob/master/03-transactions.md accessed 03.06.2018.

[RPM$^+$18c] Rusy Russel, Pierre-Marie Padiou, Landon Mutch, Olaoluwa Osuntokun, Christian Decker, practicalswift, Cristopher Jämthagen, Shannon Appelcline, Jim Posen, Fabrice Drouin, nayuta ueno, and Samuel Dobson. Bolt #4: Onion routing protocol. GitHub, May 2018. https://github.com/lightningnetwork/lightning-rfc/blob/master/04-onion-routing.md accessed 04.06.2018.

[Rus15] Rusty Russell. Reaching the ground with lightning. draft 0.2, November 2015. https://peercoin.net/assets/paper/peercoin-paper.pdf accessed 20.05.2018.

[S.A18a] Blockchain Luxembourg S.A. Bitcoin blockchain explorer, May 2018. https://blockchain.info/charts/ accessed 20.05.2018.

[S.A18b] Blockchain Luxembourg S.A. Bitcoin hashrate distribution, May 2018. https://blockchain.info/pools?timespan=4days accessed 20.05.2018.

[Sta14] William Stallings. *Cryptography and network security - principles and practice (6. ed.).* Pearson Education, 2014.

[Vas14] Pavel Vasin. Blackcoin's proof-of-stake protocol v2. *URL: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper. pdf*, 2014.

[vEPvR17] Pauline W.J. van Esterik-Plasmeijer and W. Fred van Raaij. Banking system trust, bank trust, and bank loyalty. *International Journal of Bank Marketing*, 35(1):97–111, 2017.

[Vra17] Harald Vranken. Sustainability of bitcoin and blockchains. *Current Opinion in Environmental Sustainability*, 28:1 – 9, 2017. Sustainability governance.

[WL15] Luqin Wang and Yong Liu. Exploring miner evolution in bitcoin network. In Jelena Mirkovic and Yong Liu, editors, *Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings*, volume 8995 of *Lecture Notes in Computer Science*, pages 290–302. Springer, 2015.

[Wui14] Pieter Wuille. Bip62: Dealing with malleability. GitHub, March 2014. https://github.com/bitcoin/bips/blob/master/bip-0062.mediawiki accessed 02.06.2018.