FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Payment Channel Network Analysis with Focus on Lightning Network

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Peter Holzer, BSc
Matrikelnummer 01425797

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Matteo Maffei
Mitwirkung: Dr. Bernhard Haslhofer

Wien, 5. April 2020

Peter Holzer

Matteo Maffei

FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Payment Channel Network Analysis with Focus on Lightning Network

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Peter Holzer, BSc
Registration Number 01425797

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Matteo Maffei
Assistance: Dr. Bernhard Haslhofer

Vienna, 5th April, 2020

Peter Holzer Matteo Maffei

Erklärung zur Verfassung der Arbeit

Peter Holzer, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 5. April 2020

Peter Holzer

Danksagung

Einen besonderen Dank möchte ich dem Austrian Institute of Technology (AIT), insbesondere Bernhard Haslhofer aussprechen. Unter seiner stets kompetenten und zeitnahen Betreuung konnte ich die Diplomarbeit effizient mit bedeutungsvollen Ergebnissen abschließen. Auch weitere Kollegen am AIT haben mich mit ihren Meinungen und Ideen zielführend unterstützt. Die große Hilfsbereitschaft unter den Kollegen am AIT war großartig!

Weiters möchte ich mich auch herzlich bei meinem Betreuer an der Technischen Universität Wien, Matteo Maffei, bedanken. Bei den regelmäßigen Treffen während der Diplomarbeit hat er mich jederzeit mit äußerst wertvollen Beiträgen und Ideen unterstützt. Auch am Ende der Diplomarbeit war er maßgeblich daran beteiligt, dass ich die Arbeit innerhalb meiner persönlich gesetzten Fristen abschließen konnte.

Darüber hinaus möchte ich Margit Außerhofer danken, die meine Arbeit in Bezug auf die geschriebene Sprache sehr kurzfristig und intensiv Korrekturgelesen hat.

Zuletzt möchte ich meiner Freundin danken, die mich während meines gesamten Studiums immer unterstützt hat und auch mit weniger zur Verfügung stehenden Freizeit meinerseits verständnisvoll umgegangen ist. Weiters hat mir auch meine Familie in jeder notwendigen Situation stets unter die Arme gegriffen - vielen Dank!

Acknowledgements

I want to thank the Austrian Institute of Technology (AIT), especially Bernhard Haslhofer who supervised my during my time at AIT as a colleague of mine. Under his proficient and supporting supervision, I could accomplish this master thesis very efficiently and also with meaningful results. Further, I want to thank additional colleagues at AIT, who supported me very effectively with their inputs and ideas. The enormous collaboration and cooperativeness among the colleagues at AIT was very great!

Besides, I want to thank my supervisor at the Vienna University of Technology, Matteo Maffei. During our regular meetings he always supported me with very valuable contributions and ideas. Towards the end of my master thesis he was very cooperative and helpful so that I finished the thesis within my personal deadlines.

Additionally, I want to thank Margit Außerhofer, who proofread this thesis concerning the written language very short-dated and intensely.

Finally, I want to thank my girlfriend, who relentlessly supported me during my whole studies at any time and who accepted my limited spare time with full understanding. Furthermore my family always backed me in important situations - many thanks!

Kurzfassung

Während der Finanzkrise 2008 erschien eine digitale, verteilte und dezentrale Kryptowährung. Die Absicht hinter Bitcoin war es, seinen Nutzern eine von staatlichen Behörden unkontrollierbare und zensurresistente Möglichkeit zu bieten, Vermögenswerte uneingeschränkt zu kontrollieren. Mit zunehmender Verbreitung und einer stetig steigenden Anzahl an Transaktionen haben sich Limitierungen in Bezug auf die Skalierbarkeit und die Privatsphäre gezeigt. Um diese Limitierungen zu überwinden, wurden neben anderen Lösungen Second Layer Networks entwickelt. Diese Netzwerke operieren neben der eigentlichen Blockchain. Anstatt jede einzelne Transaktion auf der Blockchain zu speichern, speichern Second Layer Networks nur Resultate von mehreren Transaktionen auf ihr ab. Dies erlaubt eine höhere Skalierbarkeit und eine bessere Privatsphäre für die Nutzer.

Wir zeigen in dieser Arbeit, dass dennoch Schwachstellen in Bezug auf die Privatsphäre im Lightning Network, einem Second Layer Network für Bitcoin, existieren. Dafür haben wir über einen bestimmten Zeitraum Daten vom Lightning Network aufgezeichnet und anschließend verschiedene Analysen darauf angewendet. Weiters haben wir Daten vom Lightning Network mit Daten von der Bitcoin Blockchain in Verbindung gebracht, um mehr Einblick in die interagierenden Teilnehmer zu bekommen. Zusätzlich haben wir ein Cryptoasset Analytics Tool genutzt, um diese Daten um Informationen, welche vom Nutzerverhalten von Bitcoin gewonnen werden können, zu erweiteren. Dies ermöglichte uns einen noch tieferen Einblick in die Teilnehmer.

Mit diesem Ansatz konnten wir Daten von der Bitcoin Blockchain, dem Lightning Network und von einem Cryptoasset Analytics Tool verbinden. Im Detail haben wir eine der größten Nodes vom Lightning Network, ein populärer Wallet Service, mit einem dem Cryptoasset Analytics Tool bekannten Cluster basierend auf einer Heuristik verknüpft. Dies erlaubte uns einen umfassenden Einblick, welchen wir in dieser Arbeit präsentieren.

Auch wenn das Lightning Network entwickelt wurde, um die Privatsphäre von Bitcoin-Nutzern zu stärken, kann die Privatsphäre durch Kombination von Daten des Lightning Networks, der Bitcoin Blockchain und Informationen eines Cryptoasset Analytics Tools verringert werden.

Abstract

During the financial crisis in 2008, a digital, distributed and decentralised crypto currency arose. Bitcoin was intended to overcome control and censorship by governmental authorities and to enable its users to be under full control of their assets at any time. With ongoing adoption and an increasing amount of transactions, limitations like scalability and weak privacy came to light. To hurdle these limitations, second layer networks were developed among other solutions. Those networks are operating on top of the blockchain. Instead of storing every single transaction on the blockchain, they only store results of several transactions on it. This allows a higher scalability and a better privacy to the users.

In this thesis, we show that there are still some privacy issues within the Lightning Network, a second layer network for Bitcoin. Therefore we collected data from the Lightning Network over a certain period and performed several analyses on it. We also linked data collected from the Lightning Network to extracted data from the Bitcoin blockchain to increase the knowledge about the interacting participants. In a final step, we used a cryptoasset analytics tool to enrich our data by information gained from the user behaviour of Bitcoin.

With this approach, we were able to link data from the Bitcoin blockchain, the Lightning Network and a cryptoasset analytics tool. In detail, we mapped one of the biggest nodes from the Lightning Network, a very popular wallet service, to the respective cluster known to the cryptoasset analytics tool based on a heuristic. This allowed us a broad insight into the participants that we will present in this paper.

Even if the Lightning Network was intended to strengthen the privacy of Bitcoin users, the privacy can nevertheless be reduced by combining data from the Lightning Network, the Bitcoin blockchain and information from a cryptoasset analytics tool.

Contents

CHAPTER 1

Introduction

During the financial crisis in 2008, a decentralised and distributed payment solution arose. Since then Bitcoin [1], introduced by Satoshi Nakamoto, has been disrupting the payment industry like banks and companies on the one hand, and also governmental authorities on the other hand. It eliminates the need of trust by applying mathematics and cryptography and in this way can eliminate intermediaries in payments. In traditional currencies like EUR or USD, banks are keeping records of the balances of their customers. Therefore a user has to trust the bank not to change his balance arbitrary. Bitcoin is storing the transaction history in a public and distributed ledger. This ledger, called the Bitcoin blockchain, is secured by several cryptography mechanisms so that the state of the blockchain is always reliable and can be trusted. Bitcoin and other crypto currencies offer their users the capabilities to strengthen their self-responsibility and to operate peer-to-peer without any control and restrictions of governmental agencies.

From then on Bitcoin and related projects and technologies have attracted very much attention. During the last decade a lot of new crypto currencies and tokens, as well as improved or completely new distributed ledger technologies (DLTs) emerged. For example the number of projects on CoinMarketCap, a popular website to observe market prices and market capitalization of many projects, increased from 14 projects in May 2013 [2] to more than 2,400 projects in October 2019 [3]. Within the last five years, the amount of Bitcoin transactions per day increased from about 88,000 transactions on 21st February 2015 to about 305,000 transaction on 22nd February 2020 [4]. This is an increase of about 347%. Also, the market capitalization increased from 3.3 billion USD on 21st February 2015 to about 174.5 billion USD on 20th February 2020. This is an increase of about 5,288% [5]. The huge increase in market capitalization is also caused by the enormous price increase of Bitcoin, which was about 245 USD per Bitcoin on 22nd February 2015 and 9,697 USD per Bitcoin on 22nd February 2020 [6]. The price per Bitcoin increased by 3,958% within the last five years.

This significant rise indicates the increasing interest in Bitcoin and therefore in decentralised and distributed payment systems of the public, as well as the research and development area. On the other hand, Bitcoin and other crypto currencies also have major limitations like scalability and privacy, only to name a few [7]. If Bitcoin faces to become a common payment solution, it has to provide a higher transactions per second (tps) rate. For example VISA, which is one of the biggest payment providers, processes about 65,000 tps [8]. Bitcoin in comparison can process 4 tps [9]. Bitcoin also has to overcome privacy concerns due to public and distributed characteristics. In general, everybody can observe the flow and the owner of Bitcoins on the blockchain, but in most cases, it is not possible to link this information to persons. Therefore Bitcoin only provides pseudo-anonymity. To overcome these two limitations, payment channel networks have been developed that are operating on top of the blockchain. The Lightning Network [10, 11], a payment channel network for Bitcoin, enables privacy-preserving high frequent Bitcoin payments to its users at very low fees.

Paid online content for example is a use case for high frequent and private payments at very low fees. In 2018 the DCI Institut and the Hochschule Fresenius Hamburg published a study on paid content in Germany [12]. Compared to 2017, the number of internet users willing to pay for online content increased. In general, the rate of the users willing to pay slightly decreased, but the total amount of internet users increased at the same time. Most popular paid content offers are music and movie services, but the news industry increased most significantly compared to 2017. When it comes to the payment method for paid content, PayPal [13] is the most popular service, followed by traditional payments like credit card or bank account. All these payment solutions are centralised which does not fit the trend to decentralised and distributed payment systems. Paid content needs an accountant model like monthly subscription or pay-per-use. Pay-per-use means that a user pays for example for every video, song or article consumed from a service. This can even go further, i.e. to pay for every second of a song or movie and every word read of an article [14]. Classical payment solutions are not constructed for such high frequent payments with very small amounts, nor is Bitcoin and most of the other crypto currencies, as we will discuss in section 2.2.17.

The Lightning Network could be a payment solution for this use case and others. Besides the mentioned advantages, there are also some limitations and weak spots within the Lightning Network, especially concerning privacy. In this paper, we will discuss these limitations in detail and we will show what information can be revealed from the Lightning Network.

1.1 Aims

Payment channel networks were developed to overcome scalability and privacy limitations of blockchain-based crypto currencies like Bitcoin. They are operating on top of blockchains and are providing better scalability and privacy by summarising an unbounded amount of transactions to only two transactions. We will analyse payment channel networks concerning privacy aspects. We will discuss that payment channel networks do not offer full privacy by default.

This paper aims to contribute to the international research in the area of payment channel networks, their analysis and their possibilities as well as their weak spots. As mentioned in [15], it is a myth that off-chain transactions, transactions that are processed within a second layer network, offer privacy by default. We will focus on observing significant network and payment details that are private and hidden in common belief.

Therefore we will focus on the following three research questions:

- What kind of information can be deduced from second layer networks such as Lightning Network for Bitcoin, based on the footprints left on the blockchain and the second layer network communication itself?
- What kind of information that in common belief is private per default can be leaked within second layer networks?
- How can we integrate off-chain analysis into existing cryptoasset analytics tools and methods?

Our main focus is to analyse the Lightning Network concerning common network characteristics like degree distribution, betweenness and closeness centrality and payment details like interacting parties, transferred amounts, etc. By linking data collected from the Lightning Network to data extracted from the Bitcoin blockchain, we will identify important interacting participants of the Lightning Network. In addition, by enriching the data by information gained from a cryptoasset analytics tool, insight into interacting participants can be deepened.

By better understanding the network, its behaviour, its participants, their roles and revealable information, we extend the knowledge base for further improvements and developments of second layer solutions for blockchains concerning privacy aspects.

1.2 Structure

This paper is structured as follows: First, we are explaining DLTs in common. Next, we are introducing details on the Lightning Network and subsequently the methodology, followed by presenting the results. Finally, we are concluding the paper. In the following, we are briefly summarising the subsequent chapters.

- **Distributed Ledger Technologies** - In chapter 2, we will introduce the reader in DLTs and its building blocks. We will discuss cryptographic primitives, blockchain fundamentals, privacy and security aspects, scaling approaches and threats and critical perspectives.

- **Lightning Network** - In chapter 3, we will introduce the reader into details on the Lightning Network. We will discuss the concepts of channels and payments, Lightning Network fundamentals, limitations and challenges, privacy and security aspects, enhancement approaches, analysis in this field and critical perspectives.
- **Methodology** - In chapter 4, we will introduce the reader into our methodology. We will explain, how we collected and extracted data from the Lightning Network and the Bitcoin blockchain and how we linked and enriched them in additional steps. Finally we will also explain our heuristic in detail to link Lightning Network data to information from a cryptoasset analytics tool.
- **Results** - In chapter 5, we will present the results based on the introduced methodology. We will present different results based on the analysis of the data from the Lightning Network, on the data from the Bitcoin blockchain, on linked data from both networks and on enriched data by a cryptoasset analytics tool.
- **Discussion & Conclusion** - In chapter 6, we will conclude the paper by summarising the key findings of our results, discussing limitations of our approach and by giving ideas for future work in this area.

We published the main parts of the source code that were used to process, transform and analyse the data on a Github repository [1].

[1] https://github.com/PeHo89/lightning_network_analysis

CHAPTER 2

Distributed Ledger Technologies

The Bitcoin blockchain, respectively a blockchain in general, is a kind of distributed ledger technology (DLT) with many advantages compared to traditional ledgers. It is public, distributed, reliable, not manipulable and does not require an administrative entity, just to name a few. On the other hand, the blockchain also has some major limitations. The Bitcoin blockchain for example has limitations concerning throughput, latency, fees and its size [7]. The most important limitations will be discussed in section 2.2.17.

Distributed ledgers are building the backbone of every project and crypto currency respectively token in this field. In the following sections we will give the reader a detailed introduction into cryptographic primitives, fundamentals of blockchains and other DLTs, their privacy and security aspects, as well as existing solutions to overcome the major limitations of blockchains. Finally, we will also point out some critical perspectives and threats concerning this new and emerging technology.

2.1 Cryptographic Primitives

DLTs are applying cryptography to ensure reliability. Bitcoin is elementarily based on several cryptographic mechanisms, such as asymmetric cryptography, elliptic curve cryptography and hash and encoding-decoding functions that will be discussed subsequently.

Hash Function SHA-256 [16] is a hash function that produces a fixed 256 bit output, independently of the length of the input. RIPEMD-160 [17] is also a hash function that produces a fixed 160 bit output. Hash functions have three main properties. First, they are one way functions that cannot be reversed. Second, the output does not reveal information about the input. Third, it is hard to find collisions so that two different inputs produce the same output. SHA-256 is used in Bitcoin in proof-of-work (PoW) and both, SHA-256 and RIPEMD-160 are used in the address generation process.

Encoding Base58 is a binary to text encoding and decoding scheme to represent large integers as an alphanumeric text. It has been modified from Base64 that ranges from a-z, A-Z and 0-9, to prevent confusions on ambiguous looking characters like 0 (zero) and O (capital letter O). The word 'Bitcoin' Base58 encoded is '3WyEDWjcVB' for example. Bitcoin addresses and keys are represented in a Base58 encoded way [18].

Asymmetric Cryptography Bitcoins are controlled by a public private key pair that is building the base for addresses and signatures. The generation of a public private key pair is based on elliptic curve cryptography. Elliptic curve cryptography is based on the discrete logarithm problem as expressed by addition and multiplication on the points of an elliptic curve [19]. Bitcoin is using elliptic curve digital signature algorithm (ECDSA) with parameters specified in SECP256K1 [20]. The curve is defined by $y^2 = x^3$ + ax + b, where a is 0 and b is 7 specified in SECP256K1. This results in $y^2 = x^3$ + 7, shown in figure 2.1. There are further parameters specified in SECP256K1, which are needed to operate on the elliptic curve, such as the base point (generator point) G, the order n of G and the cofactor h. The order of G is 1.1578 * 10^{77}, which is a little less than 2^{256}. The generator point is a specific point on the elliptic curve.

The key generation process is as following: A very big random number between 0 and n-1 (inclusive) is chosen that specifies how often the generator point G is multiplied by itself. The result is the pubic key, another point on the elliptic curve. The randomly chosen number at the beginning is the respective private key. Elliptic curve multiplication is an irreversible process, so the private key cannot be generated from the public key. Finally, the Bitcoin address is derived from the public key by first applying SHA-256 on it and then applying RIPEMD-160 on the prior result. This results in the public key hash that gets Base58 encoded and prefixed with a version. This finally results in the Bitcoin address [19]. Figure 2.2 shows the address generation process in Bitcoin.

Signatures are a third main part of the ECDSA beside the public and the private key. Signatures are used as a verification mechanism so that only the owner of a certain private key can make use of specific Bitcoins. It guarantees that the sender owns the corresponding private key without revealing it. A signature is created from the hash of the transaction data by applying the private key on it. When the transaction data and its signature is sent to the network, every node can verify that the transaction data can only have been created by the owner of the specific private key. The nodes proof this by first creating the hash of the transaction data. Next, they use the signature, the public key and the created hash to verify if the corresponding private key has signed the data. If so, the transaction is valid and gets added to the blockchain [21].

Zero-knowledge-proof (ZKP) ZKP is another fundamental building block in cryptography. Broadly speaking, a ZKP allows an entity to prove that it knows a secret, without telling it to the verifier. E.g. there is a locked door and the prover claims that he has the key to unlock it. Without showing the key to the verifier, he can prove that he owns the key by unlocking the door and showing the unlocked door to the verifier. It has

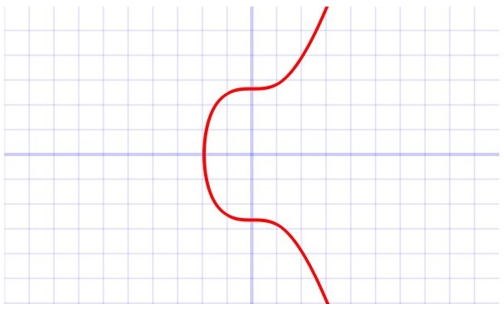


Figure 2.1: The elliptic curve ($y^2 = x^3 + 7$) used in Bitcoin for the creation of the private and public key [22].

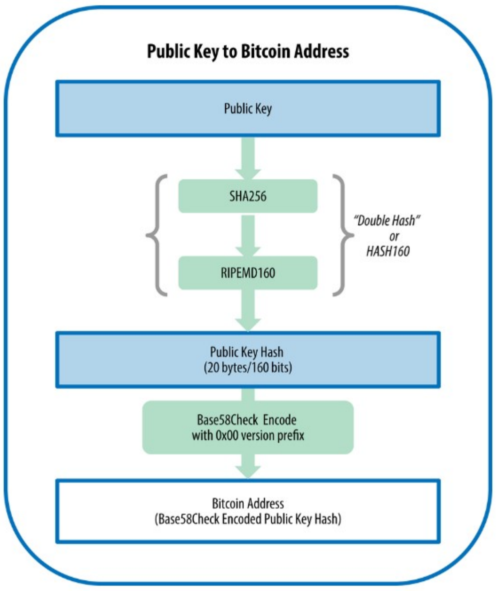

Public Key to Bitcoin Address
Public Key
SHA256
RIPEMD160
"Double Hash" or HASH160
Public Key Hash (20 bytes/160 bits)
Base58Check Encode with 0x00 version prefix
Bitcoin Address (Base58Check Encoded Public Key Hash)

Figure 2.2: The process from a public key to a Bitcoin address. This process is one-way, meaning a public key cannot be derived from an address [19].

to be guaranteed that no one else unlocked the door in the meanwhile. As introduced in [23], a ZKP consists of a sequence of steps, shown in figure 2.3. The prover and the verifier can repeat these steps several times to increase the probability that the prover knows the secret and had no lucky shot. Non-interactive zero-knowledge-proofs (ZKPs) are an extended type where the prover can publish his proof and a verifier can ensure its correctness at any time in an asynchronous way.

In this section, we have been discussed the most important cryptographic primitives used in DLTs. In the next section, blockchain and its fundamental building blocks will be introduced in detail.

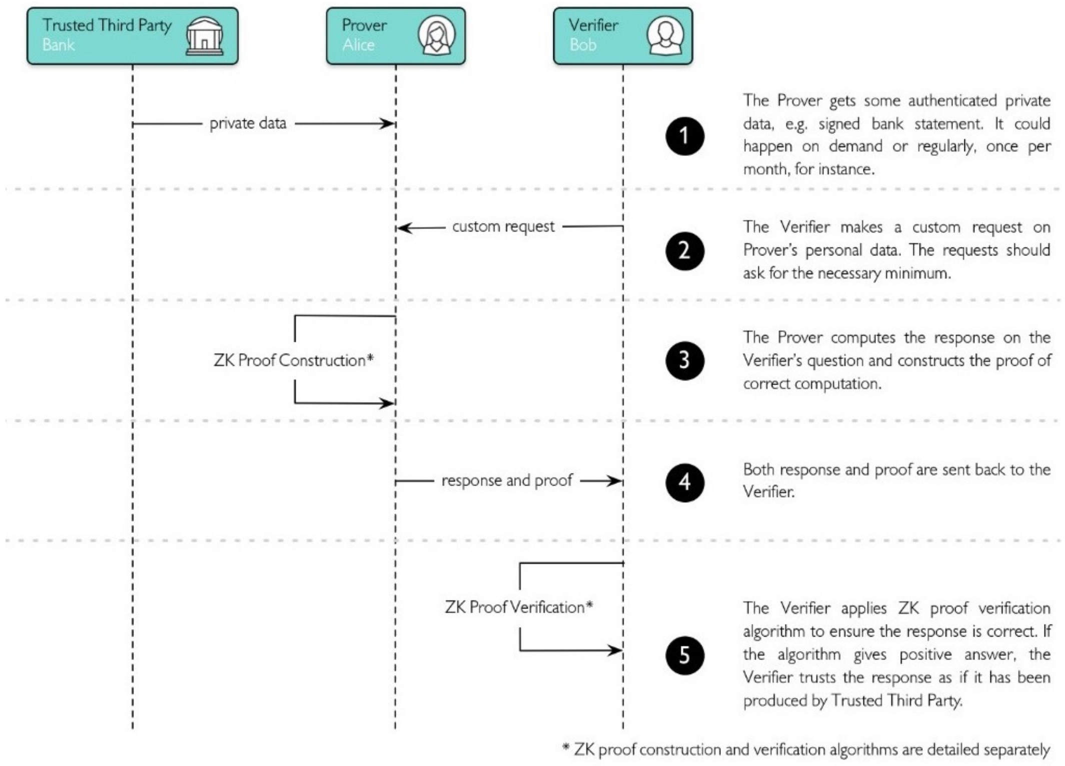

Trusted Third Party
Bank
Prover
Alice
Verifier
Bob
private data
1
The Prover gets some authenticated private data, e.g. signed bank statement. It could happen on demand or regularly, once per month, for instance.
custom request
2
The Verifier makes a custom request on Prover's personal data. The requests should ask for the necessary minimum.
ZK Proof Construction*
3
The Prover computes the response on the Verifier's question and constructs the proof of correct computation.
response and proof
4
Both response and proof are sent back to the Verifier.
ZK Proof Verification*
5
The Verifier applies ZK proof verification algorithm to ensure the response is correct. If the algorithm gives positive answer, the Verifier trusts the response as if it has been produced by Trusted Third Party.
* ZK proof construction and verification algorithms are detailed separately

Figure 2.3: Sequential steps in a ZKP. By repeating these steps, the significance of a ZKP can be increased [23].

2.2 Blockchain Fundamentals

A blockchain is a kind of DLT built on different layers. Figure 2.4 shows the six layers of a blockchain, introduced by Zhang et al. in [24]. The data layer is defining the structure of the data in which the transactions and other information are stored. The network layer describes the network architecture and further mechanisms for transmission and validation. The consensus layer specifies the type of algorithm, to reach a common consensus within the blockchain. The incentive layer is related to game theoretical aspects, like coin issuing and distribution, as well as rewards for operating entities. The contract layer defines the programmatic possibilities of a blockchain and finally the application layer is specifying high-level applications based on the blockchain.

A blockchain can be interpreted as a linked list of data that cannot be easily modified once it is inserted and confirmed [25]. The computational effort that has to be spent to change data that is already stored on the blockchain increases with the elapsed time. This is ensured by the architecture of the blockchain. A blockchain consists of linked blocks each of which contains specific data. In the case of Bitcoin it contains transaction information. Once a block is full, which means it cannot store more data than it already

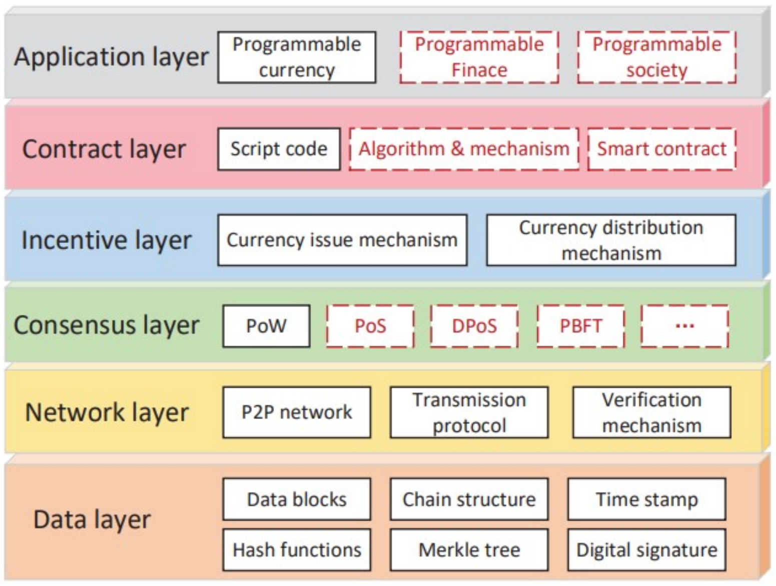


Figure 2.4: A blockchain is constructed based on several layers. Each layer describes a different architectural design dimension [24].

does, its hash including all the transaction data is calculated and is referenced in the next block. That is the reason why data cannot be altered afterwards because then the hash of the block with the altered data will change and therefore not fit the referenced hash in the subsequent block. This will lead to an invalid state of the blockchain. Figure 2.5 shows a conceptional view of a blockchain.

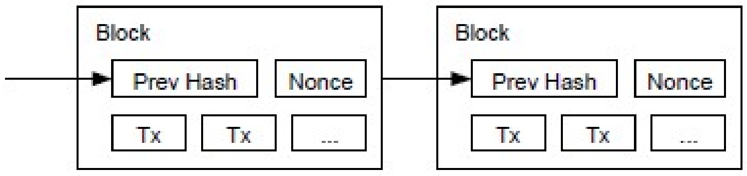


Figure 2.5: Each block in a blockchain references the former block and contains data like transactions and meta information [1].

In general, there are basically two types of nodes: mining and validating nodes. Both types of nodes maintain a copy of the blockchain locally. So every node can go back in transaction history and verify the validity of a certain transaction at any time. Once a transaction is made, this information is sent to the complete network. A validating node only verifies, if the new transaction is valid. A mining node verifies this and adds this transaction to its mempool, the 'bucket' of new and unconfirmed transactions. A mining node selects the transaction it likes - mostly based on the amount of the fees that can be earned by a transaction - from the mempool and puts it into its potential next block, called the candidate block. These transactions are unconfirmed at this time. When a block is full or even if a mining node decides it is, the node tries to set a nonce, basically a random number, to create a hash of all the containing transaction data, the hash of the previous block and the chosen nonce. This created hash has to fulfill a specific criterion. In the case of the Bitcoin blockchain, the calculated hash has to begin with a certain number of zeros. The difficulty defines how many zeros the hash has to have at the beginning of it. The higher the number of zeros required at the beginning of the hash, the harder it is to find a fitting nonce. A suitable nonce can only be found by trail and error due to the properties of a hash function, which we discussed in section 2.1. The node sets a random nonce and calculates the hash of the block. If the corresponding hash does not begin with the correct amount of zeros, another random nonce is set and the process is repeated until a proper nonce is found or another node already sends a valid block because it has found a correct nonce. This process is operated by a consensus mechanism called PoW which is part of the Nakamoto Consensus [26]. So once a proper nonce is found by a node, the next block is 'mined' and is sent to the complete network. Every node, which receives the new block, validates all the containing transactions by having a look into the local transaction history of the blockchain. If all the transactions are valid, if the provided nonce is a proper one and if the referenced previous block hash is the correct one, the block is added to the local copy of the blockchain on the node. All the containing transactions are then in a confirmed state. The node which proposed the new block gets the transaction fees of all the containing transactions and a block reward, which are newly 'created' Bitcoins.

For a certain period there may exist different states of the blockchain, called chain split [27]. This happens when different nodes find proper nonces at about the same time and send the new blocks to the network. In this case, a rule named 'Longest Chain Rule' is applied [28]. Let us assume two nodes, node A and node B, send their new blocks to the network. All other nodes will add both blocks each to their local blockchain and so two different versions of the blockchain arise. Some of the nodes will add the block to the first version whereas some nodes will add it to the second version. For the next new block each node adds it to their locally longest chain. Due to different computing power, after some time only one longest chain emerges within the network that is the valid one. All the other blocks and their corresponding transactions are then reverted. So the finality of a transaction is based on probabilities that are increasing over the time.

The idea behind PoW is to limit and harden the process of adding new transactions to the blockchain. If everybody could easily add transactions without having anything to achieve, then the blockchain will not have a common global and valid state because every node will probably add transactions from which it will benefit most. The system would be more or less worthless. PoW is one kind of consensus mechanism that is easy to implement compared to others. That is the reason why Bitcoin applied PoW. Today this consensus mechanism can be seen as proven after more than 10 years of being in production environment and also an interesting target for attacks. Beside PoW there also exist other consensus algorithms which we will discuss in section 2.4.4.

The difficulty, which means the amount of required leading zeros of a created block hash, is adjusted by the Nakamoto Consensus in a way that on average about every 10 minutes a new block is mined. The protocol does so by analysing the total hash power of the network and the average creation time of the last few blocks and their corresponding difficulties. In some cases the next new block is mined within only a couple of minutes and in other cases it takes significantly longer than 10 minutes. Over time on average, 10 minutes per block is achieved.

Bitcoin is limited to a total supply of 21 million Bitcoins which gets created by the protocol over time. At the moment, about 18.1 million Bitcoins are created by the protocol [29]. Nodes mining new blocks are rewarded with the block reward. As mentioned above, the block reward are newly created Bitcoins. The block reward started with 50 BTC per new block and is halving every 210,000 blocks [30]. The actual block reward is 12.5 BTC, which gets halved most probably in the middle of May 2020. At the time of this writing, the latest block height was 607,061 [29]. With block height 630,001 the block reward is halved to 6.25 BTC per new mined block. At some point in time, all Bitcoins are mined and miners are only rewarded with transaction fees.

The current Bitcoin network consists of about 9,500 nodes [31] that can produce about 93 EH/s [1] [32] in total and about 10 TH/s [2] on average per node.

Bitcoin is mainly a payment application, or at least it was intended to be one. To bring the advantages of blockchains to a broader audience, V. Buterin introduced a concept for distributed and decentralised applications in a generic way in the end of 2013 [33]. Ethereum [34] is a blockchain based platform that beside payments also offers the concepts of smart contracts. Smart contracts are applications that are stored and executed on the blockchain, consequently they are deterministic and trustless. We will touch on smart contracts and their use cases in sections 2.2.15, 2.2.5 and 2.2.6.

In this section, we introduced the reader into blockchain and Bitcoin in common. Next, we will discuss some design dimensions of blockchains and other DLTs.

2.2.1 Design Dimensions

Most of the existing DLT and blockchain based projects can be classified among the following characteristics.

A blockchain can either be permissioned (private), which means not everyone can be part of it, or permissionless (public), which means everyone can be part of it without any restrictions [35]. In permissioned blockchains participating nodes often have to fulfill certain criteria to be able to be part of the network, e.g. pay a certain amount of deposit, be an enterprise with a certain annual revenue, and so on. For example, Bitcoin

[1]exa hash per second

[2]terra hash per second

is a permissionless and Hyperledger Fabric, discussed in section 2.4.5, a permissioned blockchain. In permissionless blockchains the respective client can be downloaded and executed without any limitation. The main advantages of a permissioned blockchain compared to a permissionless blockchain are scalability and efficiency. In a private blockchain consensus has to be reached only within a small group of nodes. This makes it faster and requires less resources. The main benefits of a permissionless blockchain are security, immutability and censorship-resistance. In a public blockchain data is distributed across many nodes and that ensures integrity, availability and security.

Furthermore, any permissioned or permissionless blockchain can be open or closed, concerning the right of reading the stored data. This results in four different types: permissionless & open, permissionless & closed, permissioned & open and permissioned & closed [35].

Regardless of the right of reading the stored data or the possibility of contributing to the network, there are many different algorithms and mechanisms for reaching consensus in a blockchain. Based on the design dimensions, other blockchains use different consensus algorithms. An open and permissionless blockchain for example can use PoW, whereas a closed and permissioned blockchain can use proof-of-authority. Different consensus algorithms will be mentioned in section 2.4.4.

Besides, there are two types of balance models in current blockchains. A blockchain can either be unspent transaction output (UTXO) or account based [36]. UTXO based blockchains focus on the underlying assets, which means the flow of the assets is recorded and private keys are controlling single outputs of transactions. This model will be discussed in section 2.2.12. On the other hand, account based blockchains focus on single accounts and their balances, like in traditional banking systems for example, where a private key controls all the funds related to a certain account. Bitcoin is a UTXO and Ethereum is an account based blockchain. The main advantages of a UTXO based compared to an account based blockchain are scalability, privacy and its stateless nature. Not summarising all transactions into one account provides a higher privacy. Also, processing stateless UTXOs in parallel enables better scalability. The main benefits of an account based blockchain are simplicity and efficiency. To keep track of an account is more simple and efficient than to trace single inputs and outputs of transactions and summarise them.

Beside these design dimensions, the following emerging model that correlates three aspects in DLTs has to be considered. The trilemma states that a distributed system cannot maximise all three properties, i.e. scalability, security and decentralization, at full extent at the same time. E.g. a system with many nodes is more secure than one with fewer nodes, but it has limited scalability compared to the other one. The same is true for scalability and decentralization. If a system is more decentralised than another system, it is less secure, because unknown third parties are operating in the system that could act maliciously. As the reader can see, maximizing properties always minimizes other properties. E.g. Bitcoin is relatively secure and mostly decentralized and has therefore scaling limitations beside others, which will be discussed in section 2.2.17. Solving the trilemma is one of the hardest challenges in the area of blockchains and crypto currencies. Figure 2.6 is visualizing the trade-off between these three properties.

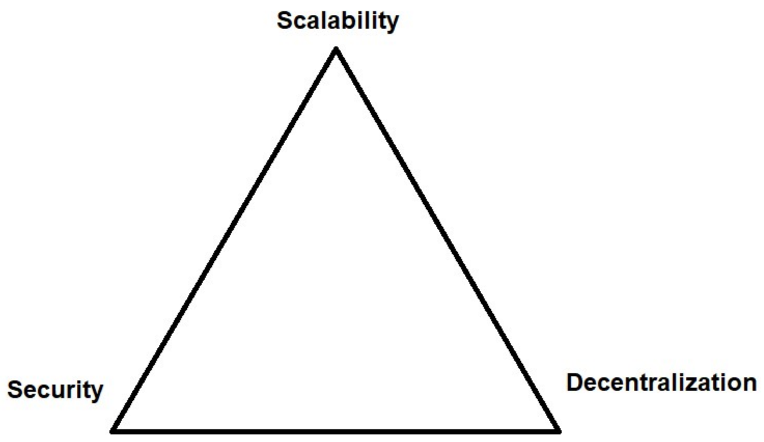


Figure 2.6: The trilemma in blockchains and other DLTs characterizes the trade-off between scalability, security and decentralization. Maximizing one of these properties is minimizing the remaining two [37].

We have discussed how different blockchains and DLTs can be designed and clustered based on the most important design dimensions. The trilemma states that there is a certain trade-off between scalability, decentralisation and security which lead to high activities in research and development during the last years. In the following sections further building blocks and fundamentals of blockchain and DLTs will be discussed.

2.2.2 Peer-to-Peer Networks

On a network level blockchains and other DLTs are often based on peer-to-peer networks, to operate in a distributed, reliable and censorship-resistant way. Communication is happens directly between nodes and information is mostly sent either by flooding or gossiping mechanisms. All nodes within a peer-to-peer systems are equal and there is no master node or similar authority with more rights than others. Nodes in a peer-to-peer network are connected in a mesh network and are both providing and consuming services [38].

2.2.3 Pools

Nodes within the peer-to-peer network and consequently within a blockchain can either operate independently or together with others in a pool of nodes. A pool is a union of several nodes operating as a single unit. A node offers its resources to a pool of resources, which splits the earned reward to the pool participants according to the amount of resources they contributed to it.

Pools based on the PoW algorithm are called mining pools. With its hashing power a node contributes to the pool and gets a part of the reward earned by the pool based on the share of the hashing power provided by the node to the total pool hashing power. If the pool hashing power is e.g. 4 EH/s and a nodes hashing power is 40 Th/s, the node gets 0.001% minus fees for the pool operator of the earned reward by the pool. In Bitcoin, four big mining pools, i.e. Poolin, F2Pool, 'BTC.com' and AntPool, control 58.8% of the total hash rate of the Bitcoin network [39]. Mining pools are often criticized concerning centralization.

Pools based on the proof-of-stake (PoS) algorithm, which we will also discuss in section 2.4.4, are called staking pools. The principle is the same as with mining pools. Instead of contributing with hashing power, a node contributes with the amount of coins or tokens it controls. Therefore it has to lock its funds in a specific address to enable the pool operator to make use of the funds in the block creation process. This is some kind of a security issue because one has to hand over the control of the funds to a trusted third party. To avoid that, there are also cold staking pools. In cold staking pools funds are locked in cold wallets, which are under control of the node, instead of locking funds in a specific address not under control of the node [40].

2.2.4 Wallets

Another important building block in blockchains and DLTs are wallets. Broadly speaking, a wallet is a 'place' where private keys are stored. There are two types of wallets: hot and cold wallets. A short introduction into both types, as well as into their advantages and drawbacks, as presented in [41] will follow.

There are three types of hot wallets: desktop, mobile and online wallets. Desktop wallets are programs installed and executed on a computer. Secured by a password they allow to control the coins via the keys. Mobile wallets are similar to desktop wallets, but instead of being installed on a computer, they are installed and executed on a smartphone. The third type are online wallets which are services like exchanges or other wallet services. Besides, hot wallets can be custodial or non-custodial. Custodial means, a service is managing the funds for the user. From a technical point of view, in custodial wallets the user does not fully controlling the coins but the wallet service or program does it for the user. At non-custodial wallets the user alone has control over the respective keys. The advantages of hot wallets are, i.e. that they are easily accessible, user friendly and usually free or inexpensive. The drawbacks are, i.e. that they are vulnerable to hacking, the user has no control over the keys in custodial wallets and they are susceptible to scams.

There are two types of cold wallets: hardware and paper wallets. A hardware wallet is a physical device which has to be connected to a node either via USB or via another interface. Private keys are stored on the device itself and do not even leave the device when creating a transaction. If the user wants to send a transaction, the transaction is sent to the device, is signed on the device and sent back to the node, from where it gets sent to the network. A paper wallet is a piece of paper printed with the private key, either in form of a QR code or in clear text. The user has to scan or type the private key into a wallet software if he wants to make a transaction. The advantages of cold wallets are i.e. that they are highly secure, not dependent on a third party and that the user is in control of the keys at any time. In case of a hardware wallet, an additional advantage is that the key never leaves the device. However, drawbacks are i.e. that they are less user friendly and more expensive than hot wallets and that they can be lost or destroyed. Once a private key is lost, also all the coins controlled by the key are lost. A study shows that about 20% of all existing Bitcoins are lost because the private keys were lost [42].

As mentioned in section 2.1, a Bitcoin address is generated from the public key. The public key on the other hand is generated from the private key which is based on a very big random number. Instead of using a very big random number, hierarchical deterministic wallets use a seed as an input for the private key generation process. This seed mostly consists of 12 or 24 randomly selected words from a predefined word list. Based on the selected words, a master private key is generated. By increasing the master private key, derived private keys are generated. From these generated private keys, Bitcoin addresses are calculated. Derived private keys can again be used as master keys for other derived keys. The depth of key levels is not limited. The main advantage of hierarchical deterministic wallets is i.e. that not every private key has to be backed up, but only the seed and the increasing mechanism has to be stored. Based on the seed, each derived private key can be re-created and so can the public keys and Bitcoin address. The concept of hierarchical deterministic wallets was introduced with Bitcoin Improvement Proposal (BIP) 32 [43]. Figure 2.7 shows the conceptual view of a hierarchical deterministic wallet.

2.2.5 Coins & Tokens

As discussed in 2.2.4, wallets can handle coins, more precisely they can handle private keys. When it comes to Bitcoin and blockchain, often the terms token and coin are used. In many cases they are mistakenly used as synonyms. However, both terms target different things and have different purposes, as summarised in [45].

A coin is the native currency of a blockchain project, e.g. in Bitcoin this is Bitcoin (BTC) and in Ethereum this is Ether (ETH). The coin is the monetary instrument for operating the network. Fees and rewards are paid in the native coin.

On the other hand, tokens are created on a blockchain by e.g. smart contracts discussed in section 2.2.15. A smart contract can issue new tokens and handle the owner of existing tokens. New tokens are released by a smart contract by receiving coins as a countertrade. What a token represents, depends on the purposes of the developers of the respective token smart contract. There are two main types of tokens: fungible and non-fungible tokens. In Ethereum, tokens are standardized by Ethereum Improvement Proposals (EIPs) based on Ethereum Request for Comments (ERCs). ERC-20 [46] is the most used fungible token standard, whereas ERC-721 [47] is the most used standard for non-fungible tokens. The most known ERC-20 token on the Ethereum blockchain is Tether [48], i.e.

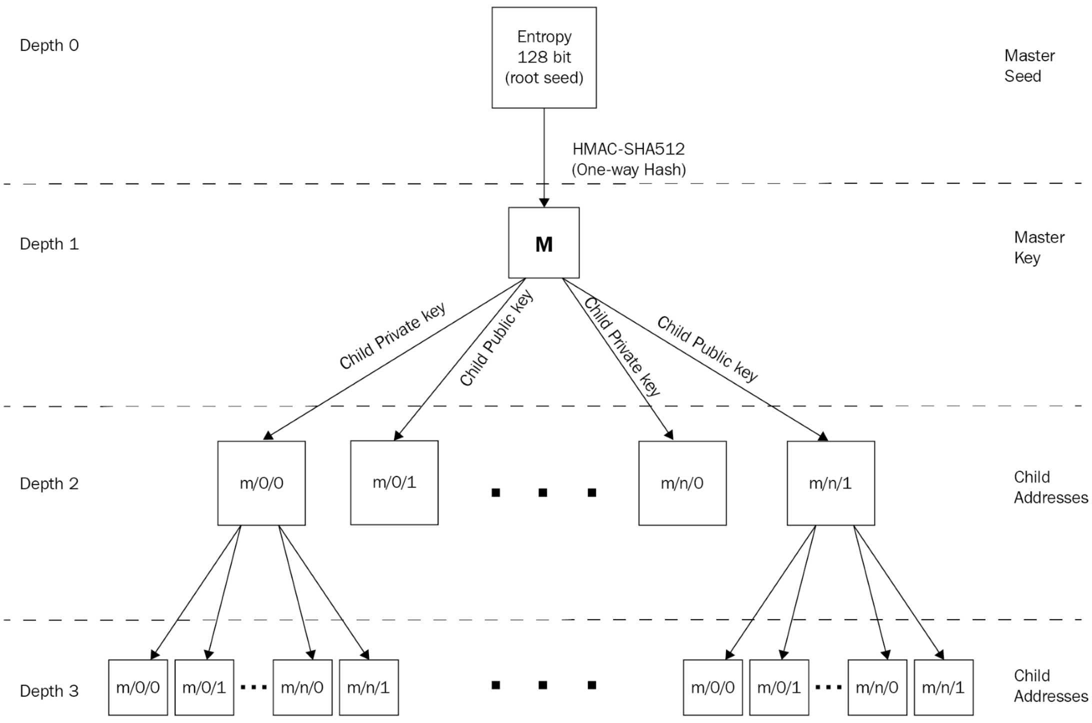


Figure 2.7: A hierarchical deterministic wallet starts with one root seed from which a master key is derived. By performing operations on the master key, child keys are derived. From each child key subsequent child keys can be constructed. The level of depth is not limited. By knowing the root seed and the operations that were performed on them, each key can be re-constructed at any time [44].

a stable-coin that is claimed to be backed by the USD [49]. The most known ERC-721 token on Ethereum is Gods Unchained Cards [50], i.e. collectable cards which are unique and virtual [51]. Tokens can also represent physical goods as discussed in section 2.2.6.

There is another distinction between different types of token concerning the value they are representing, summarised in [52]. There are i.e. utility, security, currency, reward and asset tokens. Utility tokens are used for specific products or services of a project or company, therefore they are representing access. Security tokens represent a share of a company, like digital stock certificates, consequently they represent ownership. Currency tokens are the same as coins, which we mentioned earlier in this section. Reward tokens are some kind of an incentive for the user for a specific service. Finally, asset tokens represent real world assets, e.g. a windmill. In the next section this will be discussed in detail.

2.2.6 Tokenization

Within the last few years, the term tokenization has got more and more attention in the field of blockchains. Tokenization which is explained in detail in [53], basically means the division of digital or physical goods into small parts called a token which is unique and cannot be copied. Tokens are usually maintained by smart contracts, discussed in section 2.2.15. A token is an asset in a DLT and it represents a part of a good, ideally connected with its rights and duties. For example an operator of a windmill divides one windmill worth 1 million USD in total into 1,000 tokens, worth 1,000 USD each. Therefore each token represents 0.1% of the windmill and each token owner also gets 0.1% of the produced energy or earned profit. To ensure the connection of a token to its rights and duties which is a big challenge, Liechtenstein released the legal framework for tokenization which became significant by the beginning of 2020 as the first country in the world, called the Liechtenstein Blockchain Act [54]. The token container model provides a legal framework for transferring physical goods into its DLT based representation connected with its rights and duties. As the reader can see, as a consequence some kind of governmental and central control arises, which is not intended by DLTs.

In the last two sections the concepts of coins, tokens and tokenization were introduced. Due to the increasing amount of different DLTs mentioned in the introduction of this chapter, also the need of exchanging coins and tokens across different blockchains and projects got more and more important. A way for interoperability will be discussed next.

2.2.7 Atomic Swaps

As mentioned, interoperability has got increasingly important. Atomic swaps allow two parties to exchange tokens or coins from two different blockchains. Let us for example assume that Alice and Bob want to exchange Bitcoin and Ethereum without trusting a third party, i.e. an exchange. They can exchange both coins in an atomic way, meaning either both or none of them spend their coins by using atomic swaps. Atomic swaps make either use of smart contracts, discussed in section 2.2.15 or hash time locked contracts (HTLCs), discussed in section 3.1.2, depending on the underlying blockchains and their capabilities. Atomic swaps can happen across blockchain forks discussed in detail in the next section 2.2.8, i.e. between Bitcoin and Bitcoin Cash, or between completely different blockchains, i.e. Bitcoin and Ethereum [55], both supporting atomic swaps.

2.2.8 Forks

Another important aspect in blockchains are forks. Every project within this new field is a project with ongoing development, and so is Bitcoin. During the last decade a lot of adaptions by the core developers of the Bitcoin protocol have happened. Each time a change in the software happens, all nodes within the network are either required to accept the new change or continue working on the old protocol. If not all of the nodes decide for the same, a fork happens. Two types of forks exist: soft and hard forks. Soft forks are small adaptions within the protocol, so that nodes which did not switch to the new protocol, can still validate blocks created by the new protocol. Hard forks are bigger changes within the protocol: Nodes which didn't switch to the new protocol are not able to validate blocks created by the new protocol. Then a different crypto currency forks from the former. The most known hard forks from Bitcoin are Litecoin, Bitcoin Cash, Bitcoin Gold and Bitcoin SV [56].

2.2.9 External Data & Oracles

A big challenge in DLTs is the access to external data, which will be discussed in section 2.2.17. In general, DLTs cannot access data from outside the network. That is probably one of the reasons why there is no real other killer application beside the payment application. If an application requires access to external data due to its business logic, it has to make use of oracles. Oracles are interfaces between the DLT and the world outside the network, i.e. the internet. There are five different types of oracles, mentioned in [57]:

- A software oracle maintains data from online sources like prices, events or other data.
- A hardware oracle maintains data directly from the physical world like from sensors or other entities.
- An inbound oracle provides data from the external world to the network.
- An outbound oracle provides the network with the ability to send data to the external world.
- A consensus-based oracle is directly involved into consensus creation of the DLT.

An oracle is a centralised interface which has to be trusted. It is not guaranteed per default that every node validating the execution of the application receives the same data from the external source or even data in general by the oracle. This can be caused by different reasons, i.e. time differences between requests, availability, malicious behavior, and so on. To overcome these limitations, the concept of decentralised oracles has been introduced. Decentralised oracles consist of networks of oracles, so it is necessary that at minimum of 3 out of 5 oracles provide the same date from outside the network, before a date is accepted by an application for example. This minimizes the single point of failure and also the dependency on a result of a single oracle [58]. ChainLink [59] e.g. is a project based on decentralised oracles providing secure access to data feeds, APIs and payments. Good behavior of oracles is incentivised and bad behavior will incur penalties. Reputation of oracles is public to minimize malicious behavior.

2.2.10 Trusted Execution Environments

Trusted Execution Environments (TEEs) can be used together with oracles to bring external data to the blockchain. A Trusted Execution Environment (TEE) is a protected area in a processor or even a separate processor that protects the integrity and confidentiality of data and applications. Intel SGX (Software Guard Extensions) is an implementation of a TEE. Execution and data storage is separated from the rest of the processor and secured by cryptographic assertions [60]. It is executed in an enclave, which detects altered or tampered data and denies access and disables the environment. For example Teechain [61] is an off-chain payment protocol making use of TEEs to perform secure and scalable transactions on the blockchain with asynchronous blockchain access. In a test deployment a throughput of 33,000 tps with a latency of 0.1 seconds was measured. With TEEs trust assumptions are shifted from the DLT to semiconductor manufacturers that suffer from their vulnerabilities like rollbacks and side-channel attacks [15].

2.2.11 Merkle Trees

Another important building block in the field of blockchains are merkle trees. A block in the Bitcoin blockchain contains several kinds of information, i.e. the block size, the block header, the transaction counter and the transactions itself. The block header consists of the version, the previous block hash, the merkle root, the timestamp, the difficulty target and the nonce. For efficient validating, if a certain transaction is part of a block, a node can traverse the merkle tree starting at its root, the merkle root. Based on the target hash, the node then goes to the left or the right branch. This is done several times until the hash of the transaction is found or another transaction hash is reached and therefore the searched hash is not found. A merkle tree is a binary hash tree, a data structure used in computer science for efficiently verifying and summarising the integrity of large data sets [62]. Figure 2.8 shows the conceptual view of a merkle tree in Bitcoin.

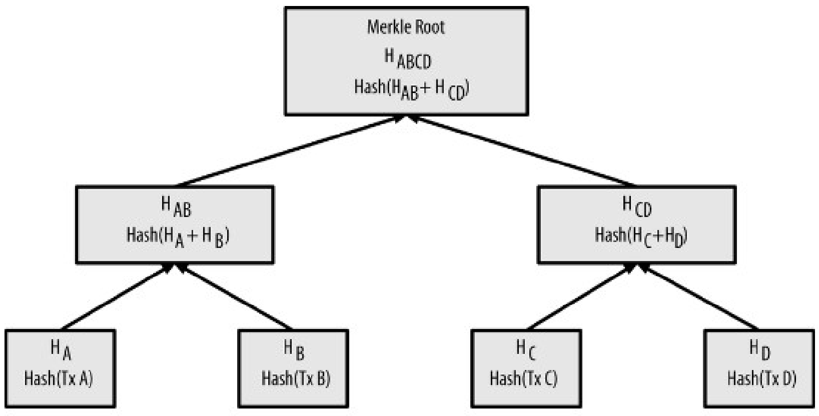


Figure 2.8: In a merkle tree each leaf stores some information and the nodes on the paths from the root to the leaves are constructed based on the children. By this, a merkle tree can be traversed very fast and it can be identified quickly whether a certain information is contained or not [62].

2.2.12 Transactions & Unspent Transaction Outputs

In general, a transaction defines the transition of assets from one entity to another. Especially in Bitcoin, another important building block is the concept of unspent transaction outputs (UTXOs). As mentioned in section 2.2.1, Bitcoin is based on the UTXO model. A transaction in Bitcoin is composed by a version, a locktime and at least one input and one output each. The input specifies from where the assets come and the outputs specify where the assets goes to. Figure 2.9 shows the conceptional view of a transaction. An output of a transaction is either an input of another transaction or it is unspent. If it is unspent, it can be used as an input for a new transaction. Transaction outputs are controlled via corresponding private keys respectively scripts which will be discussed in section 2.2.14. These scripts can be constructed in such a way that different transaction types arise as discussed in the next section 2.2.13.

As Bitcoin is not account based, a user has sum up all the values of all UTXOs under control by the owned private keys to obtain his balance in Bitcoin. In contrast, Ethereum is based on an account model where a private key controls an account. The balance of the account is continuously updated based on transactions. Both models have different advantages and drawbacks as discussed in section 2.2.1.

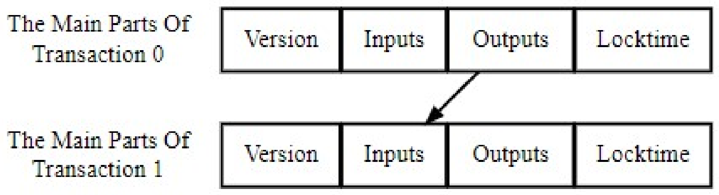

The Main Parts Of Transaction 0
Version
Inputs
Outputs
Locktime
The Main Parts Of Transaction 1
Version
Inputs
Outputs
Locktime

Figure 2.9: A Bitcoin transaction contains among others version, inputs, outputs and a locktime. Inputs of later transactions reference outputs of earlier transactions. By this, transactions can be traversed starting at any block to the very beginning of a blockchain [63].

2.2.13 Transaction Types

As discussed in the previous section, a very important building block especially of Bitcoin is the construction of transactions. In Bitcoin different transaction types based on the underlying use case exists. They are realised using scripting language Script, which will be introduced in the next section 2.2.14. First, a short overview of the most important transaction types introduced in [64] will be given.

Pay to public key hash (P2PKH) are the majority of transactions processed on the Bitcoin network. This transaction contains a locking script that prevents the output from being spent without presenting a signature and a public key produced by an entitled private key.

Pay to public key (P2PK) is a simpler form of a transaction than P2PKH. The public key itself is stored within the locking script rather than a hashed version of it like in P2PKH.

Pay to script hash (P2SH) allows transaction outputs which are bound to complex conditions, to be spent in a lightweight way. A script that matches the hash of the P2SH transaction has to be provided to be able to spend this transaction output.

Multi-signature transaction stores a set of public keys into the locking script of a transaction output. A certain amount of signatures, produced by private keys, have to be provided to be able to spend this transaction output. Multi-signature transactions are called m-of-n multi-signature transaction output, where m and n can range from 1 to 15 and m has to be less or equal to n. E.g. a 2-of-2 multi-signature transaction output can only be spent if both private keys provide their signatures. A 3-of-5 multi-signature transaction output can be spent if 3 out of 5 private keys provide their signatures. This can be any 3 out of the 5 predefined public keys.

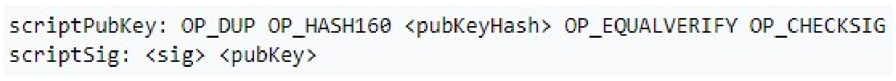
Pay to witness script hash (P2WSH) is similar to P2SH. The signatures contain the same information, but they are located in the Segregated Witness instead of in the script signature [65, 66]. The concepts of the Segregated Witness will be discussed in detail in section 2.4.2. Broadly speaking, information is split into two parts and only one fraction is part of the transaction, whereas the other part is stored somewhere else.

Pay to witness public key hash (P2WPKH) is similar to P2PKH, but again the signatures are located in the Segregated Witness instead of in the script signature [65, 66].

Data output (OP_RETURN) is used to store data to the blockchain that is not relevant to the operating principles of Bitcoin in general. An OP_RETURN output is not spendable and thereby it has a value of zero. It also is not provided with an unlocking script. It is a special form of a UTXO and is often used for anchoring information from other applications or use cases.

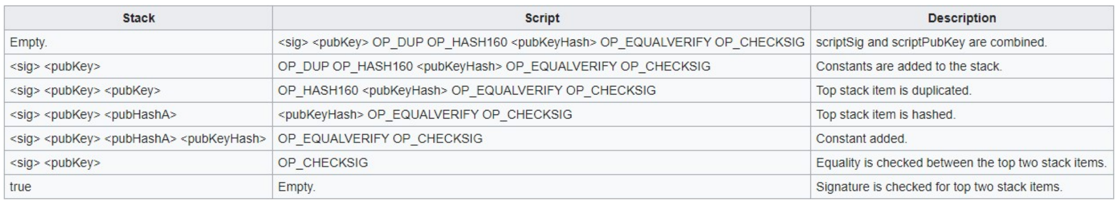
2.2.14 Scripts

As mentioned in the previous two sections, Bitcoin uses a scripting system for transactions named Script. Script is stack-based, does not support loops and is not Turing-complete [67]. Transactions can be provided with certain conditions that have to be met to spend this transaction output. Script is the base for the different transaction types, discussed in section 2.2.13. A script is a sequence of instructions, called Opcodes, from a predefined set that are processed from left to right. There are Opcodes for constants, flow control, stack operations, bitwise operations, arithmetic operations, cryptographic operations and lock times. A transaction is valid if the combined script (transaction input script + transaction output script) produces no errors during its execution. Figure 2.10 shows a script for a P2PKH transaction which we also discussed in 2.2.13. The execution of this script on the stack is shown in figure 2.11. First, the 'scriptSig' (unlocking script of the new transaction input) and the 'scriptPubKey' (locking script of the transaction output that wants to be spent) are combined. Constants are added to the stack in a next step. Third, the top stack item is duplicated and is hashed as a next step. Fifth, constants are added again. Next, equality between the top two stack items is checked. As a last step, the signature for the top two items is controlled. The sript has finished with no errors and so the transaction is processed [67].



```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
scriptSig: <sig> <pubKey>
```

Figure 2.10: Example of a P2PKH transaction. 'scriptPubKey' is the output script of the transaction output that is used for a new transaction input. 'scriptSig' is the input script of the transaction input of the new transaction [67].



Stack	Script	Description
Empty.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig and scriptPubKey are combined.
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Constants are added to the stack.
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is duplicated.
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is hashed.
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Constant added.
<sig> <pubKey>	OP_CHECKSIG	Equality is checked between the top two stack items.
true	Empty.	Signature is checked for top two stack items.

Figure 2.11: Example of a P2PKH stack execution. 'scriptSig' and 'scriptPubKey' commands are combined and put on a stack. The stack is executed from the top to the bottom. A script and thereby a transaction is valid if no error occurs on executing the whole stack command by command [67].

2.2.15 Smart Contracts

As discussed in the previous section 2.2.14, Script is the way of defining transactions and their conditions in Bitcoin. Especially in the field of Ethereum, smart contracts are a fundamental building block. Smart contracts are programs or instructions stored and executed on every node in the network of a DLT. Depending on the underlying DLT, a smart contract can map very simple to very complex business logic. Beside payments, Ethereum can handle complex business logic empowered by the Turing-complete programming language Solidity [68] and the Ethereum Virtual Machine (EVM). Smart contracts have the following properties summarised in [69]:

- **Computer programs** - Smart contracts are simple computer programs.
- **Immutable** - Once deployed, they cannot be changed. To modify a smart contract, a new instance beside the old one has to be deployed.

- **Deterministic** - The result of the execution of a smart contract is the same for every invoking entity with same preconditions and the same state of the blockchain.

- **EVM context** - Smart contracts are operating in a limited context of the own state, the transaction that called them and some information about the most resent blocks.

- **Decentralised world computer** - Smart contracts are executed by every EVM on every node within the network, each operating on the same initial state and producing the same final state.

The concepts of smart contracts were already introduced by Nick Szabo in 1997 [70], who also introduced Bit Gold [71] in 1998, a forerunner of Bitcoin. Smart contracts in Ethereum build the base for decentralised applications, called dApps. DApps can map many kinds of applications, beyond the payment use case. Some further use cases, summarised in [72] are i.e. digital identity, insurance applications, real estate recording, supply chain management, internet of things, voting, tokenization, licensing and intellectual property, certificates, health care data management or energy communities. Listing 2.1 shows an example of a smart contract. This contract can be initialized with a message which can be updated and returned by the respective methods.

```
pragma solidity ^0.5.10;

contract HelloWorld {

    string public message;

    constructor(string memory initMessage) public {
        message = initMessage;
    }

    function setMessage(string memory newMessage) public {
        message = newMessage;
    }

    function getMessage() public returns string {
        return message;
    }
}
```

Listing 2.1: Example of a simple smart contract. The contract can be initialized with a message and there are methods for retrieving and updating the message.

2.2.16 Scriptless Scripts

Another type of conditional payments are scriptless scripts. Scriptless scripts enable off-chain smart contracts to Bitcoin by using Schnorr signatures which will be discussed in section 2.3.1. By making use of signature aggregation, transaction outputs can be conditioned with such signatures. Spending conditions are not enforced by the blockchain, but rather by the parties themselves, therefore it is called off-chain. Scriptless scripts offer additional functionality, privacy and efficiency. They were introduced with MimbleWimble [73], discussed in section 2.3.1.

2.2.17 Limitations

Beside the major advantages, blockchains and DLTs are also limited in other aspects. Some are related to the used consensus mechanism, i.e. to the PoW, but others are related to the distributed approach or the underlying architecture. The major limitations of the actual Bitcoin blockchain are identified in [7, 74], discussed below.

Sustainability One major drawback of the PoW consensus protocol is its dealing with sustainability. PoW consumes a lot of energy for calculating random nonces and block hashes. Bitcoin consumes 73.12 TWh per year which is comparable to the annual power consumption of Austria [75]. A second drawback related to dealing with sustainability is the fact that more and more specific mining hardware is produced, i.e. based on application-specific integrated circuits (ASICs). This particular hardware was developed to create random nonces and calculate block hashes at a very high speed. They are several times faster than normal CPUs or graphic cards on solving this specific task, but cannot be used for something else apart from this task. Theoretically, once Bitcoin or PoW is not relevant anymore, all this mining hardware will be more or less worthless.

Usability The Bitcoin blockchain is difficult to use which is another major drawback. Only a few tools helping developers or even users to use the Bitcoin blockchain conveniently exists. Over the last decade the main focus of research and development within this field has been in developing technical solutions respectively improvements for a bunch of problems and limitations. Usability was not the focus of most of the projects. It is also necessary to work on this area to push ahead the mass adoption of Bitcoin and other crypto currencies and DLT based projects.

Throughput A very big challenge in most of existing DLTs is scalability. As discussed in section 2.2, the PoW consensus mechanism in Bitcoin is designed the way that on average every 10 minutes a new block is mined. This is controlled by the adapted difficulty. The consensus protocol calculates the actual difficulty by the provided hash power within the network in a certain period. So the protocol can control the time between new blocks on the blockchain. At the time of this writing the average transaction per block was about 2,200 [76]. Simply divided, these are about 4 tps. Compared to payment providers this is very low. For example VISA, which is one of the biggest payment providers, processes about 65,000 tps [8]. There is a huge gap for Bitcoin to become a global payment solution in its current form.

Latency As mentioned in section 2.2, the finality of transactions in Bitcoin is probabilistic and increasing over time. One transaction takes about 10 minutes to be inserted to the blockchain. Due to the Bitcoin consensus protocol, it may happen that an inserted transaction will be reverted afterwards, because of the Longest Cain Rule. Each block after the block which contains a certain transaction can be interpreted as a confirmation. A transaction is considered as final if it has six confirmations [77]. In a centralised payment system, like VISA, a transaction is processed and safe within a few seconds. This is another big challenge within this field.

Security Since i.e. in Bitcoin no one has control over the blockchain per design, also no one can revert it, once malicious activities happened contrary to the expectations. It is very unlikely that someone has the computing power to do so, but Bitcoin is vulnerable towards 51%-attack [78, 79]. If an entity controls 51% of the network mining power, it potentially has control over the network and can manipulate transactions, prevent transactions from being processed, performing a double-spend-attack and so on. However, it is not guaranteed to have success on an attack when having 51% of the network hashing power under control. Because of the rise of mining pools, discussed in section 2.2.3, 51%-attacks are feasible. As mentioned in [80], even less than 51% are required to compromise the network.

Size Another drawback is the size of the blockchain, which at the time of this writing is about 240 GB [81]. Five years ago it was about 24 GB. With ongoing mass adoption as a payment solution, the size of the blockchain will reach very high levels making it difficult to store and process it in a distributed and decentralised way, especially on mobile devices.

Fees As already mentioned in section 2.2, nodes earn transaction fees and block rewards for spending their computational power to the Bitcoin network. Those transaction fees are paid by all of the sending participants of all the transactions within a block. The current average transaction fee is about 0.15 USD. At the peak time, at the end of 2017, a transaction cost about 38 USD [82]. In comparison, transaction fees of a credit card institute range from 1.5% to 2.9% of the paid amount [83]. Therefore, in most cases, it is more expensive to send Bitcoin, than fiat money. Sending small amounts in Bitcoin, so-called micro payments, are not economically feasible at the moment. This is another big challenge that has to be solved.

External Data Another main issue with current blockchains is a reliable and deterministic access to external data discussed in section 2.2.9. In general, blockchains are not able to access data from outside the network. If a blockchain application needs access to external data, this happens via interfaces. Every node within the blockchain has to request this data via the interface when processing the application. It is not guaranteed that every node receives the same data from the external source. Depending on the application logic, reaching consensus could become impossible. Blockchain interactions are somehow limited to data from the blockchain itself. The challenge is to make the blockchain requesting external data only once and storing it to its environment.

Interoperability Another major limitation of current blockchains is the interaction or the data exchange with other blockchains. Interacting with the environment or with other blockchains is not enabled by default by most blockchains. Different approaches and technologies to overcome this limitation summarised by M. Borkowski et al. in [84] exists. Interoperability could be simple data exchange, but also complex exchange or transfer of values that opens up non trivial challenges. In section 2.2.7 one solution for interoperability was discussed.

Immutability Another major limitation is the immutability concerning illegal content. In general, immutability is an advantage of blockchains, but if illegal content is stored in the blockchain, it can also become a limitation, because this content cannot be easily removed or deleted. Development effort has to be spent on reliable mechanisms to prevent illegal content from even being stored on the blockchain.

As can be seen, there are many open challenges within this field. A lot of different approaches have been developed to solve current blockchain limitations like scalability and privacy. Approaches are ranging from very simple changes or adaptions to very complex changes or adaptions. Before giving the reader an overview of the different approaches and their known implementations, privacy and security aspects of DLTs will be introduced.

2.3 Privacy & Security

Blockchain data is public and immutable in general. This is an advantage, as well as a disadvantage, depending on the respective viewpoint. Entities and transactions are hidden in general by cryptographic mechanisms. There are several ways for linking Bitcoin addresses to individuals, e.g. if a user publishes an address for donation purposes somewhere. There are also ways for linking addresses to nodes and IP addresses [85]. Once this happened, the complete transaction history of an individual can be traversed to the past. Therefore privacy is very important in such systems. In this section, an introduction into specific privacy and security aspects of Bitcoin, blockchains and DLT in general will be given. First, privacy preserving mechanisms and approaches and second, blockchains and coins that are based on these will be discussed.

2.3.1 Privacy Preserving Mechanisms & Approaches

In Bitcoin a few mechanisms and approaches for improving privacy exist, summarised in [86] and extended by the author of this paper.

Avoiding address reuse is a mechanism where a Bitcoin address is only used once for receiving Bitcoins. A third party cannot link addresses to the same entity in general and so it does not know, how many coins an entity controls. Hierarchical deterministic wallets, discussed in section 2.2.4, are a usage of this method.

Using multiple transactions is another way for improving privacy. Instead of sending all the intended value in a single transaction, the total amount can be sent via multiple transactions particularly from different addresses. A third party cannot easily observe the total amount of sent value when this approach is used.

Coin swap is a mechanism based on the idea of atomic swaps, discussed in 2.2.7. Two parties can perform a coin swap by sending the same amount of coins to a third party, either a trusted one or a trustless script. The third party then swaps the coins and so the transaction graph is broken in general.

Mixing services are another way of breaking the transaction graph. Mixing breaks the graph by receiving Bitcoins from different addresses and sending back different Bitcoins to the sender. A third party cannot link the incoming and outgoing transactions in general.

CoinJoin is a technique to combine several payments into one single transaction. This makes it difficult to third parties to observe, which sender sent which amount to which receiver. CoinJoin can be applied to Bitcoin without any modification of the protocol [87].

Confidential transactions is a protocol which encrypts the amount of a transaction so that it is not publicly available. It is guaranteed that no values are created or destroyed in a transaction without revealing the transaction amount [88].

One-way aggregate signatures is a method for unlinking inputs and outputs of a Bitcoin transaction. This mechanism is based on composite signatures, an improvement of aggregate signatures. Aggregate signatures is a method to merge single signatures into one signature from which the single signatures cannot be recovered anymore. Additionally, composite signatures combines several transactions into one single transaction to hide the links between inputs and outputs of a transaction [89].

Homomorphic encryption is a powerful cryptography allowing certain types of computations to be performed directly on the ciphertext. The decrypted result is identical to first decrypting the ciphertext and then performing the computations on it. Homomorphic encryption allows data to be stored and modified without decrypting it. Ethereum e.g. provides homomorphic encryption on data stored on the blockchain for better control and higher privacy [24].

Attribute-based encryption binds the decryption of a ciphertext on the secret key, but also on the fulfillment of certain attributes, such as time or location, at respectively on which the decryption is allowed [24].

Off-chain transactions are another way of improving privacy in blockchains. Transactions within the payment channel network, so-called off-chain transactions, are not visible to participants that are not involved in a payment in general. A third party can observe the initial conditions of a payment channel as well as its closing conditions on the blockchain, but it cannot observe which transactions are made in detail. Only participants of the payment channel can observe that. E.g. Alice creates a payment channel with Bob and deposits an amount of 1 BTC to the blockchain. This amount is publicly known. After some time Bob closes the channel and the result of the payment channel - Alice owning 0.5 BTC and Bob owning 0.5 BTC - is stored to the blockchain. Now it cannot be derived if only one transaction happened, Alice sent 0.5 BTC to Bob for example, or if more transactions happened, Alice sent 1 BTC to Bob and Bob sent 0.5 BTC back to Alice for example. Details are only known to the participants of the payment channel. Payment channel networks will be discussed in detail in chapter 3.

Dual-Key Stealth Address Protocol (DKSAP) hides the association of a transaction output with a recipient's address, which means it hides the actual destination address of a transaction [90]. The protocol works as following: A receiver creates a pair of public private keys. One pair is created for sending transactions and one is created for scanning transactions to receive funds. Both public keys are only known to the sender and receiver, not to the blockchain. The sender creates a temporary key pair and sends the public key to the receiver. Next, both create a shared secret based on these keys. The sender now creates the destination address of the receiver and sends the payment to it. The receiver can also create this destination address and can spend the funds. To do so, the receiver has to scan all the transactions permanently on the blockchain to be aware of received funds. Actively scanning the blockchain is consuming significant resources. E.g. Monero, discussed in section 2.3.2, makes use of DKSAP to strengthen the privacy of the participants.

MimbleWimble is a privacy preserving protocol published in 2016 by T. E. Jedusor [91]. MimbleWimble is a combination of several methods such as CoinJoin, confidential transactions and one-way aggregate signatures. Besides, MimbleWimble uses a mechanism called transaction cut-through which eliminates redundant transaction information. Spent outputs and their corresponding inputs are removed so that only the very origin and the actual unspent transaction output are remaining. This reduces the size of the blockchain significantly and increases the privacy substantially. Confidential transactions require more computational resources, whereby the throughput is reduced. There is an expected trade-off between scalability and privacy, as mentioned in section 2.2.1.

Unlinkable, confidential and anonymous transactions introduced by Singh et al. in [92] are built on four roles: the sender, the receiver, the verifier and the authority. Sender and receiver are the interacting entities of a transaction. The verifier is an entity that is in charge of verifying the protocol. The developed approach is based on Ethereum, therefore the verifier is a smart contract. The authority is an instance that takes over control of the interacting user, i.e. controlling their identities. The protocol combines several steps and well-known approaches like ring signatures and elliptic curve based schemes. The presented approach hides the sender and the transferred amount, but not the receiver. Due to the complex and formal procedure and the focus of this work, we refer the reader to the paper of Singh et al. for further details.

Privacy-preserving pre-consensus protocol introduced by Yasusaka et al. in [93] hides the sent amount and balances of interacting entities. The protocol is built on two phases: the pre-consensus process and the normal transaction process. In the pre-consensus process, the interacting parties reach consensus about the transaction amount without revealing the individual balances of the entities. Both parties sign a publicly verifiable signature, a third party maintains the signature and every node can verify its correctness. The authors are using well-known approaches like ZKPs, discussed in 2.1, and homomorphic encryption, discussed in this section. Simplified, it is proven that the sender has sufficient currency or assets and the transaction is non-negative. We refer the reader to the paper of Yasusaka et al. for further protocol details due to the focus of the work in hand.

Distributed key generation (DKG) is a protocol where several entities are involved in the key generation or revealing process. A (n,t)-DKG protocol requires a set of n entities to collectively create a secret. Each node stores a share of the secret so that at least t nodes have to collaborate in an asynchronous way to re-create the secret which is the base for the private key. If less than t nodes cooperate on the process, the secret and thereby the private key cannot be created [94]. Torus [95] e.g. is using DKG based on asynchronous verifiable secret sharing [96] to provide frictionless logins for distributed applications on the blockchain based on OAuth accounts at Google and Facebook, to derive their private keys in a secure and trustless way. Thereby Torus contributes to the mainstream adoption which is braked by the inconvenient way of handling private keys.

P2SH transaction outputs provide some kind of privacy until it is spent. As discussed in section 2.2.13, in a P2SH transaction, the hash of a certain script is locked together with the amount of coins. To spend from this output, a script has to be provided that produces the required hash. When the script is provided, it's publicly available and reveals all its details, e.g. who can spend from this output in which way. One mechanism to overcome this fact is a Merkelized Abstract Syntax Tree (MAST). In a MAST all the possibilities to spend from an output are hashed and stored in a merkle tree, discussed in section 2.2.11. Only the merkle root is locked to the output. If a transaction spends from this output, it only has to reveal the one possibility which is used to spend from it, but not all the others. Everyone else can verify that this is valid or not, without needing to know all the other possibilities to spend from this output. With this mechanism the privacy of transactions can be increased significantly [97, 98]. Taproot makes use of MAST and Schnorr signatures which will be discussed in section 2.3.1, to add an extra cooperative possibility for spending from an output. A P2SH is often used for either-or transaction schemes, e.g. either Alice provides a secret and can spend immediately from an output or Bob waits for a certain amount of blocks to spend from an output. Per default there is no cooperative way for spending from it. Taproot overcomes this fact by making use of both concepts mentioned [99].

Shamir secret sharing scheme splits up the secret into several parts which are distributed to several entities. Depending on the construction of the secret sharing scheme, m out of n parties are required to collaborate to restore the original secret. Any set that is smaller than m parties, is not able to restore the secret [100]. This process is similar to the asynchronous verifiable secret sharing scheme. The main difference between both is that asynchronous verifiable secret sharing scheme works also in asynchronous networks. Bitcoin uses Shamir secret sharing scheme to require m out of n parties to agree on a transaction. Conceptually it is similar to multi-signature transactions discussed in section 2.2.13. The main difference is that Shamir secret sharing does not reveal the participating entities. Shamir secret sharing scheme can be used together with a multi-signature transaction scheme to distribute multi-signature private keys. Figures 2.12 and 2.13 show Shamir secret sharing scheme and multi-signature transaction. Both require 2 out of 3 entities to collaborate.

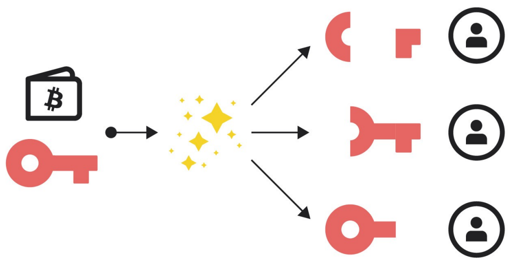


Figure 2.12: In Shamir secret sharing scheme a secret, i.e. a key, is devided by multiple parties. One party alone cannot restore the secret without cooperating with others [100].

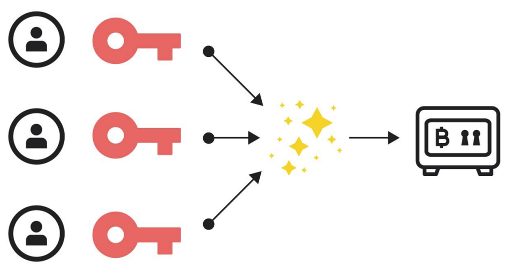


Figure 2.13: In a multi-signature transaction a signature is created by several cooperating parties. One party alone cannot create a respective signature and an observer can see that a signature is created by more than one party [100].

Schnorr Signatures are a different type of signatures, than ECDSA signatures. As discussed in section 2.2.12, new transactions are based on unspent outputs. Those outputs can belong to different private keys. Broadly speaking, Schnorr Signatures are collaboratively created over the sum of different public keys. Figure 2.14 shows a transaction with multiple transaction inputs, each signed with ECDSA. An observer can identify the multi-signature pattern. Figure 2.15 shows a transaction with multiple transaction inputs signed with Schnorr Digital Signature Scheme. A third party cannot observe anymore if the transaction input is a multi-signature input or not. Schnorr signatures are linear, meaning they can be added and subtracted on each other, called signature aggregation. E.g. signature 1 + signature 2 = signature 3, corresponding to public key 1 + public key 2 = public key 3. Signature 3 is valid as well and can only be created by adding signature 1 and 2. By using Schnorr Signatures, the privacy and also the scalability can be improved in Bitcoin and other blockchains. Multi-signature transaction inputs signed with Schnorr Digital Signature Scheme requires less storage, compared to multi-signature transaction inputs signed with ECDSA. Thereby the amount of transactions in a block can be increased.

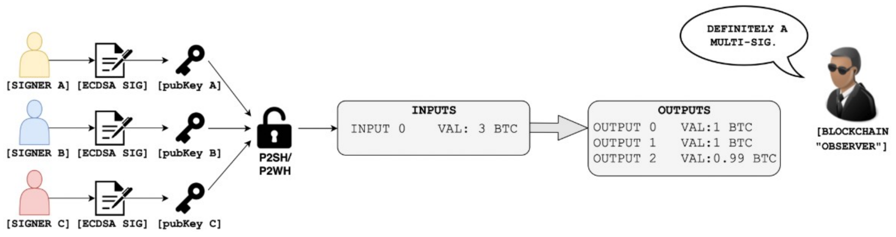


Figure 2.14: In a transaction signed with ECDSA, several parties are required to collaborate on signature creation and an observer can see that a signature was created by more than one party [101].

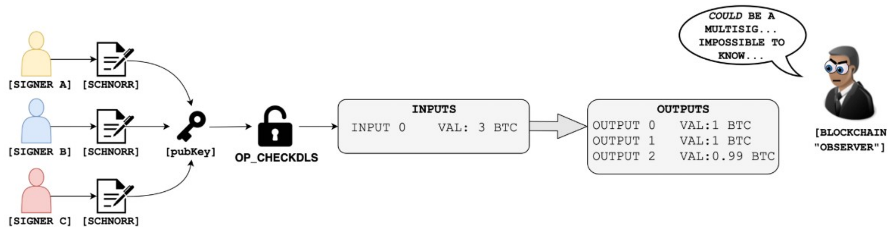


Figure 2.15: In a transaction signed with Schnorr Digital Signature Scheme, several parties are still required to collaborate on signature creation, but an observer cannot see anymore if the transaction was signed by one individual or by several parties [101].

Ring signatures are digital signatures created by a member of a group, where each member has their own key. It is computationally infeasible to determine the entity from the group that created the signature. A transaction message signed with a ring signature can be verified by any group member without revealing the sender, the receiver, the transaction amount and the signatory. Monero was the first blockchain that implemented ring signatures [102, 103]. Monero and further privacy preserving blockchains will be discussed in the next section 2.3.2.

As the reader can see, there are several mechanisms and approaches to protect the privacy and pseudo anonymity. Due to the public access and the immutability of data in a blockchain, privacy protecting mechanisms are very important. This has lead also to the rise of privacy preserving blockchains and coins within the last years. Next some of the most known privacy blockchains and coins will be discussed.

2.3.2 Privacy Preserving Blockchains & Coins

Some of the major privacy preserving blockchains and coins using the discussed privacy preserving methods and approaches are discussed in this chapter.

Monero is a private digital currency focusing on privacy [104]. A transaction in Monero obfuscates the address of the sending entity, the address of the receiving entity, as well as the transacted amount. This is different to Bitcoin or Ethereum where information is publicly visible to everyone. Monero uses ring signatures, discussed in section 2.3.1, to protect the input side of a transaction. The transaction amount is hidden by ring confidential transactions. The sender only reveals as much information as needed by the miners to confirm the transaction. The receiver is protected by stealth addresses, which are one time public keys and therefore make linking more difficult for a third party [105].

Like Monero, Zcash is a privacy-protecting and digital currency [106] focusing on privacy. Zcash is based on zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARK), a special form of a ZKP, in which the prover and the verifier even do not have to interact with each other [107]. ZKPs were discussed in section 2.1. In Zcash there are private (z-addresses) and transparent addresses (t-addresses) and therefore four types of transactions: z-to-z (private), z-to-t (deshielding), t-to-z (shielding) and t-to-t (public). Z-to-z (private) transactions are stored on the public blockchain, so everybody knows that a transaction happened, but addresses and transaction amount are encrypted. Using encryption on the blockchain requires a ZKP to be able to validate transactions. T-to-t (public) transaction are like transactions in Bitcoin. Addresses and the amount sent are visible to everybody on the blockchain. Z-to-t (deshielding) or t-to-z (shielding) transactions reveal either the sender or the receiver and in both cases the transaction amount [108].

Both, Grin [109] and Beam [110] are privacy coins implementing the MimbleWimble protocol discussed in section 2.3. Both projects are promising scalable and confidential transactions. There are two main differences between these projects. Grin is a community-driven project whereas Beam is developed by an enterprise. Both projects use PoW based consensus algorithms designed to be resistant to hardware arms-races. Grin is based on Cuckoo Cycle PoW algorithm [111] and Beam on Equihash PoW algorithm [112].

While Monero, Zcash, Grin and Beam are privacy preserving UTXO based blockchains respectively protocols, BlockMaze is the first privacy preserving account model based blockchain. BlockMaze is built on zk-SNARK and protects transaction amounts and the relationship between sender and receiver. To hide the transaction amount, BlockMaze uses a dual-balance model that splits up the amount into two parts: a plaintext balance and a zero knowledge balance. To hide the relationship between sender and receiver of a transaction, BlockMaze uses a two-step fund transfer procedure. In the first step, the sender creates a commitment on the transferred amount and computes a zero-knowledge transaction to transfer committed funds. Second, the receiver recovers the transfer commitment from the sender and creates a zero-knowledge transaction to deposit funds from the sender [113].

In the last two sections, we introduced the reader into the major aspects of privacy and security in current blockchains and DLT. Next, we will introduce different approaches to overcome the limitation of scaling in blockchains and DLT.

2.4 Scaling Approaches

Different scaling approaches ranging from small adaptions of parameters, over splitting up the network in parts, over changing the way for reaching consensus, over building solutions on top of the blockchain and in this way 'outsourcing' some problems to other networks, to completely new distributed ledger technologies concerning the construction of the underlying architecture exist. In this section, the most relevant ones, starting with the simplest and ending with the most complex approaches will be presented.

2.4.1 Block Size

The simplest approach to scale the transaction throughput is by increasing the block size. The block size limits the amount of transactions that can be stored in one block based on the storage ability. In 2010, the block size was limited to 1 MB by Satoshi Nakamoto [114]. A transaction with one input and two outputs is about 250 bytes [115]. This results in about 4,000 transactions per block at maximum. Increasing the block size will increase the throughput, but it also has some drawbacks, i.e. more data has to be transferred and validated by each node at the same time. Increasing the block size makes it harder for a single node to process one block in a certain period of time.

2.4.2 Segregated Witness

A more complex way to scale the throughput of transactions in a blockchain is to decrease the storage size of a single transaction respectively to outsource some of its information. In August 2017 Segregated Witness (SegWit) was implemented in Bitcoin that separates the witness, which means the signature, of a transaction from the transaction itself [116]. In SegWit a transaction only holds the hash value of the signature which is smaller than the signature itself. The implementation of SegWit resulted in a soft fork, which will be discussed in section 2.2.8. Nodes, which did not activate SegWit, can still process SegWit-transactions, but cannot fully validate transactions based on it. Also the block size was increased by SegWit from 1 MB to about 2 MB [117].

2.4.3 Sharding

Another more complex way of scaling blockchains to a higher level is to use sharding [118]. Sharding divides the network into smaller areas responsible for mining and validating only certain blocks or transactions instead of all blocks and transactions. Shards can be built based on different properties like location, version and so on. Sharding makes the blockchain faster because it acts more in parallel. On the other hand, it also has some drawbacks. The fewer nodes a blockchain maintains, the higher the probability of being successfully performing attacks, i.e. a 51% attack discussed in section 2.2.17. A second important disadvantage are cross-shard transactions. As shards are only aware of transactions within their responsibility, sending transactions between shards is more complex than within a not-sharded blockchain or the same shard. There are synchronous and asynchronous cross-shard transactions. Synchronous cross-shard transactions require both validators on both shards to collaboratively confirm the transaction. Asynchronous cross-shard transactions work in two steps. The receiving shard queries a user account whether the outgoing transaction in the sending shard has enough confirmations. There is a non-zero chance that any of the blocks of one of the shards gets orphaned and therefore invalid. As the reader can see there is again a certain trade-off between scalability and security.

2.4.4 Consensus Protocols

Another way to overcome major limitations of blockchains such as energy consumption or scalability issues, is by using different consensus algorithms in blockchains. As mentioned in section 2.2, the Bitcoin blockchain uses PoW. PoW is a mechanism to harden the possibility for adding new information to the blockchain. To add a new information, e.g. a transaction or a bunch of transactions, a mathematically hard problem has to be solved. In case of PoW, a nonce has to be found which together with the transaction data produces a hash starting with a certain amount of zeros. To find a respective hash, a node can only set a random node and produce the hash of the block data. If the hash fits the requirements, a new block is mined. If not, the node can only set another random node and try again until it finds a required hash or another node finds one. Due to the downsides of this mechanism, such as high energy effort, other consensus protocols have been developed. An overview of the most relevant and important consensus algorithms which are partially summarised in [119] will be given next.

PoS follows a different approach. Nodes controlling a certain amount of a crypto currency or scarce token in a blockchain, are in general allowed to add new information to it. It depends on the concrete implementation how a node that is allowed to add the next block is finally selected. It is mostly based on a random factor on every new block. The major advantages are energy-efficiency, throughput and scalability [120]. One drawback is the nothing-at-stake problem which describes the fact that a vote does not cost anything and so a node can vote for different versions of a blockchain simultaneously. This can be punished by losing all the coins, once it is detected by the protocol for example. Another downside is the exclusion of nodes owning fewer crypto currencies than required to participate in the consensus. This leads to some kind of centralisation based on the amount of money one has. But this is similar to PoW where entities with more money have more performant hardware to mine and outperform those who have fewer financial possibilities. Another drawback in PoS is the disclosure of the identity and the wealth of a participant [121]. Ganesh et al. proposes a privacy preserving PoS protocol in [121]. Ouroboros Crypsinous is another privacy preserving PoS algorithm proposed by Kerber et al. in [122]. E.g. Nxt [123] and Cardano [124] blockchains uses PoS based algorithms for reaching consensus. Cardano call their algorithm Ouroboros which is the first algorithm that is based on formal and peer-reviewed academic research [125]. Ethereum is also planning to switch from PoW to PoS with Ethereum 2.0 [126]. PoS often seems to be the next step in many blockchain projects because it has many advantages compared to PoW, but it also has some limitations such as nothing-at-stake.
Delegated proof-of-stake (dPoS) is an advancement of PoS to involve smaller nodes as well as prevent centralisation. Nodes that are allowed to add new information to the blockchain, are selected by the amount of crypto currency or scarce token they own and of votes from other nodes within the network. In certain cases dPoS can also lead to more centralisation if the voting nodes are malicious. E.g. Eosio [127] blockchain uses dPoS.

Beside PoW and PoS there are a lot of further concepts and ideas for consensus algorithm. E.g. there are proof-of-activity [119], proof-of-burn [119], proof-of-capacity [119], proof-of-authority [128], proof-of-reputation [129], proof-of-energy [130], proof-of-renewable-energy-production [131], proof-of-elapsed-time [132] and proof-of-play [133].

Beside the proof-of-x consensus algorithms, others operating in a different way exist, like cellular consensus and cellular automata [134], fast probabilistic consensus [134, 135], practical byzantine fault tolerance [136, 137] and LibraBFT [138], which is based on HotStuff [139].

As the reader can see, a lot of different consensus mechanisms and ideas, each having advantages as well as drawbacks exist. These lists do not claim to be complete as there also are other consensus mechanisms and protocols. Some of them are only ideas for algorithms which have not yet been implemented and tested. Next, we will discuss how different architectures can scale blockchains to a higher level.

2.4.5 Architecture

An already very complex way for scaling blockchains is to change or adapt the underlying architecture to a more scalable data structure. As mentioned in section 2.2, a blockchain can be interpreted as a linked list of data. Data can only be added at one single point, namely at the end of the list. Beside a linked list, there are also graph based architectures, called directed acyclic graph (DAG) based ledgers, where data can be added at multiple points. Data in a DAG based ledger is organized differently than in a blockchain. A DAG consists of edges and vertices. Edges illustrate the link between transactions (vertices). DLTs offer some advantages compared to blockchains, but also struggle with some limitations, as discussed in section 2.2.17, and the trilemma, mentioned in section 2.6. IOTA [140] for example built their DLT on the Tangle [141] that is based on a DAG. Figure 2.16 shows the conceptional view of the Tangle. IOTA claims to be the payment solution especially for the Internet of Things, which will request high frequent micro payments [142]. In the Tangle every transaction is referencing and therefore validating two other transactions [143]. Each block in figure 2.16 represents a single transaction. One main characteristic of the Tangle is that the transaction speed increases with the number of nodes and transactions. IOTA can process about 1,500 tps [144]. This is a huge improvement compared to the Bitcoin blockchain. Another main improvement is that transactions are feeless. One major drawback of IOTA is the coordinator which is currently active due to security reasons and the limited amount of nodes within the network. The coordinator is a central authority operated by the IOTA Foundation that validates the state of the network making the Tangle not decentralised at the moment. Once the network reaches a certain size, IOTA plan to shutdown the coordinator [145]. From that on the Tangle should operate in a distributed and also in a decentralised way. A second drawback is the security of the Tangle. While in a blockchain, as mentioned in section 2.2.17, it is in general necessary to control 51%, or even less, of the hash computing power to act malicious attacking the Tangle only requires 34% of the computing power [78].
Beside IOTA, some further concepts and projects are built on top of a DAG based data structure, e.g. Hashgraph by Hedera [146], Nano's Block Lattice [147] and an Account-wise-transaction-chain (AWTC) that was introduced by Locus Chain [148], just to name a few.

Beside DAG based DLTs, there are further data structures used to overcome scaling

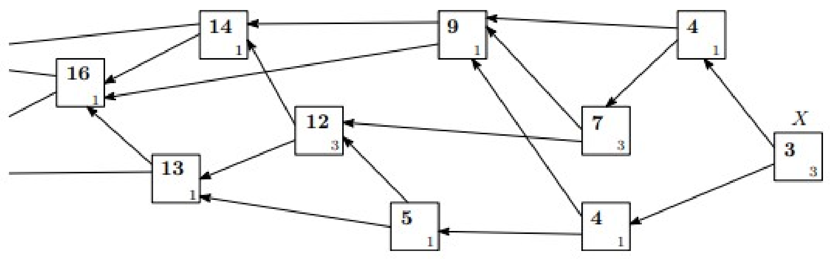


Figure 2.16: In the Tangle, each transaction references and thereby validates two former transactions. There is more than one end where a new transaction can be attached [141].

limitations. A blockchain can be interpreted as a linked list of data. To reduce the search effort in a linked list, it can be extended to a skip list. A skip graph is a decentralised and distributed type of a skip list [149]. LightChain [25] for example is a peer-to-peer distributed hash table overlay based on a skip graph and it targets a communication and storage efficient architecture.

Also, we would like to give the reader an idea about the variety of different solutions to overcome limitations of blockchains. In Holochain [150] for example every node maintains its ledger and a copy of validation rules of other ledgers from other nodes, instead of one public distributed and replicated ledger that is accessed and maintained by all nodes within the network [151, 152]. Polkadot [153] is a project to overcome the limitation of interoperability between blockchains, as we discussed in sections 2.2.17 and 2.2.7, consisting of multiple chains and bridges across them [154].
Hyperledger Fabric [155] is a permissioned blockchain. Common blockchains are based on an order-execute scheme, while Hyperledger Fabric is based on an execute-order-validation scheme. A node in an order-execute blockchain is first ordering the transactions by the consensus algorithm and then executing them on its blockchain. Every node executes all transactions which slows the blockchain down and rises issues with non-deterministic executions, e.g resulting in forks. An execute-order-validation scheme based blockchain operates in three steps. First, transactions are executed by several peers producing a proof of execution. A transaction might not be executed by every peer. Second, transactions, results and proof of executions are ordered and finally validated. If they are valid, they are added to the blockchain [156, 157].

In this section, we discussed different architectural approaches to overcome scaling and other limitations of DLTs. The next section will introduce a promising scaling approach for blockchains operating on top of the blockchain, called second layer networks.

2.4.6 Second Layer Networks

Another complex way to overcome major limitations of blockchains that we will discuss in this paper, is by using networks on top of the blockchain. These networks are called off-chain networks or second layer networks where the blockchain builds the first layer respectively the base. Those networks are independent and use existing blockchains for anchoring data and states. In this section, different types of second layer networks will be discussed. In chapter 3, we will introduce this type of scaling solution in more detail as it builds the base for this thesis.

The first second layer network introduced is a payment channel network. Payment channel networks enable micro payments with low fees and low latency. They use the underlying blockchain for anchoring. The procedure is as following: Two participants who would like to do several transactions together, agree on that and set up a payment channel. This is done by depositing a certain amount of crypto currencies to the corresponding blockchain. This amount can be interpreted as a credit which can be transferred at maximum and cannot be exceeded without locking additional funds. So the blockchain knows the initial conditions. Subsequently, transactions happen within the payment channel and are not known to the underlying blockchain and therefore not known to the public. Each participant maintains his or her actual balance on his or her own. When a party decides to close the channel because both finished exchanging values, it commits the actual balances to the blockchain. The blockchain validates the state and only the sum of all the single transactions processed in the channel is stored to the blockchain and consequently public again. Not involving the blockchain and its mechanism on every single transaction allows a very high transaction speed at very low fees. One is talking about a payment channel network because a payment can even be processed via other channels within the network, if two parties are not connected directly via a special channel. As the reader can imagine, there is a certain trade-off between scalability and security, due to the time range of being off-chain. Several mechanisms ensure that the payment channel network is reliable nevertheless. The Lightning Network is a payment channel network implementation for Bitcoin which will be discussed in more detail in chapter 3. Beside an implementation for Bitcoin, there also exists one for Ethereum called the Raiden Network [158].
Beside the mentioned payment channels, there is also the concept of virtual payment channels. Virtual payment channels is a technique that avoids involvement of intermediaries for each payment. Two interacting parties get directly (virtually) linked via the intermediary without involving it in each payment. The intermediary can be seen as payment hub that locks funds for a defined period for arbitrary channel routing [15] and not for specific payments. A virtual payment channel reduces latency and costs and improves privacy. Intermediaries cannot observe the amount of funds that is transferred, even if it is transferred through them. Perun for example is an off-chain payment channel network which introduced the concept of virtual payment channels [159].
In payment channel networks, there also exists the concept of payment channel hubs. Payment channel hubs are central nodes within a payment channel network which acts as a bridge between nodes. Therefore a node maintains many payment channels with other nodes to reduce collateral lockup costs and number of intermediates and simplify routing complexity. A payment channel network with multiple payment channel hubs should reduce the overall path length and routing fees [15]. On the other hand, centralization is

also a potential attack vector within the payment channel network using hubs.

Another type of second layer network are state channel networks. State channels generalise the concepts of payment channels to enable the execution of any state transition, beside the transition of assets. Therefore payment channels are a special type of state channels, bounded to the context of payments. State channel networks can be built on top of smart contract platforms like Ethereum because typically they involve two smart contracts on state transitions. One smart contract maintains the state channel itself and the second smart contract executes the respective application [15]. For instance playing a game can be implemented with state channels with every move being a state on which both participants have to agree. At the end,there is one final state that discloses the winner of the game.

A third type of second layer networks are commit-chains. Commit-chains bring permissionless blockchains and centralised but untrusted operators together. Commit-chains are maintained by one node which acts as an intermediate for transactions. The operator of the commit-chain maintains the communication between interacting parties and is responsible for collecting commit-chain transactions from the users and periodically anchoring a commitment of all transactions to the parent blockchain [15, 160]. Plasma e.g. is a specification and framework for a commit-chain based on Ethereum which is constructed on the UTXO model. Plasma Cash is an improved version of Plasma, which enables also non-fungible tokens by making funds independet and unique [161, 160].

The last type of second layer networks we will mention in this paper, are side-chains. Side-chains are separated systems with own consensus algorithms [15] which are operating beside the corresponding blockchain and not on top of it. Beside scaling capabilities, side-chains are also intended to improve interoperability. Side-chains are two-way pegged which enables assets to be transferred between different blockchains [162]. Assets that should be transferred are therefore locked in an output address. Once they are locked and this state is confirmed with a certain probability on the blockchain, the side-chain gets notified and the corresponding amount of assets are released on the side-chain for further transfers. Moving assets from the side-chain back to the blockchain works in the same way [163].
Rootstock [164] e.g. is a side-chain project which enables smart contracts to Bitcoin. Therefore Rootstock forked the EVM and modified it. The Rootstock Virtual Machine is compatible with Ethereum smart contracts and other development and testing tools for Ethereum and Solidity, which makes it powerful in general. Smart Bitcoin is the native token within Rootstock. Rootstock scales up to about 100 tps while still being decentralised through to merged-mining. Merged-mining enables miners to mine both, Bitcoin and Smart Bitcoin at the same time with the same hardware. Rootstock is therefore PoW based and makes use of fraud proofs, probabilistic verification and sharding [165, 163].
Another example is the Liquid Network [166], another side-chain project for the Bitcoin blockchain. The Liquid Network offers faster exchange of assets, higher efficiency and a better privacy [167]. The Liquid Network makes use of the L-BTC which is the

complement to BTC. There are two roles within the Liquid Network: block signers and watchmen. A block is created every minute by block signers in a round-robin fashion and has to be signed by at least two-thirds of all block signers to be valid. Block signers do not sign blocks that would cause a reorganization of more than one block. That is why transactions can be considered as final once they received two confirmations. Watchmen are responsible for pegging funds in, meaning moving BTC to the Liquid Network, or for pegging out, meaning moving L-BTC to the Bitcoin network [168].

In this section, we have discussed different scaling approaches based on second layer networks. A second layer network, especially a payment channel network, is building the base for this paper. The next section, will give some ideas about threats and critical perspectives on blockchains and DLTs in general.

2.5 Threats & Critical Perspectives

DLTs offer big advantages, but on the other hand, they also struggle with major technical limitations. In this section, some major threats for DLT based projects and crypto currencies concerning economy and technology will be discussed and critical perspectives on this technology will be mentioned.

The first threat for Bitcoin and co. we would like to mention is regulation concerning their mass adoption. Some countries prohibited the use and mining of Bitcoin in history and even nowadays. China for instance banned Bitcoin transactions in 2013 and both exchanges and initial coin offering (ICO)s in 2017. In 2019 China opened its perspective on Bitcoin and other DLT based projects to be more attractive for innovation [169]. It will be interesting to see how states are reacting on the undermining of their power concerning governmental currencies. On the other hand, regulation can also be beneficial for the mass adoption of these technologies if the legal framework is constructed positively. Not exactly knowing what is allowed and what is prohibited, can harm innovation in this field in the same way as negative constructed regulation. Liechtenstein e.g. is very innovative and open-minded concerning this technology and its possibilities, as we discussed in section 2.2.6.
The second threat we would like to mention is quantum computing. Quantum computing seems to be the next evolutionary step in computing and it is, in some parts, a threat for DLTs. But not only for blockchain and DAG based projects and applications, it is a threat for some mechanisms in cryptography itself. Quantum computing will not break every cryptographic mechanism. Hash functions for example are considered as more quantum resistant than ECDSA. That means, a quantum computer will be able to compute the private key based on its corresponding public key. Shor's algorithm can be modified to solve the discrete logarithm problem on elliptic curves. Quantum computers will not be able to compute a nonce based on a certain hash in economic feasible time if the output space is large enough. E.g. SHA-256 requires 2^{256} steps on a classical computer, but 'only' 2^{128} steps on a quantum computer, which is still a very large amount of steps. Recently, Google has announced that they successfully launched

a quantum computer with 53 qubits in a laboratory environment. Optimistic science estimates that about 1,000 qubits are necessary to break 160-bit ECDSA. This seems to be a huge gap, but history showed that computational power has increased almost exponentially within the last couple of decades. So, building a quantum computer with 1,000 qubits and beyond will most probably not take many decades from now and one has to consider post-quantum cryptography when designing and implementing DLT based systems today [170, 171].

After having discussed two threats on DLTs, we would also like to mention critical perspectives on this technology. Edmund-Philipp Schuster describes and argues a sceptical point of view compared to most of the other commonly positive, views concerning the adoption and impact of blockchain based applications and processes within the next years in his paper [172]. The author claims that the enthusiasm for and the understanding of the technology are, at best, orthogonal features. From a legal point of view, it is unlikely that smart contracts will remove e.g. lawyers and courts, because there are so many different scenarios which cannot be handled by smart contracts only in a reliable way. Next, the legal framework of nations has to be fundamentally changed to support blockchain technology e.g. concerning assets, wherein good operating states are not interested in per default. The author argues that the interaction between the blockchain assets and the real world is non-trivial and if it is supported efficiently and reliably, all the benefits from a blockchain will be gone. E. Schuster concludes his argumentation that there is no trust problem into well-known, existing and widely used technologies and frameworks and that cost-benefits-analysis is often based on flawed reasoning.
Besides, Nouriel Roubini, a well-known professor of economics, and Warren Buffett, a well-known investor, have also a critical view on Bitcoin and co. Nouriel Roubini said, 'Bitcoin is the mother of all scams' and 'blockchain is most over-hyped and least useful technology in human history' [173]. Warren Buffett called Bitcoin a 'gambling device with a lot frauds connected to it' and 'rat poison squared'. 'It doesn't do anything, it just sits there', he stated [174].

As the reader can see, not everybody has a positive attitude towards Bitcoin and co. and there are threats concerning the mass adoption of DLT based projects. Before going on to the next chapter, the ideas and concepts discussed in chapter 1 will be summarised.

2.6 Summary

In this chapter, the reader has been introduced into the most relevant aspects of DLTs concerning our work. We started with cryptographic primitives, passed on to blockchain fundamentals, described privacy and security aspects, introduced different scaling approaches and also mentioned threats and critical perspectives on DLT. As discussed at the beginning of this paper, Bitcoin and other blockchains have big advantages, as well as some major drawbacks slowing down mass adoption. To overcome the scaling and privacy limitation, second layer networks arose among other solutions. In the next chapter, we will introduce payments and channels and the Lightning Network in detail, before we introducing in chapter 4 how data across different layers can be linked and enriched by information from a cryptoasset analytics tool.

CHAPTER 3

Lightning Network

The Lightning Network is a second layer network, especially a payment channel network, on top of Bitcoin. It is intended to overcome some of the major limitations of the Bitcoin blockchain such as low transaction throughput, high latency and high fees. The idea of the Lightning Network is to reduce the amount of transactions on the Bitcoin blockchain. This means, important transactions are stored on the blockchain and other transactions are handled and propagated off-chain. By not involving the blockchain and its consensus algorithm on every single interaction, one can reach higher throughputs at lower fees. Figure 3.1 shows the conceptual relation between the Lightning Network and the Bitcoin blockchain.

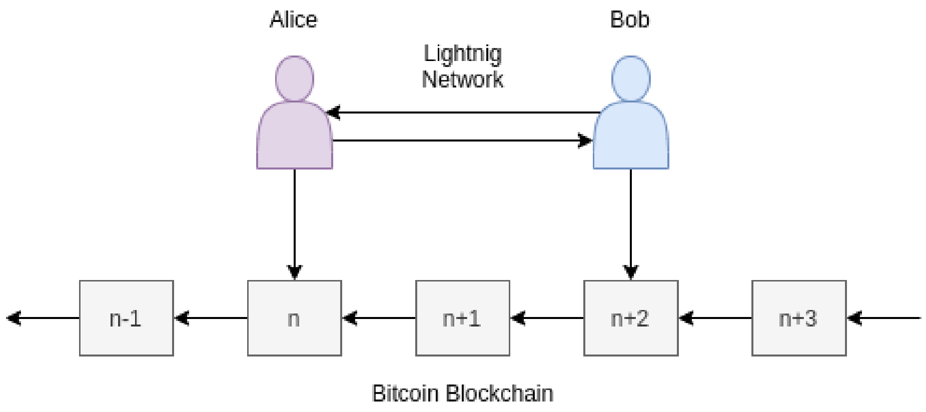


Figure 3.1: In the Lightning Network, two interacting participants are processing transactions off-chain, meaning not involving the blockchain. They only involve the blockchain at the beginning and at the end of processing transactions together.

The Lightning Network consits of nodes, each representing a participant of the network. If two parties, e.g. Alice and Bob, want to exchange Bitcoins via the Lightning Network, they create a payment channel by locking certain amounts of Bitcoins on the blockchain which can be transferred to the counterparty at maximum. This is achieved by sending a double-signed funding transaction to the blockchain. After the funding transaction is confirmed, both can exchange Bitcoins through the created channel. After transactions are done, Alice or Bob can close the payment channel by sending the settlement transaction to the network. Once the channel is closed, the sum of all the off-chain transactions is built and stored to the blockchain. This reduces the amount of blockchain transactions drastically if many transactions happen between both interacting parties.

To illustrate the example in more detail, we will use the following example. Alice opens a payment channel with Bob by locking 10 BTC. Bob locks 0 BTC. Alice sends 5 BTC to Bob and Bob sends 3 BTC back to Alice. Alice again sends 5 BTC to Bob. If one party now closes the payment channel, the result of these three transactions is stored to the Blockchain. Alice owns 3 BTC and Bob controls 7 BTC after these three transactions. On the blockchain only two, not three transactions happened. We will discuss later on why not only one transaction happened on the blockchain. This consumes fewer resources and involves the consensus algorithm less often. Therefore off-chain transactions are cheaper concerning transaction fees and they happen instantly. As the reader can see, the more transactions happen, the fewer proportional resources from the blockchain are consumed.

In this chapter, the focus will be on a detailed overview of channels and payments and the Lightning Network. Limitations and challenges, as well as security and privacy aspects will be discussed and enhancement approaches and analysis done within this field will be introduced. The chapter will be completed by critical perspectives and a precise summary.

3.1 Channels & Payments

Before explaining the details of the Lightning Network, the focus will be put on building blocks of channels and payments in this section. Channel types, HTLCs, single-hop and multi-hop payments as well as routing types of multi-hop payments will be looked at.

3.1.1 Public & Private Channels

In general, a payment channel can either be public or private. In Lightning Network, public channels are known to every node within the network and private channels are only known to the node itself and the neighboring nodes. Public channels are also used to route payments across the network, i.e. multi-hop payments, while private channels cannot be used for payment routing in source routing based networks but to protect the privacy between two participants [175]. We will discuss multi-hop payments and different routing types later in this section. Next, we will introduce HTLCs, a very important concept in payment channel networks.

3.1.2 Hash Time Locked Contracts

HTLCs are a fundamental building block in payment channel networks such as the Lightning Network. A hash time locked contract (HTLC) is used when a payment is routed through more than one channel. Alice and Carol e.g. want to exchange values within a payment channel. They do not have a direct channel, but Bob has a channel with both Alice and Carol. So Alice can use Bob to send a payment to Carol. To do so, she and also Bob make use of an HTLC. We will take a closer look on multi-hop payments later in this section. An HTLC guarantees that locked values are released either by presenting a secret that produces a certain hash or by waiting until the specified amount of time has passed. Time within the Bitcoin blockchain is measured by using block heights, e.g. 1,000 blocks in Bitcoin is a time range of about 1 week (1,000 blocks * 10 minutes per block = 10,000 minutes = 7 days).

The workflow is as following: Carol creates a random secret and produces its SHA-256 hash. She passes the hash on to Alice who locks the amount to send and also some fees for Bob in an HTLC. The HTLC transfers the amount back to her after 1,000 blocks are created or transfers the amount to Bob if he can provide a secret that produces the stored hash. Bob does the same with Carol, but instead he only locks the amount to send and does not add any fees for Carol. Second, he chooses a block height lower than the one in the HTLC with Alice. If Carol now presents the secret to the HTLC with Bob, the produced hash matches the stored hash and the HTLC with Bob releases the amount to Carol. Next, Bob presents the same secret to the HTLC with Alice and also this HTLC releases the amount to Bob. Bob earned a small fee as an incentive for enabling the payment through his channel [176].

We discussed channel types and HTLCs in the sections above and will explain single-hop and multi-hop payments in detail next.

3.1.3 Single-hop Payment Channel

A payment channel between two participants is created by a funding and closed by a closing transaction. Updates of the payment channel are achieved by off-chain commitment transactions. The conceptual lifecycle of a single-hop payment channel based on an example, introduced in [177] will be explained next.

Alice creates a payment channel with Bob. While doing so, Alice locks 10 mBTC to the channel. At the moment, payment channels are single funded in LND [178], an implementation of the Lightning Network which we will discuss in section 3.2. In future versions of LND, payment channels are intended to be dual-funded [179]. The creation of the payment channel causes three transactions: A funding transaction, which is sent to the Bitcoin blockchain, and two commitment transactions, one for Alice and one for Bob. Each commitment transaction holds the amount of Bitcoins one owns, a randomly generated secret, its hash and the hash of the secret of the channel counterparty. The commitment transaction can be interpreted as the current balance sheet. Figure 3.2 is visualizing the funding transaction. After a few confirmations of the funding transaction on the blockchain, the payment channel can be considered as opened.

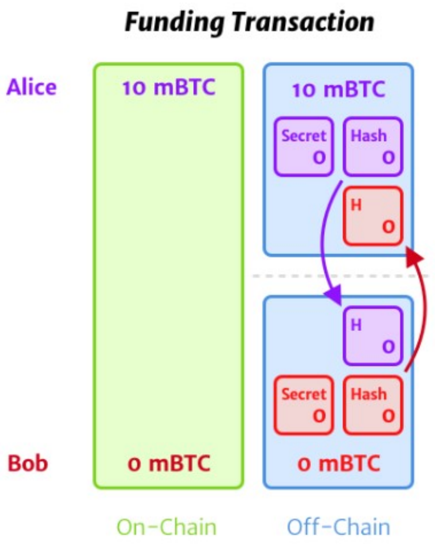


Figure 3.2: A funding transaction of the payment channel between Alice and Bob. The payment channel is funded with 10 mBTC by Alice [177].

Next, Bob sends Alice an invoice about 4 mBTC. She pays the invoice and two new commitment transactions are created. Each payment channel participant creates a new secret and its hash. Each participant sends the created hash and the secret of the past commitment transaction to the counterparty. We will go further in details about the secret and the its exchanged hash later when closing the payment channel. Both new commitment transactions are also holding the new balances of Alice and Bob. Figure 3.3 shows the new commitment transactions. Alice's new state is: 4 mBTC for Bob and either 6 mBTC for Alice after 1,000 blocks or 6 mBTC for Bob, provided he knows Alice's secret. Bob's new state is: 6 mBTC for Alice and either 4 mBTC for Bob after 1,000 blocks or 4 mBTC for Alice if she knows Bob's secret. The reason why there is a block amount limitation in some cases will be pointed out later in this section.

Let us consider the same payment again. Bob sends Alice another invoice for 4 mBTC that she pays. Figure 3.4 shows the new commitment transaction. Alice's new state is: 8 mBTC for Bob and either 2 mBTC for Alice after 1,000 blocks or 2 mBTC for Bob, on condition he knows Alice's secret. Bob's new state is: 2 mBTC for Alice and either 8 mBTC for Bob after 1,000 blocks or 8 mBTC for Alice if she knows Bob's secret.

Let us now say Alice and Bob are done with exchanging Bitcoins and want to close the channel. There are three ways of closing a channel: mutual (cooperative), forced

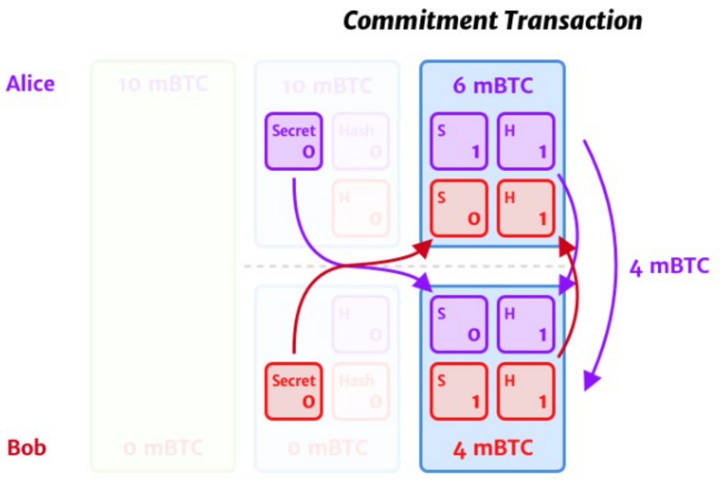


Figure 3.3: A commitment transaction after Alice sent 4 mBTC to Bob. Alice owns 6 mBTC and Bob owns 4 mBTC [177].

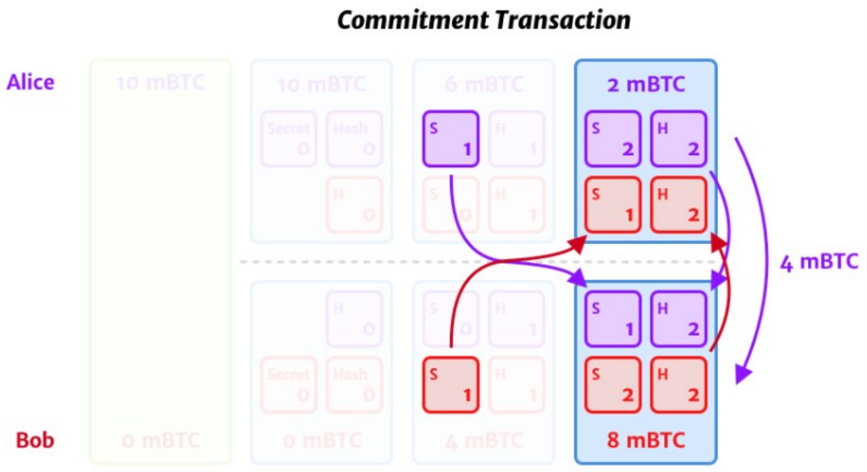


Figure 3.4: A commitment transaction after Alice sent another 4 mBTC to Bob. Alice owns 2 mBTC and Bob owns 8 mBTC [177].

(uncooperative) and false (uncooperative) close.

In a mutual close, both parties send their latest commitment transaction to the network. In that case Alice receives 2 mBTC and Bob 8 mBTC. After some confirmations of the respective transaction on the blockchain, this can be interpreted as final which is the intended scenario in payment channels. Figure 3.5 shows a mutual close of the payment channel between Alice and Bob.

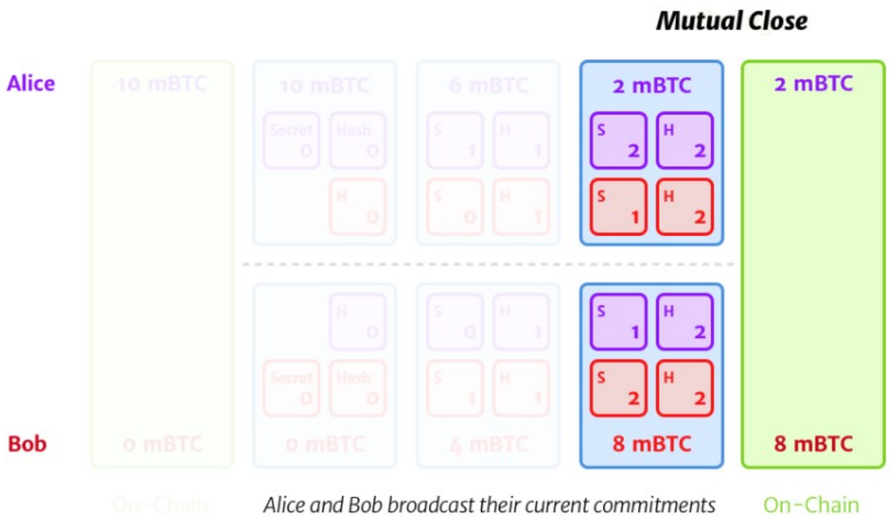


Figure 3.5: A mutual closing transaction of a payment channel between Alice and Bob. Alice owns 2 mBTC and Bob owns 8 mBTC [177].

In a forced close, only one party sends his or her latest commitment transaction to the network. If Alice e.g. disappeared and Bob sends his latest commitment transaction, Alice receives 2 mBTC immediately and Bob has to wait for 1,000 block, to receive 8 mBTC. This is necessary for Alice to come back and revoke this commitment transaction, if Bob acted maliciously and did not send the latest commitment transaction. Figure 3.6 shows the forced closing transaction.

In a false close, one party sends not the latest but an earlier commitment transaction where the closing party had a higher balance than in the latest commitment transaction. If Alice sends an earlier commitment transaction, Bob immediately receives 8 mBTC. As Bob knows Alice's secrets from the previous commitment transactions, he can claim the remaining 2 mBTC from Alice by presenting the secret to the sent commitment transaction. In that case Alice gets punished for malicious behaviour. This mechanism ensures that if a counterparty of a payment channel acts maliciously, the honest counterparty gets all the funds from the payment channel. Figures 3.7 and 3.8 show Alice's false closing transaction and Bob's breach remedy transaction.

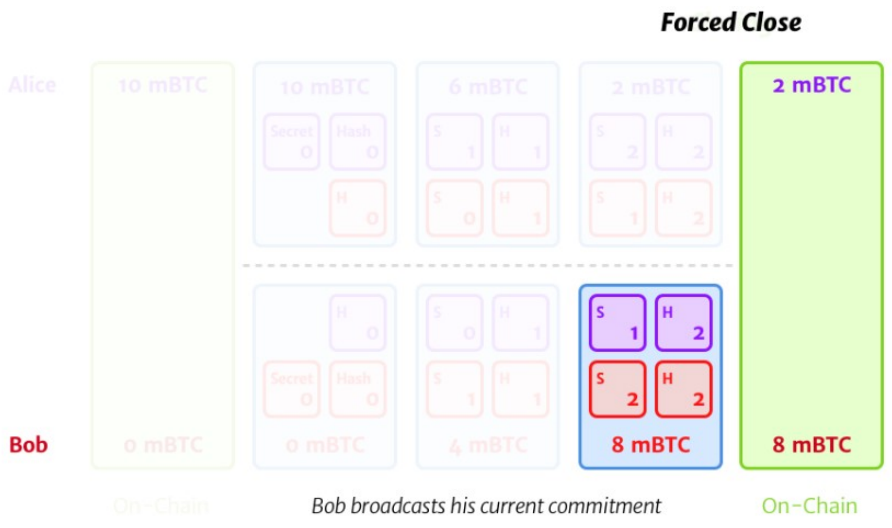


Figure 3.6: A forced closing transaction of a payment channel between Alice and Bob by Bob. Alice receives 2 mBTC immediately and Bob has to wait for 1,000 block to receive 8 mBTC [177].

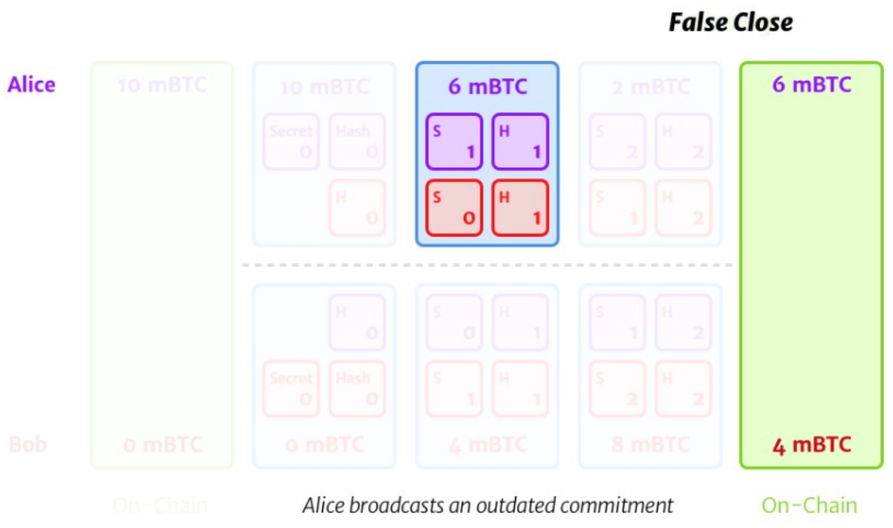


Figure 3.7: A false closing transaction of a payment channel between Alice and Bob by Alice. Bob receives 4 mBTC immediately and Alice has to wait for 1,000 blocks to receive 6 mBTC [177].

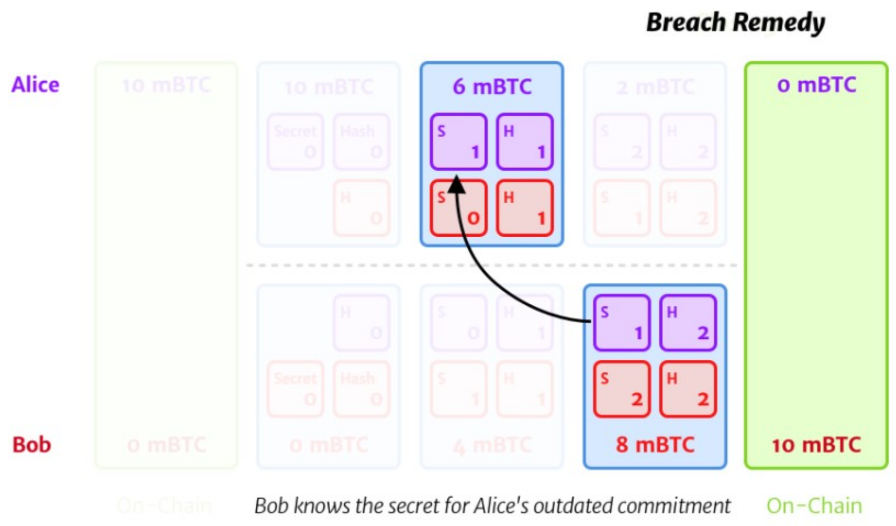


Figure 3.8: A breach remedy transaction from Bob within the 1,000 blocks that reveals the false closing transaction by Alice. Bob now receives all the remaining funds from the channel, resulting in Bob owning 10 mBTC and Alice owning 0 mBTC [177].

Commitment transactions are more complex than illustrated in this example. We will explain the workflow and the building blocks of commitment transactions in more detail in section 3.2.3. A payment can also be routed via others within the network if no direct channel between to participants exists, who want to interact. This type of payment channel will be demonstrated in the next section.

3.1.4 Multi-hop Payment Channel

Payments between two participants can even be achieved if those two parties do not have a direct payment channel, but a route through the payment channel network can be found from one participant to another. The lifecycle of a multi-hop payment channel based on an example, introduced in [180] and slightly modified to explain the workflow more accurately will no be explained.

Let us consider the following example: Alice wants to transfer Bitcoins to Carol. Figure 3.9 shows the initial state of the payment channel network. Alice wants to transfer 4 mBTC to Carol, but Alice only owns 2 mBTC in the channel between Carol and herself. The payment can be routed through the channel with Bob who also has a channel with Alice. Both channels have to have sufficient funds to route a payment because funds cannot simply be transferred from one channel to another. Also the payment flow is in reverse order, which we will explain later. In this example, both channels are sufficiently funded to proceed the intended payment of 4 mBTC from Alice to Carol via Bob.

Carol creates a secret and sends its SHA-256 hash to Alice who locks 4 mBTC together with the hash of the secret in an HTLC to Bob. As mentioned in section 3.2, the HTLC

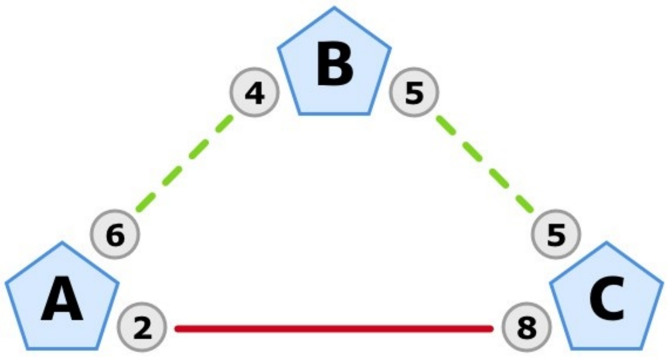


Figure 3.9: The initial state of the payment channel network, slightly adapted from [180]. E.g. A has a channel with B. A can send 6 mBTC at the maximum to B and B at the maximum 4 mBTC to A.

transfers 4 mBTC to him, if he knows a certain secret that produces the locked hash. Bob does the same with Carol. Once all the HTLC along the payment route are set up, the payment flow can begin. Carol is presenting the secret to the HTLC with Bob who releases the locked 4 mBTC to her. Figure 3.10 shows the state of the payment channel network after Carol claimed 4 mBTC from Bob.

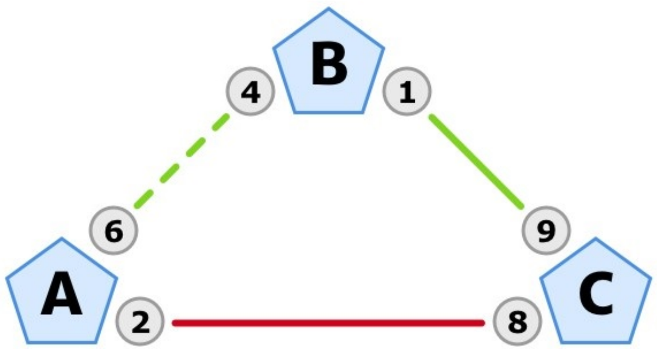


Figure 3.10: The new state of the payment channel network after Carol claimed 4 mBTC from Bob, slightly adapted from [180].

As Bob also knows the secret now, he can provide it to the HTLC with Alice that releases 5 mBTC to Bob. The payment is done. Figure 3.11 shows the final state of the payment channel network. As can be seen, Bob now has 10mBTC in total whereas previously he had 9 mBTC in total. He earned 1 mBTC as a fee for providing a routed payment through him as an incentive for doing so. Since funds cannot be transferred from one to another channel in real, Bob now has different channel balances with his two counterparties than before the payment routed through him.

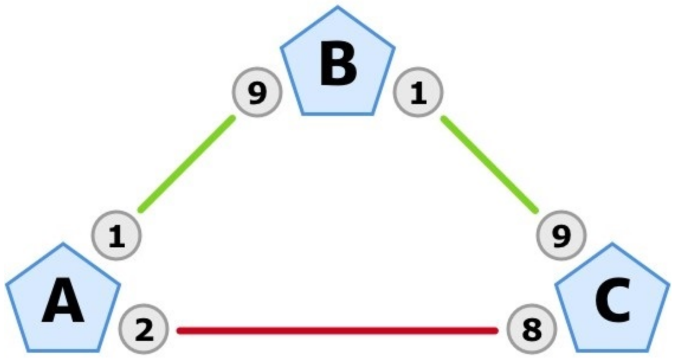


Figure 3.11: The new state of the payment channel network after Bob claimed 5 mBTC from Alice, slightly adapted from [180].

However, in this example the payment could also be split into two payments: 2 mBTC from Alice directly to Carol and 2 mBTC from Alice to Carol via Bob. This could reduce the fees for Alice if they were calculated based on the transferred amount which is currently not the case in LND. This approach is called atomic multi-path payments and is discussed together with further optimizations and improvements in section 3.5.

As the reader can imagine, in very large payment networks it is not easy to calculate the most efficient route from a source of a payment to its destination. Therefore different routing algorithms exist which will be discussed next.

3.1.5 Payment Routing

As mentioned in the last section, efficient and reliable payment routing is a major building block in payment channel networks to offer multi-hop payments. Gudgeon et al. summarise in [15] key aspects of routing in payment channels, as well as investigating and comparing different routing algorithms and mechanisms. A routing algorithm should provide the following properties:

- **Effectiveness** - The algorithm should find a path that maximizes the probability of success of a payment even if the balances of the channels are changing.
- **Efficiency** - Overhead of path finding should be low and topology changes should entail low update overhead costs.
- **Scalability** - A routing algorithm should be effective and efficient, even in very large payment channel networks.
- **Cost-Effectiveness** - Paths with low transaction fees should be found by the routing algorithm.

- **Privacy** - Path finding should be successful, even without disclosing involved parties, transactions values and further private data.

Depending on the knowledge of every node about the topology of the network, there are different routing approaches. One can distinguish between global and local routing. With global routing, every node maintains a state of the complete network containing all nodes and at least all public channels. With local routing a node is only aware of neighboring nodes and channels. A more detailed look at both types is following.

Global routing allows a node to calculate all possible routes from a source to a destination of a payment. Lightning Network and Raiden Network use source routing in which the source of a payment is calculating and selecting the payment route to the destination. Source routing takes no channel balances along the path into consideration. Due to this fact, it could happen that a payment turns bidirectional payment channels implicitly into unidirectional channels by selecting channels with low balances along the payment route, or prevent other payments to be routed through specific channels. Besides, in global routing there is more effort to update all nodes within the network about new nodes and channels. Updating the network with this information also reduces the privacy of the nodes and channels discussed in chapter 4. SpiderNetwork [181] e.g. is an improvement of local routing, which introduces three major modifications: routes are getting biased concerning their optimization of channel balances along the route, routes are including active channel rebalancing and routing is relying on packet switched networks to split payments into several parts and route each of them individually [15].

In local routing a source of a payment cannot calculate the complete payment path to its destination because the knowledge of the network for each node is restricted to the neighboring nodes and channels. Local routing algorithms use well known concepts, like network embeddings, landmark routing, flow algorithms and distributed hash tables to successfully route a payment from a source to the destination by only being aware of the local environment [15]. There are several routing algorithms which make use of these concepts, as summarised in [15].

In the above sections, building blocks and concepts of payment channels have been explained and now the focus will be put on details concerning the Lightning Network.

3.2 Lightning Network Fundamentals

In this section, we will introduce some details on the Lightning Network, concentrating on the description of different implementations of the Lightning Network, giving some key numbers on the network and explaining commitment transactions in more detail.

3.2.1 Implementations

Based on different programming languages, different implementations of the Lightning Network exist. Here is a short overview of the most important ones, listed in [182]:

- LND - Lightning Network Daemon implemented in Go [178]
- eclair - Scala based implementation of the Lightning Network [183]
- lit - Lightning Network node in Go [184]
- c-lightning - Lightning Network implementation in C [185]
- rust-lightning - Lightning Network implementation in Rust [186]
- lightning-onion - Onion routed micropayments for the Lightning Network implemented in Go [187]
- ptarmigan - C++ BOLT-Compliant Lightning Network implementation [188]

We used LND to explore and analyse the Lightning Network as described in chapter 4 because it is a well-documented, used and tested implementation of the Lightning Network.

3.2.2 Network Statistics

To give the reader an idea about the network, we will present some statistics first. As of 10^{th} November 2019, 4,821 active nodes were spanning 30,920 open channels across the Lightning Network. All channels together were funded with 811.169 BTC resulting in an average node capacity of 0.343 BTC and an average channel capacity of 0.0263 BTC.

As of 5^{th} November 2018, 1,751 active nodes were spanning 9,188 open channels across the network. All channels were funded with 107.943 BTC resulting in 0.0118 BTC per channel and 0.127 BTC per node [189].

Within this year, this is a node increase of about 175% and a channel increase of 237%. Total funds of all channels increased by 651%. Funds per channel increased by 123% and funds per node increased by 170%. Table 3.1 shows the comparison of these figures in 2018 and 2019.

Date	Nodes	Channels	Total Capacity	Node Capacity	Channel Capacity
2019-11-10	4,821	30,920	811,169 BTC	0.343 BTC	0.0263 BTC
2018-11-05	1,751	9,188	107.943 BTC	0.127 BTC	0.0118 BTC
Change	+175%	+237%	+651%	+170%	+123%

Table 3.1: Lightning Network statistics 2019 vs. 2018.

3.2.3 Commitment Transactions

As mentioned in section 3.1.3, commitment transactions are a fundamental building block in the concept of payment channels in the Lightning Network [11]. Commitment transactions guarantee that two interacting parties that do not trust each other per default, are acting honestly anyway. A commitment transaction can be seen as the balance sheet of the channel. Once a channel is opened, three transactions are created. The funding transaction which is send to the Bitcoin network right after all three transactions are created and signed, and two commitment transactions about the current balance of the channel which are not send to the network. Figure 3.12 shows the funding transaction (F) and two commitment transactions (C1a & C1b) as an example where Alice and Bob are creating a payment channel and funding it with 0.5 BTC each at the beginning. Transactions in purple can only be send by Alice whereas blue transactions only by Bob. Summarised, Bob can claim 0.5 BTC immediately and Alice can claim 0.5 BTC after 1,000 blocks if she sends C1a. Yet: If Bob sends C1b, Alice can claim 0.5 BTC immediately and Bob can claim 0.5 BTC after 1,000 blocks.

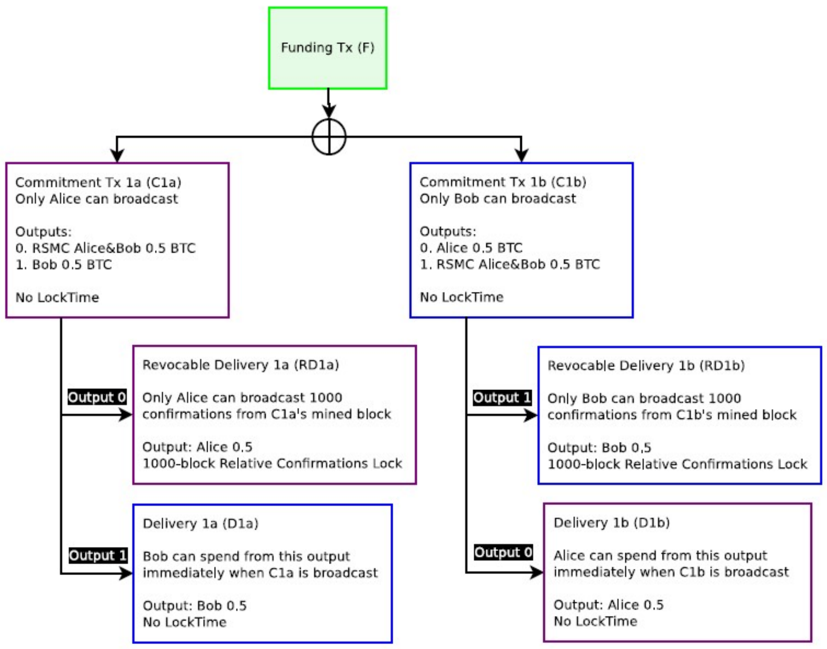

Funding Tx (F)
Commitment Tx 1a (C1a)
Only Alice can broadcast
Outputs:
0. RSMC Alice&Bob 0.5 BTC
1. Bob 0.5 BTC
No LockTime
Commitment Tx 1b (C1b)
Only Bob can broadcast
Outputs:
0. Alice 0.5 BTC
1. RSMC Alice&Bob 0.5 BTC
No LockTime
Output 0
Revocable Delivery 1a (RD1a)
Only Alice can broadcast 1000 confirmations from C1a's mined block
Output: Alice 0.5
1000-block Relative Confirmations Lock
Output 1
Revocable Delivery 1b (RD1b)
Only Bob can broadcast 1000 confirmations from C1b's mined block
Output: Bob 0,5
1000-block Relative Confirmations Lock
Output 1
Delivery 1a (D1a)
Bob can spend from this output immediately when C1a is broadcast
Output: Bob 0.5
No LockTime
Output 0
Delivery 1b (D1b)
Alice can spend from this output immediately when C1b is broadcast
Output: Alice 0.5
No LockTime

Figure 3.12: The fist commitment transaction pair after a channel opening between two participants [11].

If Alice intends to transfer 0.1 BTC to Bob, a new pair of commitment transactions (C2a & C2b) is created, signed and exchanged. Both revocable delivery transactions (RD1a & RD1b), belonging to the prior commitment transactions (C1a & C1b), are invalidated. Also, Alice and Bob both creates a breach remedy transaction (BR1a & BR1b), spending from the prior commitment transactions (C1a & C1b). They also exchange the private keys for this transaction to enable the counterparty to spend from this transaction. This procedure contains the commitment aspect that no party send an outdated commitment transaction. If a party nevertheless sends an outdated commitment transaction, the other party can claim its assigned funds and also all the remaining funds from the payment channel by spending from the respective breach remedy transaction as a penalty for the malicious behavior. E.g. Alice sends C1a to the network, Bob receives 0.5 BTC from output 1 (D1a) and also 0.5 BTC from output 0 (BR1a). Note that Bob has not received 0.1 BTC yet because only one transaction can spend from the funding transaction. In that example, C1a spends from F, in which Bob owns 0.5 BTC, and not C2a or C2b. Alice got penalized for bad acting. From that time on, both interacting parties are incentivised to act honestly. Figure 3.13 shows the discussed state of the different transactions.

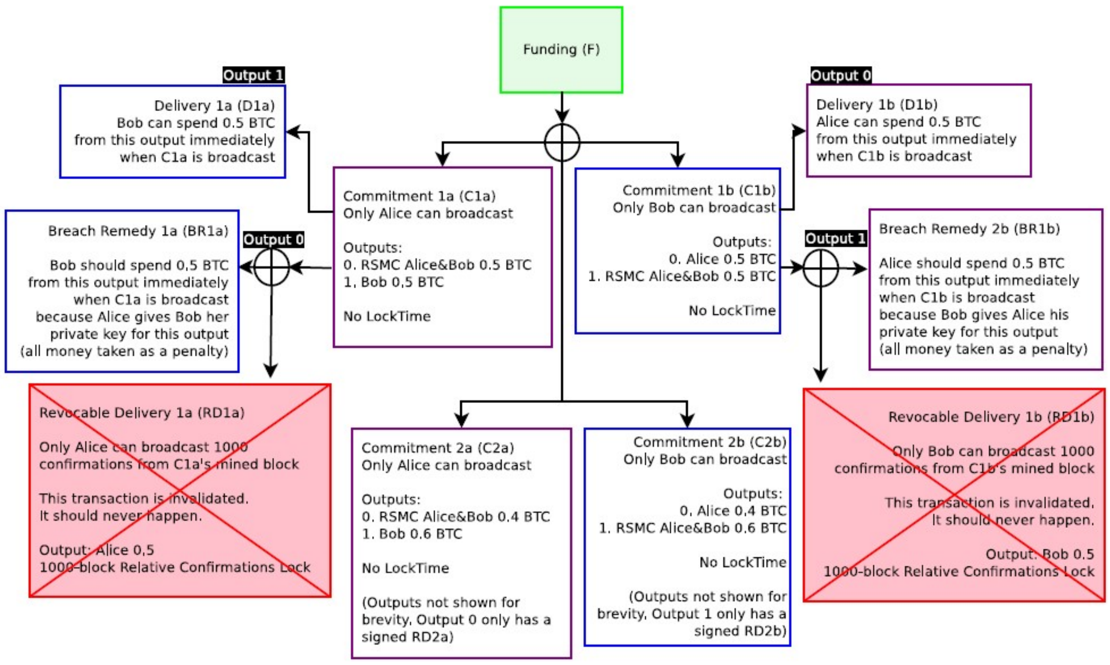

Funding (F)
Output 1
Delivery 1a (D1a)
Bob can spend 0.5 BTC
from this output immediately
when C1a is broadcast
Output 0
Delivery 1b (D1b)
Alice can spend 0.5 BTC
from this output immediately
when C1b is broadcast
Commitment 1a (C1a)
Only Alice can broadcast
Outputs:
0. RSMC Alice&Bob 0.5 BTC
1. Bob 0,5 BTC
No LockTime
Commitment 1b (C1b)
Only Bob can broadcast
Outputs:
0. Alice 0.5 BTC
1. RSMC Alice&Bob 0.5 BTC
No LockTime
Breach Remedy 1a (BR1a)
Bob should spend 0,5 BTC
from this output immediately
when C1a is broadcast
because Alice gives Bob her
private key for this output
(all money taken as a penalty)
Output 0
Output 1
Breach Remedy 2b (BR1b)
Alice should spend 0.5 BTC
from this output immediately
when C1b is broadcast
because Bob gives Alice his
private key for this output
(all money taken as a penalty)
Revocable Delivery 1a (RD1a)
Only Alice can broadcast 1000
confirmations from C1a's mined block
This transaction is invalidated.
It should never happen.
Output: Alice 0,5
1000-block Relative Confirmations Lock
Commitment 2a (C2a)
Only Alice can broadcast
Outputs:
0. RSMC Alice&Bob 0.4 BTC
1. Bob 0.6 BTC
No LockTime
(Outputs not shown for
brevity, Output 0 only has a
signed RD2a)
Commitment 2b (C2b)
Only Bob can broadcast
Outputs:
0. Alice 0.4 BTC
1. RSMC Alice&Bob 0.6 BTC
No LockTime
(Outputs not shown for
brevity, Output 1 only has a
signed RD2b)
Revocable Delivery 1b (RD1b)
Only Bob can broadcast 1000
confirmations from C1b's mined block
This transaction is invalidated.
It should never happen.
Output: Bob 0.5
1000-block Relative Confirmations Lock

Figure 3.13: The second commitment transaction pair after a channel update between two participants [11].

If Alice validly closes the payment channel, Bob receives 0.6 BTC immediately and Alice has to wait for 1,000 blocks to receive 0.4 BTC. If Alice and Bob agree on the closing of the channel, Alice can claim 0.4 BTC and Bob 0.6 BTC immediately. They agree on a settlement transaction which has no time locking output and so none of the two parties have to wait for 1,000 blocks, regardless of who intended to close the channel. Figure 3.14 shows the settlement transaction on cooperative channel closing.

In this section, some details on the Lightning Network, i.e. implementations, statistics

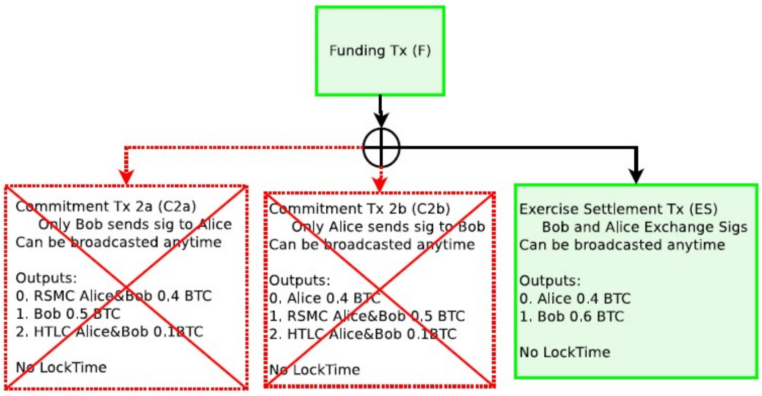

Funding Tx (F)
Commitment Tx 2a (C2a)
Only Bob sends sig to Alice
Can be broadcasted anytime
Outputs:
0. RSMC Alice&Bob 0.4 BTC
1. Bob 0.5 BTC
2. HTLC Alice&Bob 0.1BTC
No LockTime
Commitment Tx 2b (C2b)
Only Alice sends sig to Bob
Can be broadcasted anytime
Outputs:
0. Alice 0.4 BTC
1. RSMC Alice&Bob 0.5 BTC
2. HTLC Alice&Bob 0.1BTC
No LockTime
Exercise Settlement Tx (ES)
Bob and Alice Exchange Sigs
Can be broadcasted anytime
Outputs:
0. Alice 0.4 BTC
1. Bob 0.6 BTC
No LockTime

Figure 3.14: A settlement transaction after a channel update on a cooperative closing between two participants [11].

and the workflow of commitment transactions have been discussed. The next section will explain limitations and challenges of the Lightning Network and payment channel networks in general.

3.3 Limitations & Challenges

There are several limitations and challenges within the Lightning Network and the field of payment channel networks in general. In this section, we will discuss some of the major limitations and challenges, i.e. liquidity, routing, uptime, node centrality, data loss, privacy-utility trade-offs and deadlocks.

Liquidity A very important property of payment channel networks is liquidity. The more liquidity a payment channel network provides, the more efficient and the faster payments can be processed within the network. There are two types of liquidity: inbound and outbound liquidity [180]. Inbound liquidity refers to the amount of funds a node can receive from the counterpart node. On the other hand, outbound liquidity refers to the amount of funds a node can send to an counterpart node. Rebalancing discussed in section 3.5.4 and splicing discussed in section 3.5.5, are some mechanisms to provide and manage liquidity within the network.

Routing As in every network, efficient routing is a challenge also in payment channel networks. It is not easy to find the best and most efficient route for a payment in a system which changes over time. The amount of hops, fees along the path, channel balances along the path, etc. has to be taken into consideration when calculating a route from a source to a destination of a payment.

Uptime Another important challenge in payment channel networks is the fact that nodes have to be online to send, receive or route transactions. There are mechanisms to overcome some of the limitations like time locks or watching services discussed in section 3.5.6. Asynchronous payments, as discussed in section 3.5.8, reduce this limitation.

Centrality Beside these, also the centrality of the network is an important challenge. Node centrality is double-edged. On the one hand it can provide fast and efficient payments within a payment network, as discussed in section 2.4.6, but on the other hand it is a security and privacy vulnerability.

Data Loss Another challenge in payment channel networks is the durability of data, respectively the risk of data loss. Channels are stateful, therefore every participant has to know and maintain the latest state of a channel and all its data. If a participant loses all the data of a payment channel, the counterparty can close the channel with an invalid commitment transaction and the party who lost the data cannot revoke the invalid commitment transaction and receive all the remaining funds because he or she does not know the details of the invalidated commitment transaction anymore. Besides, if a node with partially data loss sends a wrongly believed latest commitment transaction, the counterparty can claim all the remaining funds from the payment channel. In October 2019 there was a case, where a Lightning Network user lost approximately 4 BTC because he mistakenly published an outdated commitment transaction [190]. His node was not in sync at the time he closed the channel. Therefore secure storage of payment channel data is essential [11].

Privacy-Utility Trade-off The next challenge we would like to mention is the privacy-utility trade-off. Tang et al. discuss in [191] the trade-off between privacy and utility at routing in payment channel networks. The more information of the network and its participants is publicly available, the more efficiently and effectively a payment routing can be calculated and selected, but the weaker is the privacy of the participating nodes. On the other hand, the less information of a network and its participants is publicly available, the stronger is the privacy of all participants of the network, but the more difficult and the more error-prone a calculation and selection of a payment routing is. The authors conclude that there is no perfect trade-off between these two properties. A payment channel network should either provide full privacy with all its drawbacks or full utility by sacrificing the privacy of the participants.

Deadlocks The last challenge discussed in this context are deadlocks in payment routes [192] within a payment channel network. A deadlock can occur if two concurrent multi-hop payments are processed and if they use the same nodes or partially the same nodes for routing the payment. Figure 3.15 is illustrating a deadlock caused by two concurrent multi-hop payments, routed partially through the same nodes. In this example, Alice wants to transfer Bitcoins to Gabriel and Bob wants to transfer Bitcoins to Edward at the same time. Alice's payment, i.e. the red payment route, is routed through Carol, Edward and Fabi to Gabriel. Bob's payment, i.e. the blue payment route, is routed through Fabi, Gabriel and Carol to Edward. In the first step, Alice & Carol and Bob & Fabi set up an HTLC between each other. Next, Carol & Edward and Fabi & Gabriel set up one between them. Third, Edward & Fabi and Gabriel & Carol set up an HTLC. Then the deadlock occurs because next Fabi & Gabriel and Carol & Edward want to set up an HTLC between them, but both do not have enough funds left in their channels for doing so. Both payments got stuck within the payment channel network. In [192] two solutions for avoiding deadlocks within a payment channel network were introduced: a payment-blocking and a payment-non-blocking solution. The payment blocking approach simply aborts both payments if a deadlock occurs and therefore no state is changed. The payment-non-blocking approach prioritizes a payment based on a global order and, in case of a deadlock, proceeds with the payment which has the higher priority and aborts the other one.

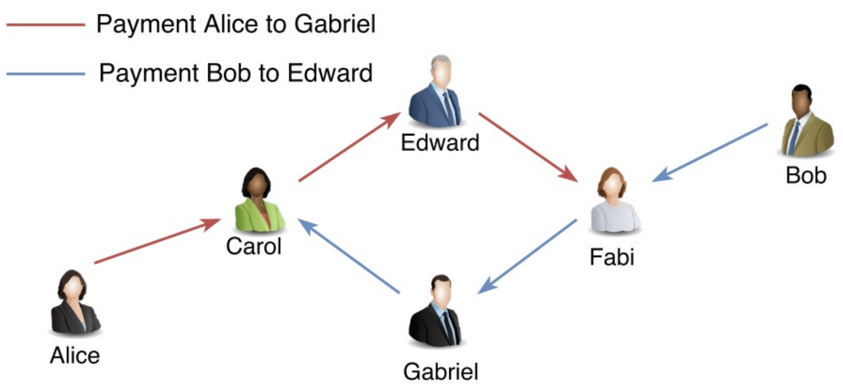


Figure 3.15: A deadlock in a payment channel network caused by two concurrent multi-hop payments. The red payment route got stuck at Fabi and the blue payment route got stuck at Carol because they have too few funds for routing another payment [192].

In this section, the focus has been on some of the major limitations and challenges in payment channel networks like the Lightning Network. Next, privacy and security attacks will be treated.

3.4 Privacy & Security

In this section, we will discuss some of the most important attacks concerning security and privacy within payment channel networks and the Lightning Network. Furthermore, wormhole attacks, mechanisms to break the relationship anonymity and observe channel balances, as well as balance availability and denial-of-service attacks will be explained.

3.4.1 Wormhole Attack

A wormhole attack can be executed in a multi-hop payment channel with more than three hops between the origin and the destination of a payment [193]. By performing this attack, honest intermediates within a payment flow are cut off the routing process by malicioius nodes. This is why the wormhole attack targets the fees of the intermediates. Figure 3.16 shows the concept of the wormhole attack. In this example presented in [193], Alice wants to send a multi-hop payment to Edward. She wants to transfer 10 BTC, but she adds 1 BTC for each intermediate: Bob, Carol and Dave. The workflow illustrated in green is the honest one, whereas the red one is the malicious one. The black workflow is the same for both workflows. First, Edward creates a secret and its SHA-256 hash, which he provides to Alice. Alice locks 13 BTC, the hash of the secret and a block height, until the secret has to be provided to release the locked amount in an HTLC to Bob. Bob does the same to Carol, but he only locks 12 BTC and a block height lower than the locked block height in the HTLC between Alice and him. Carol in return does the same with Dave, but again with a reduced amount and block height. Next, Dave does the same with Edward, also with a reduced amount of 10 BTC and a reduced block height, until the HTLC releases the locked funds under the condition that no secret is provided which matches the stored hash. These steps are equal in both, honest and malicious behavior.

In the honest behavior Edward provides the secret to the HTLC with Dave and receives 10 BTC. Dave does the same with the HTLC with Carol and receives 11 BTC. Carol does the same with the HTLC with Bob and receives 12 BTC. Bob does the same with the HTLC with Alice and receives 13 BTC. In this case Edward received 10 BTC and each intermediate received 1 BTC for routing the payment through their nodes.

In the malicious behavior, Dave and Bob are either the same node or are collaborating to steal the fees designated for Carol. When Edward provides the secret to the HTLC with Dave and claims 10 BTC from it, Dave forwards a 'payment failed' notification to Carol, but passes the secret to Bob, who can claim 13 BTC from the HTLC with Alice for providing the secret. Carol receives 11 BTC back from the HTLC with Dave after 1,000 blocks. Bob also gets 12 BTC back from the HTLC with Carol after 1,000 blocks, because she didn't provide the secret. In that case, Bob and Edward got the fees intended for Carol. The more intermediates are involved in a payment flow and the closer the two malicious acting nodes are located to the origin and the destination of a payment, the higher the fees they can steal.

3.4.2 Breaking Relationship Anonymity

Interacting participants in a payment channel network cannot be identified by intermediates per default. In a special scenario, like a wormhole attack, cooperative intermediates which are located right next to the sender and right before the destination of a payment, can identify interacting participants based on the hash of the secret which is locked in the HTLCs along the payment route [194]. Figure 3.17 shows the concept on how to break the relationship anonymity. If E1 and E2 are cooperating, they can identify that

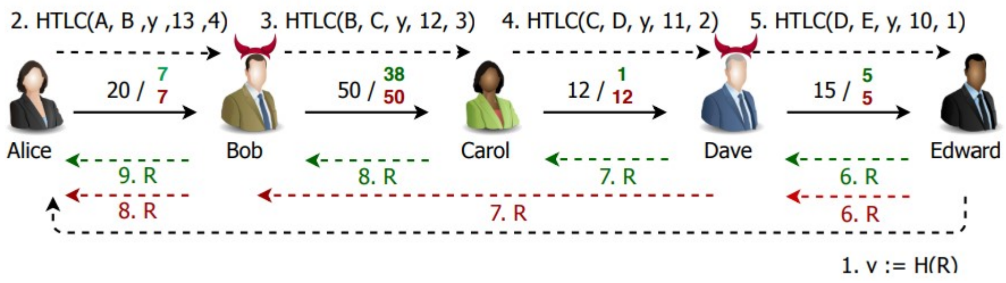


Figure 3.16: The setup for performing a wormhole attack in a multi-hop payment between Alice and Edward. Bob and Dave cooperate and in this way steal funds intended for Carol [193].

A sent a payment to C and that A' sent a payment to C'. Other intermediates, i.e. B, cannot observe that. For other intermediates it looks the same like A sent a payment to C' and A' sent a payment to C.

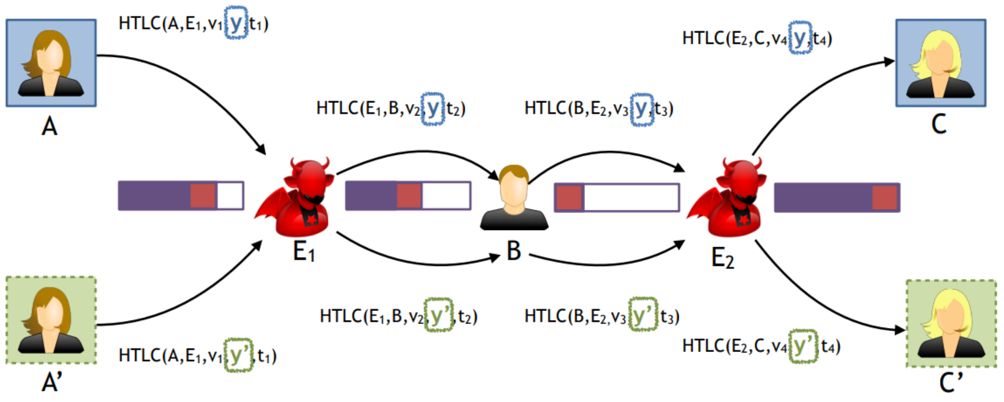


Figure 3.17: The setup for breaking relationship anonymity by cooperative intermediates. E1 and E2 are next and before the sender and receiver and so they can reveal which party sends how much tho whom by looking at the respective secrets of the HTLCs [194].

3.4.3 Channel Balance Discovery

Herrera-Joancomartí et al. discuss in [195] a way to discover balances of payment channels. Figure 3.18 shows the network setup for the attack. E.g. Alice, i.e. A, and Bob, i.e. B, have a payment channel, i.e. AB, with a capacity C_{AB}. Mallory, i.e. M, creates a payment channel with Alice, i.e. MA, and a capacity C_{MA}. Mallory is now sending payments to Bob. By gradually increasing the amount, at some point the payment will fail because C_{AB} is too less. From then on, Mallory knows the balance of channel AB. The attack requires C_{MA} to be higher than C_{AB}. In order not to send payments to Bob each time, Mallory creates fake payments which fails in transition from Alice to Bob, but not from Mallory to Alice. Depending on the balance height where the failure occurs, Mallory can identify the balance in C_{AB}.

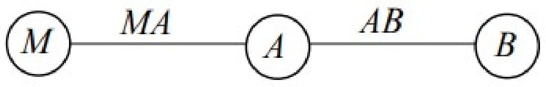


Figure 3.18: The node setup for performing a balance discovery attack. M has a channel with A, funded with balance MA and A has a channel with B, funded with balance AB [195].

3.4.4 Balance Availability Attack

Pérez-Solà et al. introduce LockDown in [196], an attack against balance availability of payment channels. The attack can be used in multi-hop payments because during the payment path setup, every node locks the required amount of funds into an HTLC with the subsequent node until the path reaches the target of the payment. In honest behaviour, the target will release the secret immediately to the HTLC with the node before in a payment path to receive the funds once the path is set up. This node unlocks the HTLC with the node before to receive the funds from its predecessor. This process is continuing until it reaches the source node of a payment. If a malicious target node does not reveal the secret to the HTLC with the node before in the payment path, all funds are locked until a timeout expires. During this time, locked funds cannot be used for routing other payments through the respective nodes and channels within the network until all of the HTLCs expire or the secret is revealed and funds are released.

3.4.5 Denial-of-service Attack

Like other networks, the Lightning Network is also vulnerable to denial-of-service (DoS) attacks. An attack in March 2018 where 20% of all nodes were down due to a distributed denial-of-service (dDoS) attack showed this [197]. As discussed by Seres et al. in [198], a DoS attack can be very harmful if it is well coordinated. By taking down for example the node with the highest degree, the Lightning Network fragmented into 37 connected components. This could avoid that payments are processed in the network between two nodes because no longer a path between them exists. Also targeted DoS attacks against certain nodes can prevent them from revoking invalid commitment transactions and in this way they lose funds.

In this section, we introduced the most important privacy and security attacks concerning payment channel networks and the Lightning Network. Next, we will discuss some major enhancement approaches to harden those attacks and to weaken other challenges and limitations.

3.5 Enhancement Approaches

In this section, we will discuss some enhancement approaches for payment channel networks, as well as anonymous multi-hop-locks, atomic multi-path payments, split payments, payment channel rebalancing, splicing, watching services, trampoline payments, asynchronous payments, multiplexed payment channels and deliberately selecting longer payment routes.

3.5.1 Anonymous Multi-hop-locks

Anonymous multi-hop-locks is a concept presented by Malavolta et al. in [193]. As discussed in sections 3.4 and 3.1.4, a payment route can be identified by the locked hash of the secret which has to be provided to release the locked funds from the HTLC. This is a privacy issue. By using anonymous multi-hop-locks each two interacting parties along a payment route create and commit the hash of their own secret for locking funds in HTLCs. This is realised during the setup phase of a payment route which makes it also more complex. On the other hand, it helps to prevent the privacy within a payment channel network and also removes the possibility of wormhole attacks.

3.5.2 Atomic Multi-Path Payments

Atomic multi-path payments over the Lightning Network were proposed by O. Osuntokun in [199]. The main idea is to use different channels to process a payment to a target. That is why the payment has to be atomic, meaning either all single payments succeed or none of them. This eliminates the limitation that one path has to exist which can provide enough liquidity for a payment. It could also reduce fees if routing nodes charge different fee rates. Multi-path payments also increase the privacy by splitting one payment into several ones. However, it is a big challenge to handle multi-path atomicity reliably.

3.5.3 Split Payments

D. Piatkivskyi and M. Nowostawski proposed in [200] a related idea for splitting a payment into smaller parts, into atomic multi-path payments discussed above. The main difference compared to atomic multi-path payments is that sub-payments are also timely spread. This increases the probability of a successful payment. Split payments achieve an improved total liquidity of the payment network, reduce the amount of funds needed to be locked in a payment channel and improve the privacy of the payment channel network consequently. Handling multi-payments is also a big challenge with split payments.

3.5.4 Payment Channel Rebalancing

If payments are routed in the same direction more often than in the other, channels within a payment channel network get unilaterally. At some point in time a channel cannot route a payment anymore. To avoid unbalanced channels, a node increases the balance of a channel by the expense of decreasing the balance of other channels. This requires interactions with other nodes which causes fees [201]. Channel rebalancing is important to keep the payment channel network operable.

3.5.5 Splicing

Another way of adapting channel balances is splicing. Splicing-in is the process of adding funds to a payment channel whereas splicing-out is the process of removing funds from the channel. Splicing closes the channel and re-opens it with one single blockchain transaction. This also causes fees [202]. Splicing is a necessary mechanism to keep the network liquid.

3.5.6 Watching Services

If Alice closes a channel while Bob is offline, Alice has to wait for 1,000 blocks to receive her funds. Bob receives his funds immediately and has the chance to claim also the funds for Alice if Alice sent an invalidated commitment transaction as a penalty. To do so, she has to come online within the 1,000 blocks' confirmation. If she does not, Bob receives his funds, even if he acted maliciously. Watching services are third parties acting in behave of another party in such a case. If Bob is offline, the watching service overtakes the responsibility of claiming the funds for Alice from the payment channel, in the case Alice sent an outdated commitment transaction. Also if Bob loses all the data from a payment channel, the watching service can protect a loss of funds. For doing so, the watching service gets some fees from Bob [203].

3.5.7 Trampoline Payments

O. Wagih proposed a different way of payment route calculation in [175]. In current implementations, the source of a payment has to calculate the route through the network to the destination. To do so, the node has to maintain a state of the network of all nodes and public channels. If the Lightning Network grows to much higher levels, calculating the optimal route could become a very hard problem for nodes with less computational power. For example node A wants to send a payment to node F in a channel network graph which looks like $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F$. Maybe node A only knows node B and node C, but node C has a full view of the complete graph and enough computational power to calculate a route to node F. Node A could send the payment to node C and ask if node C could route the payment to node F. In that case node C is acting as a trampoline node. Node C could charge fees for keeping the complete state of the graph and for calculating the route to node F.

3.5.8 Asynchronous Payments

In general, two interacting nodes have to be online at the same time to process a payment through a payment channel. Lightning Rod Protocol [204, 205] overcomes this limitation by using an intermediate node which is always online and forwards the payment once the target of a payment is offline and comes online again. The main difference of the Lightning Rod Protocol compared to standard multi-hop payment setup is, the construction of HTLCs along the payment route. Based on the following example, in which Alice wants to send a payment to Bob via an intermediate node, we will explain the differences.

1. Alice sends a secret to Bob and can go offline.
2. Bob also creates a secret and sends a hash of both secrets to the intermediate node. Bob can go offline now as well.
3. The intermediate node sends Alice the hash of Bob's secret along with a HODL invoice when she comes online again. A HODL invoice allows the receiver to await or abort the payment while the sender remains committed. Alice can validate the origin of the payment request.
4. Once the request has been validated, Alice starts to set up an HTLC with the intermediate node which sets up an HTLC with Bob when he comes online again. From that point on the process is the same as in multi-hop payments.

Non-atomicity of the multi-hop payment allows the payment to be processed in an asynchronous way between two nodes via an intermediate.

3.5.9 Multiplexed Payment Channels

In default payment channels, only two participants can exchange values even if third parties are used for multi-hop payment routing. Pan et al. introduce Gnocchi in [206], a multiplexed payment channel scheme. This scheme offers a novel method to set up off-chain payment channel systems, to allow multi-party payments in one channel. Several participants can request the opening of a multiplexed payment channel with the cooperation of a transaction supervisor. Anyway, the supervisor is not able to modify the balance of any channel participant without its authentication. The multiplexed payment channel consists of a novel fixed-member channel state consensus mechanism and only the supervisor is required to be online. Beside these improvements, Gnocchi reduces routing costs and payment fees for multi-hop payments compared to existing payment channel networks such as the Lightning Network.

3.5.10 Deliberately Selecting Longer Payment Routes

Beres et al. describe in [207] that many of the payments within the Lightning Network are either single hop payments or multi-hop payments with only one intermediary in between. Knowing that most of the routed payments contain only one hop, an intermediary can easily de-anonymize sender and receiver of a payment. One way to overcome this fact is to deliberately choose longer payment routes by the sender in the process of source routing discussed in section 3.1.5. This results in slightly higher fees and a lower payment speed, but in better privacy for the sender and receiver. There is a certain trade-off between fees, speed and privacy.

In this section, we have discussed some of the most promising enhancement approaches for payment channel networks concerning security and privacy attacks and other limitations. Next, we will concentrate on analysis, which is done on the Lightning Network to gain a deeper insight into it.

3.6 Analysis

In this section, we will introduce some of the existing approaches to analyse payment channel networks and discuss in detail two important works in this field from which we also reproduce the results presented in chapter 5.

3.6.1 Evaluating Methods for the Identification of Off-Chain Transactions in the Lightning Network

First, we will discuss the paper of M. Nowostawski and J. Ton [208], an approach to analyse payment channels within the Lightning Network. Broadly speaking, they look for possible closing transactions on the blockchain and for possible corresponding funding transactions based on a certain transaction scheme. By combining these potential channel openings and closings with data collected from the Lightning Network, false positives can be reduced significantly. We will discuss the methodology in detail, on which we also built up parts of our methodology which we will present in chapter 4.

Methodology The first step is to find potential closing transactions on the Bitcoin blockchain. A closing transaction consits of one single input which is a 2-of-2 multi-signature redeem script. Beside Lightning Network channel closing transactions, there are also other transactions of this type. Therefore this mechanism produces also false positives. The input of the closing transaction references the possible funding transaction. Figure 3.19 shows the concept of these two corresponding transactions.

To identify possible closing transactions, one has to have a look into all transaction inputs, from transactions with only one single input. This single input has to be a 2-of-2 multi-signature script. Once a possible closing transaction is identified, the corresponding funding transaction can be found by the referenced transaction hash in the transaction input. These two identified transactions are a possible funding and closing transaction pair of a Lightning Network channel. The interacting addresses and amounts, which were initially locked in the payment channel, can be observed beside several other information. As already mentioned, this mechanism also provides false positives which have to be filtered and minimized with additional approaches.

As discussed in 3.1.3, this is not the only way a payment channel can be closed. One also has to have a look at transactions which are created on an uncooperative payment channel closing. Figure 3.20 shows the respective transactions caused by an uncooperative

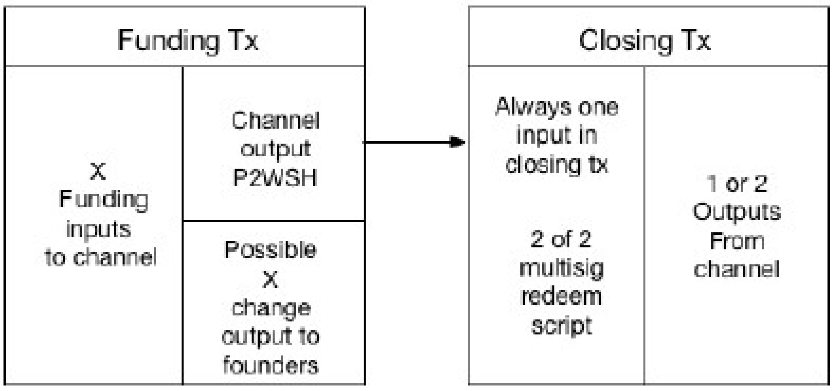


Figure 3.19: A channel funding and closing transaction pair. The input of the closing transaction references the respective output of the funding transaction [208].

payment channel closing. As mentioned in sections 3.1.3 and 3.2, the party closing the payment channel in an uncooperative way has to wait for a certain number of blocks to enable the other party to revoke the transaction if it contains a wrong state. Therefore one transaction output is timelocked.

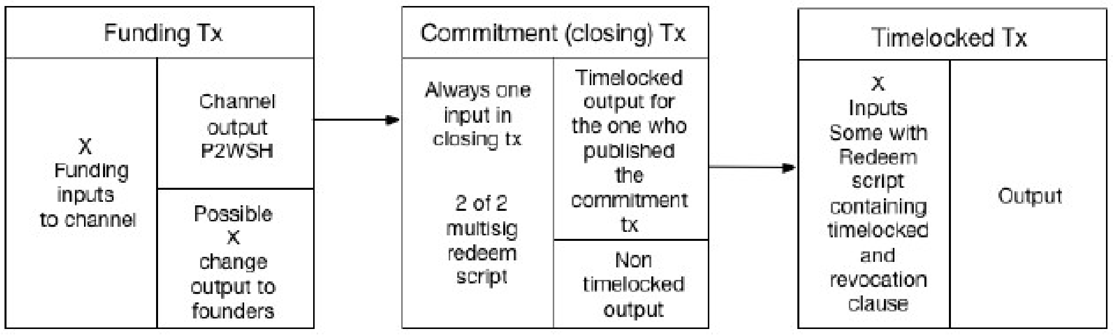


Figure 3.20: A funding, closing and timelocked transaction trio. The input of the closing transaction references the respective output of the funding transaction. The timelocked transaction input references the output for the party closing the channel [208].

To identify timelocked transactions, one has to have a look into all transaction inputs, of transactions which only has one input. This single input has to be a redeem script with a timelocked and a revocation clause. Once a timelocked transaction is identified, the corresponding closing transaction can be found by the referenced transaction hash of the transaction input. In turn, the corresponding funding transaction can be found by the referenced transaction hash of the transaction input from the closing transaction. Transactions identified by this scheme are likely to be Lightning Network related transactions, yet not with certainty.

A third way a channel can be closed is that if an HTLC cannot be resolved off-chain, the channel will be closed and the HTLC will be handled on chain. As this is a state which does not occur very often compared the prior two types of channel closings, Nowostawski and Ton did not focus on this in their analysis and nor do we.

As a next step they compared the identified possible Lightning Network related transactions with data collected from the Lightning Network. For this they modified an LND client to gain further insights into the network. By doing this, Nowostawski and Ton could reduce false positives. They disabled the feature that inactive nodes and channels are removed from the maintained graph of a node. This allowed the authors to have a more complete view of the network graph.

We have discussed the methodology in this section and most important results presented by Nowostawski and Ton will be summarised following.

Results Nowostawski and Ton analysed the Lightning Network concerning the size, reused keys, centrality measures and other key metrics which we will not mention here because we will present another paper in this field in the next section. Also, the authors run two data collection intervals, both on the Bitcoin mainnet and the testnet. The second snapshot on the mainnet consits of 1,151 blocks and they identified 667 potential Lightning Network channels based on the discussed identification scheme. During the second data collection on the mainnet they identified that more than 75% of 2-of-2 multi-signature transactions are most likely Lightning Network channels.

In this section, we summarised a paper which introduced a mechanism to identify potential Lightning Network channels on the Bitcoin blockchain. Next, we will discuss another interesting paper performing different analysis approaches on the Lightning Network, on which we also partially reproduced the presented results.

3.6.2 Topological Analysis of Bitcoin's Lightning Network

In [198] Seres et al. discuss the topological analysis of the Lightning Network. They first exported the graph from an LND client and further analysed different network metrics based on it. Below we will discuss the methodology and the results presented by the authors.

Methodology Seres et al. parsed a graph representation consisting of vertices (nodes) and edges (channels) of the Lightning Network, created by a LND client and further processed different metric calculations on it like network properties, degree distribution, betweenness and closeness centrality and clustering coefficients. In the following section a short overview of the most interesting results of their work will be given.

Results As of January 2019 Seres et al. identified 2,344 nodes spawning 16,617 payment channels, which results in an average degree of about 7.09 channels per node. The average shortest path between two nodes within the network is about 2.81 nodes. Most of the nodes have only a few while other nodes have many channels. The Lightning Network consists of a small central clique and loosely connected periphery. All of the presented network metrics indicate that the Lightning Network is currently not robust against targeted attacks. Additionally, Seres et al. show that by removing the node with the highest degree, the Lightning Network gets fragmented into 37 connected components. Removing the biggest 30 nodes resulted in 424 components most of which are isolated nodes. They summarised that the Lightning Network is indeed robust against random failures, but not against targeted attacks due to the current structure. The authors conclude that the developers of the Lightning Network also have to focus on building a resilient network topology.

In this section, we introduced two papers in the field of Lightning Network analysis which are important for this thesis. Next, we would like to give the reader some ideas about critical perspectives on the Lightning Network.

3.7 Critical Perspectives

The Lightning Network offers big advantages, but on the other hand it also struggles with major limitations discussed in this chapter. In this section we will mention some critical perspectives on this new technology.

Beres et al. describe and argue a sceptical point of view compared to most of the other opinions concerning the impact and adoption of the Lightning Network in [207]. The authors examine the Lightning Network under economical and privacy aspects. Beres et al. implemented a traffic simulator to empirically analyse transaction fees and privacy provisions within the Lightning Network. At the time of the release of the paper in November 2019, they summarised that it is not economically rational to operate a routing node within the network. Either the traffic or the fees have to increase to make it feasible to participate in the network. Second, they conclude that statistical evidence on payment sender and receiver can be inferred because in most payments only one intermediary is in between both interacting parties. To overcome this fact, Beres et al. introduce a method to inject additional hops to obfuscate payment details. This method was already discussed in section 3.5.

Other critical views are targeting the usability of the current Lightning Network and its implementations in [209] and in [210]. The Lightning Network is a very new technology and there is a lot of development going on in this area. In the future, usability and other technical limitations have to be eliminated or reduced for the Lightning Network to become a widespread scaling solution for the Bitcoin blockchain.

As the reader can see, not everybody has a positive attitude towards the Lightning Network in the research field. Following we will summarise this chapter before going on to the next chapter.

3.8 Summary

In this chapter, we confronted the reader with the Lightning Network and payment channel networks in common. We started with the concepts and details of payment channel networks such as channel types, HTLCs, single- and multi-hop payment channels and routing in payment channel networks. Then we moved on to specifics of the Lightning Network i.e. implementations, network statistics and commitment transactions. Next, we discussed limitations and challenges as well as security and privacy attacks in payment channel networks in general, respectively the Lightning Network in detail. Subsequently, we presented the most promising enhancement approaches for many of the limitations and challenges discussed. After that, we presented relevant papers in the field of Lightning Network analysis and also mentioned critical perspectives on the Lightning Network. In the next chapter, we will introduce our methodology, on how we linked and mapped data from the Lightning Network with blockchain data and data from a cryptoasset analytics tool before presenting our results in chapter 5.

CHAPTER 4

Methodology

In this chapter we will explain how we gathered, linked and enriched data from the Bitcoin blockchain and the Lightning Network. First, we collected data on both layers. Next, we analysed them separately to gain a deeper insight into both networks. Subsequently, we linked both datasets to reproduce the results presented in [208] and to enhance the importance of the data. Next, we enriched the linked data with information from a cryptoasset analytics tool. Finally, we mapped data from a cryptoasset analytics tool to nodes from the Lightning Network based on a heuristic.

4.1 Data Collection & Extraction

We collected and extracted data from both networks, the Bitcoin blockchain (on-chain) and the Lightning Network (off-chain). For data extraction from the Bitcoin blockchain, BlockSci [211, 212], a high-performance tool for blockchain data extraction was used. BlockSci was available to us on a server within the network of the Austrian Institute of Technology. For the extraction of data from the Lightning Network, we used a locally installed and operating LND client. Bellow the data gathering process on both networks is described in detail.

4.1.1 Off-chain

We operated an LND node to collect data from the Lightning Network which offers the possibility to export the network graph maintained by the node via the command 'describegraph'. To regularly extract the graph, we created a cronjob which extracted the graph on an hourly basis. This enabled us to analyse channels and nodes and their evolution over time. Besides, we modified the LND client to export further data like interacting Bitcoin public keys on channel creation. While taking snapshots, we also modified the client not to remove inactive channels from the network graph maintained by the node. Preventing to remove inactive channels was also introduced in [208]. Removing inactive yet not closed channels would limit the set of compoundable data. Next, we will introduce the reader into the dataset we collected.

Dataset & Structure We exported snapshots from the Lightning Network on an hourly basis. We started the cronjob to do so on 28th November 2019 at 09:00:01 p.m. (CET) and ended it on 30th January 2020 at 08:00:01 a.m. (CET). In this period we collected 1,500 snapshots of channels and nodes from the Lightning Network. To operate on the data more easily, we stored them in a locally operating SQLite3 database [213]. A snapshot of the Lightning Network graph consists of nodes and channels between them. Listing 4.1 shows an example of a node and listing 4.2 an example of a channel between two nodes captured and exported by the LND client with the mentioned built-in command.

Listing 4.1: An example of the details of a node from the Lightning Network.

```
{
   "last_update":1567114656,
   "pub_key":"02022a6bcbd75baa2efb1c5f7badcc90337fc9a2f5b7730f0d35a69e560c79a99f",
   "alias":"lightning-hostel",
   "addresses":[
      {
         "network":"tcp",
         "addr":"71.58.99.157:9735"
      }
   ],
   "color":"#627277"
}
```

Listing 4.2: An example of the details of a channel from the Lightning Network.

```
{
   "channel_id":"595290988591185920",
   "chan_point":"8be38d557a01c3b73842335d02b29fe35c5d57ce27ca3a726495bec437dfe47e:0",
   "last_update":1576992565,
   "node1_pub":"031678745383bd273b4c3dbefc8ffbf4847d85c2f62d3407c0c980430b3257c403",
   "node2_pub":"031e47777e35627e1018bc886ee24f81633a54527cfce24df0183a8fee7b3a24ef",
   "capacity":"5000000",
   "node1_policy":{
      "time_lock_delta":144,
      "min_htlc":"1000",
      "fee_base_msat":"1000",
      "fee_rate_milli_msat":"1",
      "disabled":false,
      "max_htlc_msat":"5000000000",
      "last_update":1576914051
   },
   "node2_policy":{
      "time_lock_delta":144,
      "min_htlc":"1000",
      "fee_base_msat":"1000",
      "fee_rate_milli_msat":"1",
      "disabled":false,
      "max_htlc_msat":"5000000000",
      "last_update":1576992565
   },
   "btc1_pub":"02cb44fc859ca8ab83c0616e6ea3c430b5abba0de6b3ca1e594690dbbeafe72ea2",
   "btc2_pub":"02828a140346e865023df5e3150f3c077a27be54d5d486176a5bc6b8c8e92f6dc5"
}
```

4.1.2 On-chain

We used BlockSci to extract relevant data from the Bitcoin blockchain and followed the approach discussed by M. Nowostawski and J. Ton in [208] to identify possible Lightning Network channel transactions on the blockchain, as introduced in section 3.6. In detail, we traversed the blockchain block by block and transaction by transaction from the latest block towards the genesis block and analysed every transaction for the specific transaction scheme within the respective block range. We extracted the possible Lightning Network channel transactions as well as possible cooperative and possible uncooperative Lightning Network channel transactions separately. We also stored the data in the same SQLite3 database for easier processing it later on. Next, we will explain the dataset we collected.

Dataset & Structure To validate how many of the identified potential Lightning Network transactions are real Lightning Network channels, we had to extract the possible Lightning Network channel transactions from the Bitcoin blockchain in the same period, as we also collected data from the Lightning Network. Therefore we traversed the blockchain from block height 615,178 to block height 605,798 and analysed each transaction in that block range for the respective transaction scheme. Block 605,798 was mined on 28th November 2019 at 09:05:44 p.m. (CET) and block 615,178 was mined on 30th January 2020 at 08:00:50 a.m. (CET). In total, 19,094,656 transactions happened in these 9,381 blocks. We stored the transactions which matched the respective transaction scheme in a way to be able to link them to the collected off-chain data. Listing 4.3 shows an example of an on-chain extracted potential Lightning Network channel.

Listing 4.3: An example of a potential Lightning Network channel transaction extracted from the Bitcoin blockchain.

```json
{
   "funding_tx_id":"4e811782fc94c89cad238d01cb5acbe24a142a2a9c544d5f28ac1c2bdd39ecd6",
   "closing_tx_id":"2020ece61ad7fb5470c356edfee2352b77185c19de0486e22b4f491bceb68d74",
   "block_height_funding_tx":566101,
   "block_height_closing_tx":610379,
   "btc_key_1":"027a25b1864604689af4cd0fbcde8133884f861ceab98245aae07a9510cc11d509",
   "btc_key_2":"02e068b72c0b9fbc2df2c18fefe0be3c78c266bb8404fecbedbe8a7d99d475b095",
   "btc_addr_1":"18TqnEJyAyL9vBuX5yB8de1saSRMjy5xrf",
   "btc_addr_2":"13wD8rJaKHmwgh6ib7G6kgRr95d17aAj4J",
   "output_values":[
      112470,
      127155
   ]
}
```

4.2 Data Linkage

Once we had extracted and stored both datasets, we could link them based on the interacting Bitcoin public keys, the channel balances and the channel points. Figure 4.1 shows how data between the Lightning Network and the Bitcoin blockchain can be linked. The first part of the channel point references the funding transaction and the second part indicates the transaction output index of this transaction. The channel capacity can be linked to the value of the transaction output with the certain index. If the channel is already closed, this transaction output can be linked to the transaction input of the funding transaction, and so can the value and the address. In this way, also the closing transaction can be identified. Also, the interacting Bitcoin keys of the Lightning Network channel can be identified in the script of the transaction input from the closing transaction.

4.3 Data Enrichment

In this section, we will explain the last processing step we performed on the already linked on- and off-chain data. To further extend the importance of the linked data, we enriched them by data from a cryptoasset analytics tool. GraphSense [214], a cryptoasset analytics tool implemented by the Austrian Institute of Technology, uses the common-input-ownership heuristic to cluster addresses. This heuristic is based on the assumption that all inputs of a transaction belong to the same user [215]. By performing this heuristic on all transactions from the blockchain, GraphSense can create entities based on these transactions. An entity consits of clustered addresses and all the information based on these addresses. Entities evolve based on user behaviour and represent a more detailed view on real users than single addresses do. To enrich our data by entities, we loaded every input address of each channel funding transaction and if the channel was already

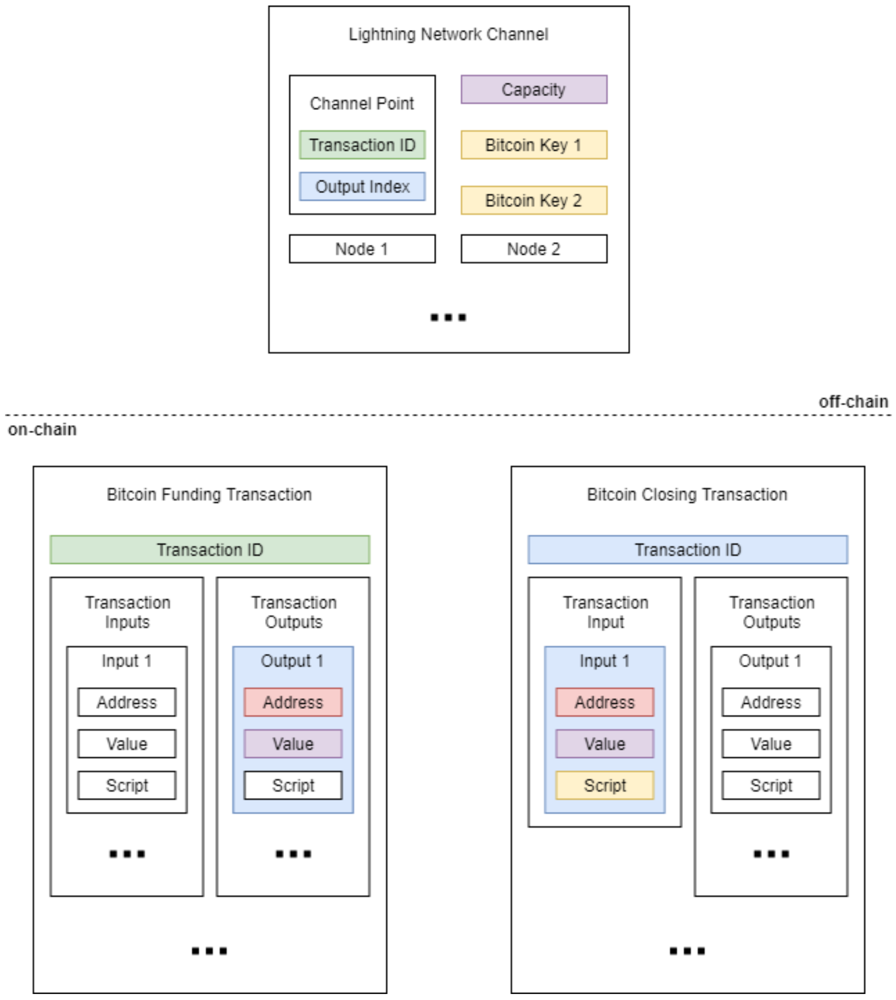


Figure 4.1: The relationship between details from the Lightning Network and extracted information from the Bitcoin blockchain. Colored data is linked across both networks.

closed, every output address of the channel closing transaction. By this, we were able to link all emerged channels over the captured period to entities from GraphSense. Figure 4.2 shows how on-chain transactions from a Lightning Network channel can be linked to entities from GraphSense.

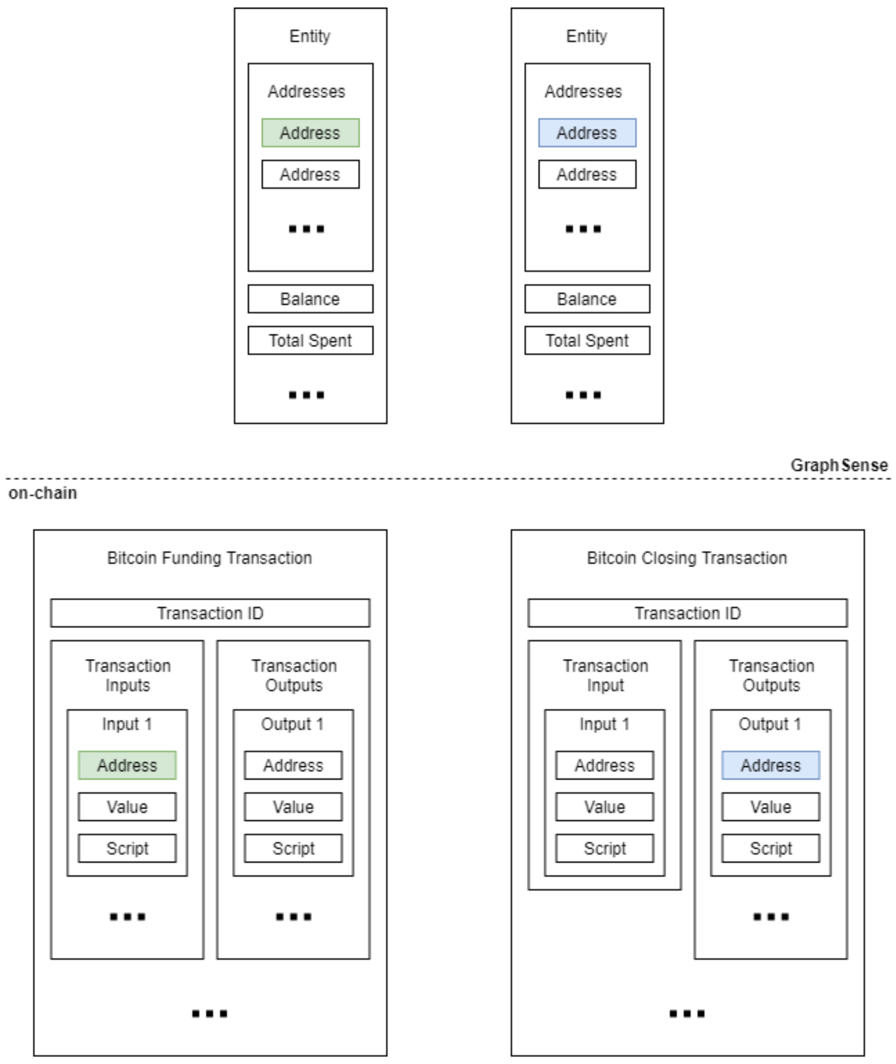


Figure 4.2: The relationship between on-chain Lightning Network transactions and entities from GraphSense. Colored data is linked across both layers.

4.4 Entity & Node Mapping

With the information of related channels and entities, we could furthermore map the GraphSense entities to the Lightning Network nodes based on a heuristic. To do so, we followed two sequential steps. The first step is to identify the node which occur most often among all to an entity related channels. The second step is to iteratively cross-check and improve the existing unique mapping.

Let us consider the example shown in table 4.1 to explain the first step of our heuristic.

In this example, one entity is linked to two channels, each of which consists of two nodes. By counting the node occurrences in all channels, we can heuristically identify the node, which is related to the entity. Table 4.2 shows the counts of the occurred nodes within all channels based on this example. We assume that the entity controls the node with the highest occurrence instead of all the other nodes.

Entity	Channels
Entity 1	[(Channel 1, Node 1, Node 2, ...),
	(Channel 2, Node 1, Node 3, ...)]

Table 4.1: An entity mapped to two channels, each of which consists of two nodes.

Entity	Node	Node Count / Channels
Entity 1	Node 1	2/2
	Node 2	1/2
	Node 3	1/2

Table 4.2: Node count in channels compared to all related channels to an entity.

After applying this first step to all entities and their linked channels, the result is a set of entities linked to one node or in some cases to several nodes if a mapping is not distinct. For example, this can happen if an entity is only related to one single channel. Then both nodes of this channel occur equally often. To extend the existing unique mapping furthermore, we iteratively cross-check the actual mapping.

Let us consider the example shown in table 4.3 to explain the second step of our heuristic. There is one entity, which is uniquely mapped to a node and two other entities that are not unique mapped to a node. By eliminating all the unique mapped nodes from all the other mappings, we can extend the existing unique mappings. In this example, after performing the first improvement iteration, a second node can be uniquely mapped to an entity. Table 4.4 shows the existing mappings after the first iterative improvement. By performing this iteration a second time, we can uniquely map a third entity. Table 4.5 shows the result after the second iterative improvement.

By performing the first step of our heuristic one time and the second step as often as improvements can be achieved, we can heuristically map entities to nodes. Besides we assume that the entity controls the node with the highest occurrence instead of all the other nodes.

In the above sections, we introduced the reader into our methodology in detail. Before we will present our results in the next chapter, we will following summarise the major steps of our methodology.

Entity	Nodes
Entity 1	[Node 1]
Entity 2	[Node 1, Node 2]
Entity 3	[Node 1, Node 2, Node 3]

Table 4.3: Entity-node-mappings before first improvement iteration.

Entity	Nodes
Entity 1	[Node 1]
Entity 2	[Node 2]
Entity 3	[Node 2, Node 3]

Table 4.4: Entity-node-mappings after first improvement iteration. By removing uniquely mapped nodes from other mappings, the existing unique mappings can be extended.

Entity	Nodes
Entity 1	[Node 1]
Entity 2	[Node 2]
Entity 3	[Node 3]

Table 4.5: Entity-node-mappings after second improvement iteration. By removing uniquely mapped nodes from other mappings, the existing unique mappings can be extended.

4.5 Summary

In this chapter, we introduced the reader into the methodology followed in our work. First, we collected data from the Lightning Network over a certain period. For this, we used LND and created snapshots from the Lightning Network on an hourly basis. Next, we linked them to data from the Bitcoin blockchain discussed in detail in section 4.2. We extracted relevant data from the Bitcoin blockchain using BlockSci. We also enriched the already linked data by information from a cryptoasset analytics tool discussed in detail in section 4.3. We used GraphSense to link Lightning Network data to entities. As a final step, we mapped entities to nodes. To do so, we applied a heuristic explained in section 4.4. Summarised, the presented methodology shows how data across GraphSense, the Bitcoin blockchain and the Lightning Network can be linked and mapped and what information can be obtained.

CHAPTER 5

Results

At the beginning of the this paper in hand we formulated the following three research questions in chapter 1.

- What kind of information can be deduced from second layer networks such as Lightning Network for Bitcoin, based on the footprints left on the blockchain and the second layer network communication itself?
- What kind of information that in common belief is private per default can be leaked within second layer networks?
- How can we integrate off-chain analysis into existing cryptoasset analytics tools and methods?

In this chapter, we will partially present answers to these questions, we will present results of off- and on-chain analysis, results based on the analysis of linked data from on- and off-chain networks and results achieved by analysing the enriched and linked data. Furthermore, we will show an example of an entity-node-mapping and finally summarise the key findings of our results.

5.1 Deduced Information from both Networks

To answer the first research question, we collected 1,500 hourly snapshots from the Lightning Network in about two months. At the same period, we also extracted relevant transactions from the Bitcoin blockchain. Below we will give some insights into the Lightning Network and the evolution over time within this period and show results concerning the size of nodes and channels, the distribution of balances, key figures related to nodes and channels and network centrality measures. After that, we will also present the evolution over a longer period of all transactions and of 2-of-2 multi-signature transactions on the Bitcoin blockchain, which are the fundamental building block in the transaction scheme introduced in [208] to identify potential Lightning Network channels. As a last step, we will present the analysis of true- and false-positive Lightning Network channel transactions as well as the results based on the open- and closed-state of channels.

5.1.1 Size

During the hourly snapshots, we collected 6,877 nodes and 43,151 channels in total. Right at the end of the regular snapshots, the locally operating LND node was aware of 6,419 nodes and 36,405 channels. As per 2^{nd} February 2020 at 08:28:19 a.m. (CET), about 16% of all 43,151 recorded channels were already closed. Therefore we checked the respective transaction output of the funding transaction of each channel. If it was already spent, the channel was already closed and if not, the channel was still open. As already mentioned, the LND client removes inactive channels and nodes from its maintained network state. We bypassed this implementation, but not from the beginning of our snapshots. There are channels in the set of all unique channels which are not part of every snapshot. In detail, there are 13,483 channels which are not part of every snapshot during their active time. That means, these channels were noticed by the node e.g. at time t and at time t+2, but were not recognized at time t+1. The same is true for 1,129 nodes of all unique nodes. Figures 5.1 and 5.2 show the progression of nodes and channels over time within the captured time period. As the reader can see, due to the spreading mechanism of channels and nodes, it took some time for the locally operating node to fully synchronize all the network details at the beginning. Besides, there is a jump in node and channel count between the snapshots recorded on 31^{st} December 2019 at 10:00:01 am and 11:00:01 a.m. (CET) which can be recognized in both figures. The node count increased by 101 and the channel count increased by 983 within this single hour. Online research resulted in no information about any event or happening on the Lightning Network in this period.

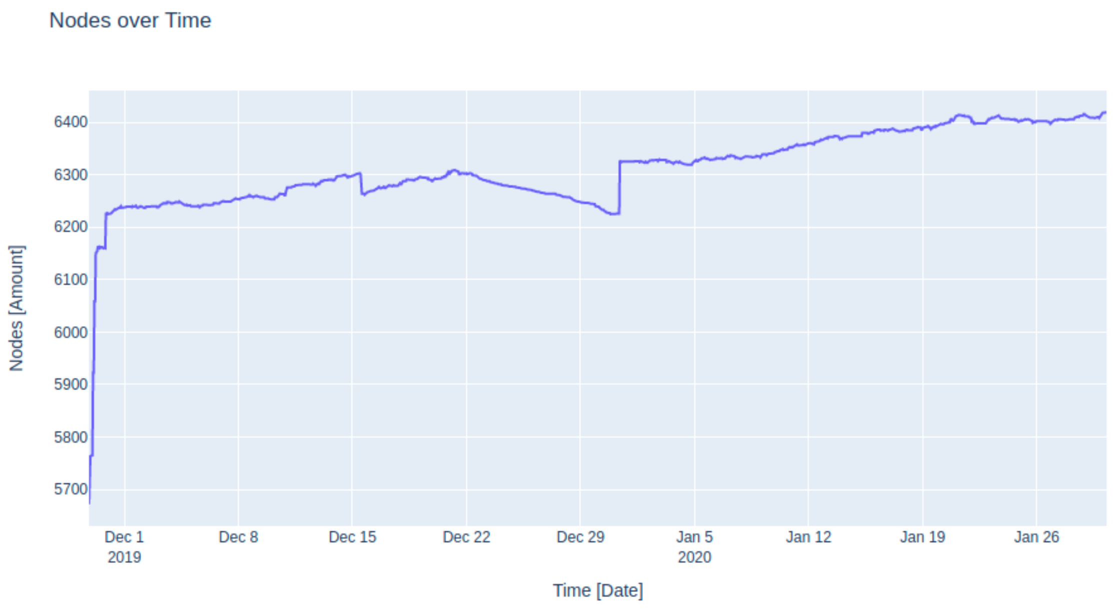


Figure 5.1: The number of active nodes over the time period of collecting data from the Lightning Network.

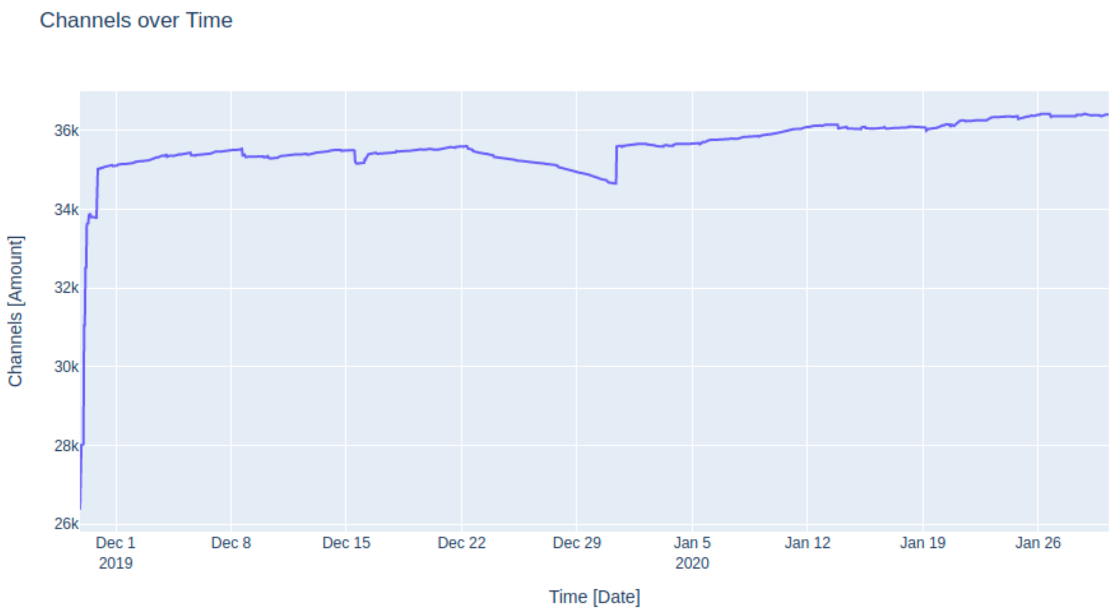


Figure 5.2: The number of active channels over the time period of collecting data from the Lightning Network.

5.1.2 Balances

We analysed the Lightning Network concerning the channel balances. As mentioned in 3.2.2, there are more than 800 BTC maintained by the Lightning Network. The average channel balance is about 2,456,929.3 Satoshi, which is about 0.024569 BTC. The channel balance median is 500,000 Satoshi, which is 0.005 BTC. The factor of 5 between the median and the average indicates that there are some big channels concerning the balances. Figure 5.3 shows the distribution of balances concerning the channels. The vast majority of channels has a balance less than 0.2 BTC and one channel has a balance of about 2.7 BTC. This distribution indicates the centralization of the Lightning Network. Table 5.1 shows the balances of the top five channels with the highest channel balances. It is conspicuous that 10 channels have the same balance as the last listed channel in the table. For a better overview, we only listed one of them in the table below.

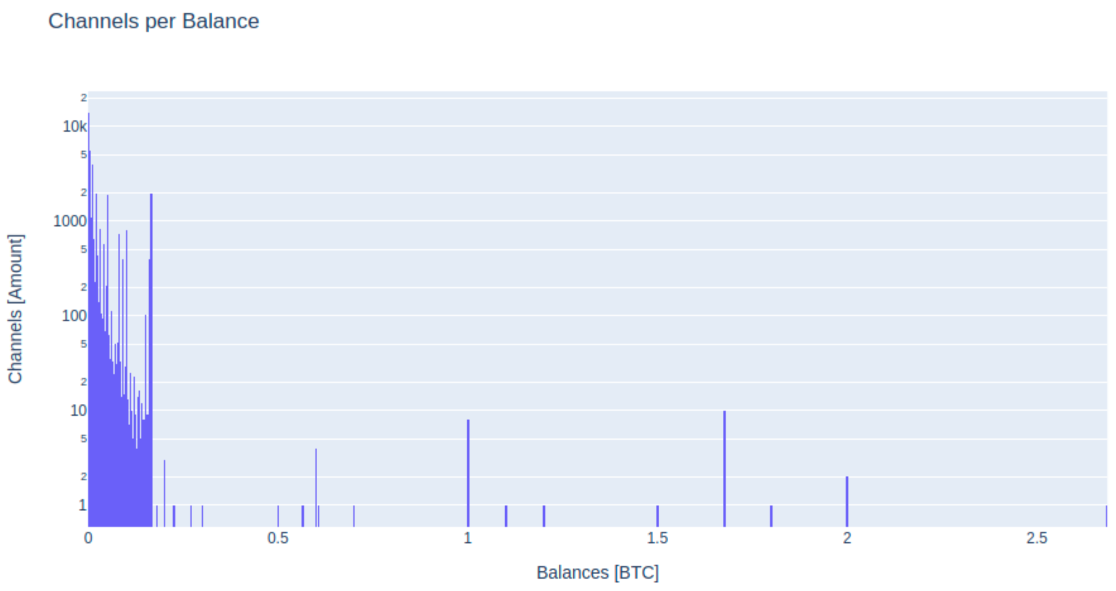


Figure 5.3: The amount of channels per balance.

Channel Point	Balance in Satoshi
f6b84ef2cabe68afb50d9a626014508ea8084235c6105c3bb14fc9c79583b451:0	268,435,456
6f3dca9d9b5e32dea00847936dd8d43d5e3657b045f9533d4ad4bafa7d05da15:1	200,000,000
95b7acf7e93b97aeab4d0d73d71c99b0b3ad4cbb1412d9d0928f0d43acec4157:0	200,000,000
fc6b607a528fe96225b64e9ad64253148d4a950ce4c418b9f66cb9a118434364:1	180,000,000
c92918abf887052465aaba959d2a3a515e8dca91c5e3bd6689f142c903ef0924:1	167,772,150

Table 5.1: The top five channels with the highest channel balances.

5.1.3 Nodes and Channels

We related the channels to the nodes to identify the degree distribution of the nodes. Figure 5.4 shows the distribution of channels concerning the nodes. Some nodes have many channels, but many nodes have only one channel, which means, these nodes cannot be used for routing payments across the network unless they are either sender or receiver of payments. To enable multi-hop payments within the Lightning Network, nodes have to have more than one channel to act as an intermediary. This distribution also indicates the centralization of the Lightning Network. From a security perspective, targeted attacks against central nodes can easily split the network in several parts. Table 5.2 shows the channel count for the top five nodes with the most channels.

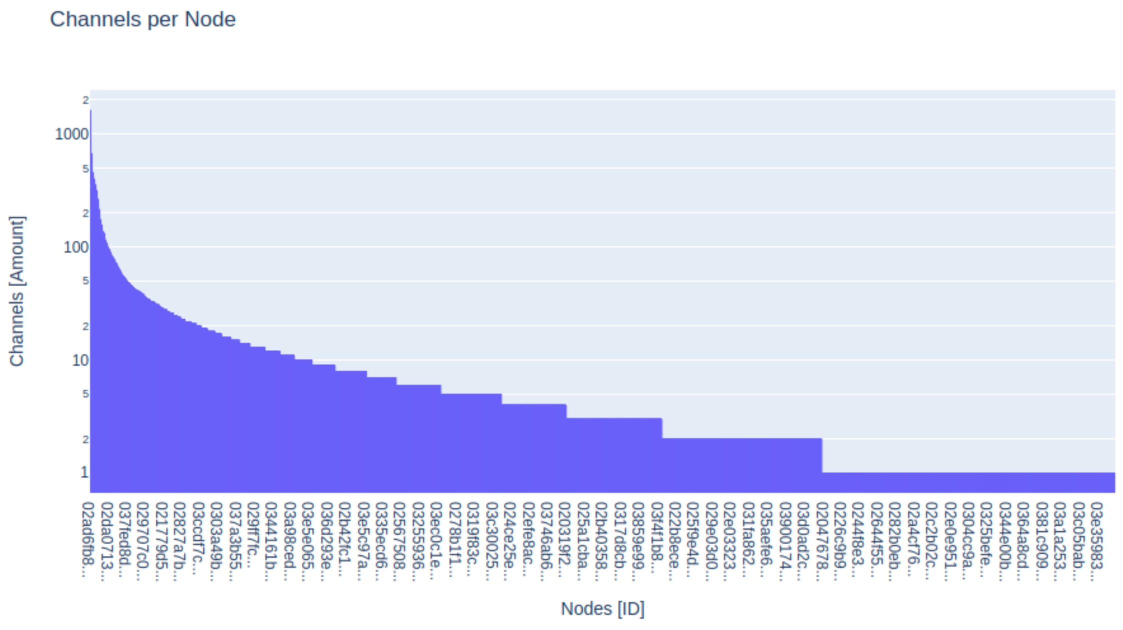


Figure 5.4: The amount of channels per node.

Node	Channel Count
02ad6fb8d693dc1e4569bcedefadf5f72a931ae027dc0f0c544b34c1c6f3b9a02b	1,630
0331f80652fb840239df8dc99205792bba2e559a05469915804c08420230e23c7c	1,308
03864ef025fde8fb587d989186ce6a4a186895ee44a926bfc370e2c366597a3f8f	1,193
0217890e3aad8d35bc054f43acc00084b25229ecff0ab68debd82883ad65ee8266	1,018
0279c22ed7a068d10dc1a38ae66d2d6461e269226c60258c021b1ddcdfe4b00bc4	880

Table 5.2: The top five nodes with the most channels.

5.1.4 Centrality Measures

We investigated different centrality measures for the Lightning Network graph to get an overview of the importance of the most central nodes and the centralization of the

Lightning Network in general. Table 5.3 shows the top five nodes with the highest degree centrality. The degree centrality is a measure for the amount of channels one node has with other nodes. Table 5.4 shows the top five nodes with the highest closeness centrality. The closeness centrality of a node is a measure for the average path length of a node to all the other nodes. Table 5.5 shows the top five nodes with the highest betweenness centrality. The betweenness centrality is a measure for how often a node acts as a bridge for any two other nodes on the shortest path between them. In general, high centrality measures of nodes indicate that the network is centralized, at least in some major parts. It is conspicuous that node '02ad6fb8...' has the highest measures in all three centrality types. This is also the node with the most channels as shown in section 5.1.3. In later sections we will again highlight the importance of this node.

Node	Centrality
02ad6fb8d693dc1e4569bcedefadf5f72a931ae027dc0f0c544b34c1c6f3b9a02b	0.214078
0331f80652fb840239df8dc99205792bba2e559a05469915804c08420230e23c7c	0.155759
0217890e3aad8d35bc054f43acc00084b25229ecff0ab68debd82883ad65ee8266	0.137435
03864ef025fde8fb587d989186ce6a4a186895ee44a926bfc370e2c366597a3f8f	0.131617
0279c22ed7a068d10dc1a38ae66d2d6461e269226c60258c021b1ddcdfe4b00bc4	0.116201

Table 5.3: The top five nodes with the highest degree centrality.

Node	Centrality
02ad6fb8d693dc1e4569bcedefadf5f72a931ae027dc0f0c544b34c1c6f3b9a02b	0.495602
0279c22ed7a068d10dc1a38ae66d2d6461e269226c60258c021b1ddcdfe4b00bc4	0.460296
03864ef025fde8fb587d989186ce6a4a186895ee44a926bfc370e2c366597a3f8f	0.457637
0331f80652fb840239df8dc99205792bba2e559a05469915804c08420230e23c7c	0.452014
0242a4ae0c5bef18048fbecf995094b74bfb0f7391418d71ed394784373f41e4f3	0.441927

Table 5.4: The top five nodes with the highest closeness centrality.

Node	Centrality
02ad6fb8d693dc1e4569bcedefadf5f72a931ae027dc0f0c544b34c1c6f3b9a02b	0.177765
0331f80652fb840239df8dc99205792bba2e559a05469915804c08420230e23c7c	0.101515
03864ef025fde8fb587d989186ce6a4a186895ee44a926bfc370e2c366597a3f8f	0.093676
0217890e3aad8d35bc054f43acc00084b25229ecff0ab68debd82883ad65ee8266	0.074861
0279c22ed7a068d10dc1a38ae66d2d6461e269226c60258c021b1ddcdfe4b00bc4	0.056917

Table 5.5: The top five nodes with the highest betweenness centrality.

5.1.5 2-of-2 Multi-signature Transactions

We analysed 2-of-2 multi-signature transactions which are the main part of the identification scheme of potential Lightning Network channel transactions from the Bitcoin blockchain introduced in [208]. To get a better overview of the amount of 2-of-2 multi-signature transactions compared to all transactions per block, we analysed all the transactions from each block starting at block 400,000 and ending at block 615,432. Figure 5.5 shows the percent of 2-of-2 multi-signature transactions per block. As the reader can see, the amount of 2-of-2 multi-signature transactions started to increase from about block height 500,000, which was mined on 18th December 2017 at 07:35:25

p.m. (CET). At the beginning of 2018 the first beta of LND version 0.4 was released [216]. This partially validates the approach presented in [208] as it seems that the amount of 2-of-2 multi-signature transactions are correlating with the launch and usage of LND respectively the Lightning Network. In general, between 0.05% and 0.25% of all transactions in 2019 were 2-of-2 multi-signature transactions. This share indicates the low adoption of the Lightning Network compared to other transactions at the moment.

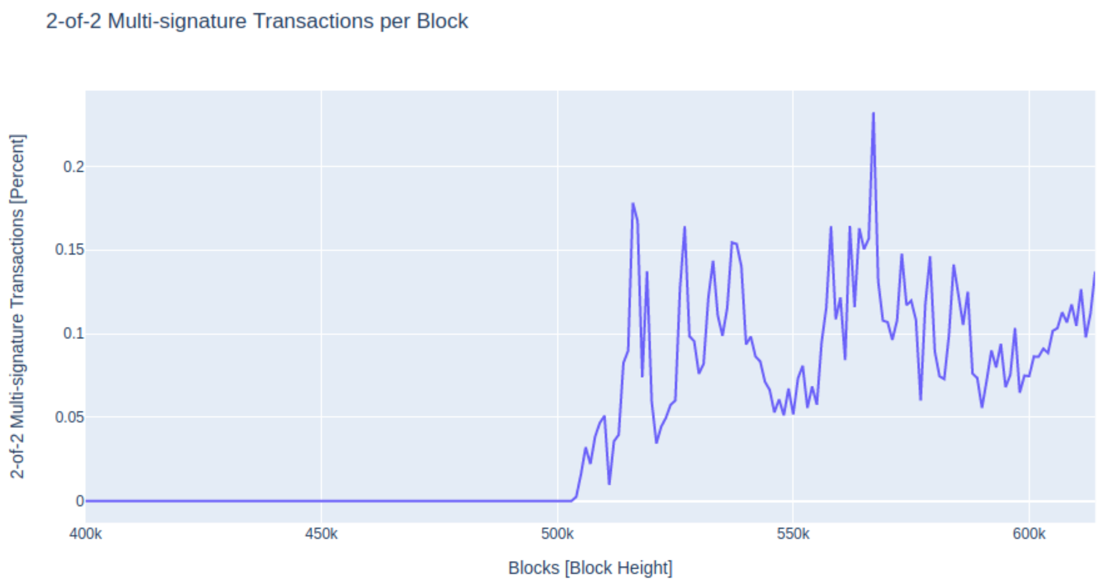


Figure 5.5: The percent of 2-of-2 multi-signature transactions per block.

5.1.6 Potential Lightning Network Channel Transactions

As mentioned in 4.1.2, we analysed 9,381 blocks and 19,094,656 transactions between block heights 605,798 and 615,178. Among these about 19 million transactions, we identified 21,787 (0.1%) potential Lightning Network channel transactions by applying the transaction scheme introduced in [208]. These potential transactions contained 1,954 (9%) timelocked transactions which are uncooperative channel closings between two channel participants and 19,833 (91%) cooperative channel closings.

We compared the potential on-chain Lightning Network transactions with the collected off-chain data. 6,486 transactions (30%) of the on-chain identified potential Lightning Network transactions were verified to be true-positives and 15,301 transactions (70%) were false-positives. Compared to the results presented in [208], the rate of true-positives is low as they verified about 75% of potential transactions to be actual channels. We verified our identification method by extracting potential Lightning Network in the same block range as M. Nowostawski and J. Ton and extracted the same amount of potential Lightning Network channels in that block range. As we are sure that our on-chain identification method is correct, possible reasons for the gap between 30% and 75% can be other applications or use cases producing similar footprints on the blockchain that were not active during the time the authors published their paper or that today partially more private channels compared to public channels were active than in the past. E.g. the research team of BitMEX estimates the amount of private channels compared to public channels to about 28% as per January 2020 [217].

Furthermore, we analysed the 6,486 true-positives concerning the block height of the funding transaction, closing transaction and the block durations of the channels. Figure 5.6 shows the block heights of the funding transactions and figure 5.7 shows the block heights of the closing transactions of all true-positive channels. As the reader can see, the number of channel openings increased over time with the usage of the Lightning Network and the number of channel closings in a certain range is constant over time. Figure 5.8 shows the duration in blocks of the true-positive channels. Most of the channels had a short duration whereas some had a very long duration. The average duration of these channels is 24,938.5 blocks which is about 173.2 days. The channel duration median is about 19,286 blocks which is about 133.9 days. It can be seen that the median and the average are nearly the same which means the data is uniformly distributed.

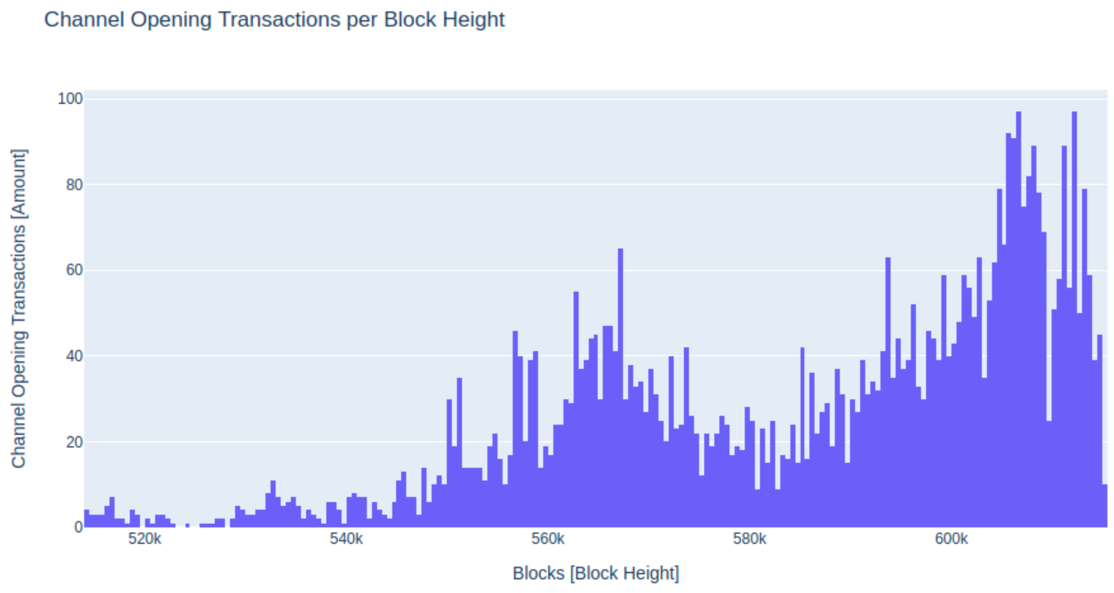


Figure 5.6: The amount of channel opening transactions per block.

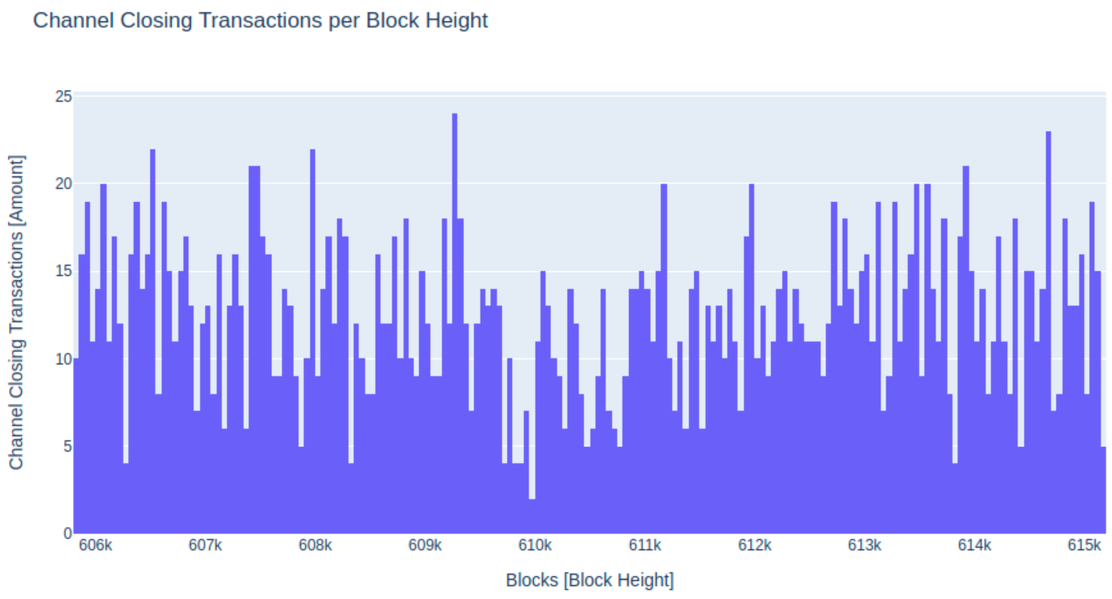


Figure 5.7: The amount of channel closing transactions per block.

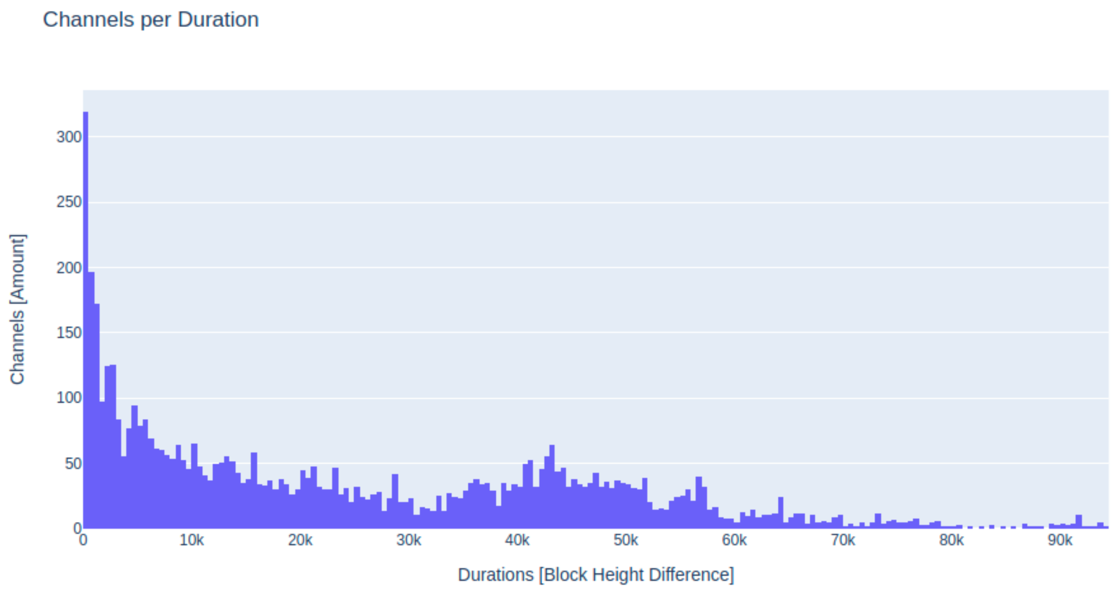


Figure 5.8: The amount of channels per channel duration.

5.1.7 State Verification of Probably Closed Channels

We verified the state of absent channels which were present at any time t, but were not present at any time t+1. This can be caused by two reasons: first, the channel was actually closed and second, the node removed the channel from its maintained network graph due to inactivity. We analysed all channels that were present at any time t but not at the time of the latest snapshot. We identified 6,745 channels matching this scheme. Next, we validated the state of these channels on the Bitcoin blockchain by investigating the respective transaction output of the channel funding transaction. If the output was spent, the channel was closed and if it was not spent, the channel was still open. 6,493 (96%) of the channels were closed and 252 (4%) were still open.

5.2 Private Information from the Lightning Network

After we answered the first research questions, we analysed linked and enriched data concerning the privacy of the participants to answer the second research question. We will present details on node aliases, IP addresses, geo-locations and on Bitcoin address and Bitcoin key reuse. Besides, we will present results of enriched data by information from GraphSense such as the analyses of entities and entity-node-mappings.

5.2.1 Node Aliases

We noticed 6,877 unique nodes during the captured period of about 2 months. 5,111 (74.3%) of them have an alias which in some cases makes it possible to link a Lightning Network node to a public service. In section 5.2.6 we will present one example where we linked an entity to a node which is a wallet service.

5.2.2 Node IP Addresses and Locations

We analysed the IP addresses and based on these, the geo-location of the nodes. We observed 6,877 unique nodes during the snapshots. Table 5.6 shows the network addresses related to the count of nodes. It is evident that most nodes are either without any address information or hidden by the Tor network. Most of the other IP addresses operate less than 5 nodes. IP address ’195.95.224.2’ is located in Paris and IP address ’188.165.223.61’ is located in Roubaix, both in France. This implies that the biggest not hidden node operators are located in France. Figure 5.9 shows the geo-locations of the 250 nodes with the most channels. It is obvious that the Lightning Network is highly concentrated in Europe and Northern America. Table 5.7 shows the locations of the top five nodes with the most channels. For geo-location of IP addresses we used the service ’ipstack.com’ [218].

Address	Node Count
No address	2,556
Tor network	1,546
Others	37
195.95.224.2	20
188.165.223.61	5

Table 5.6: The top five nodes with the most channels.

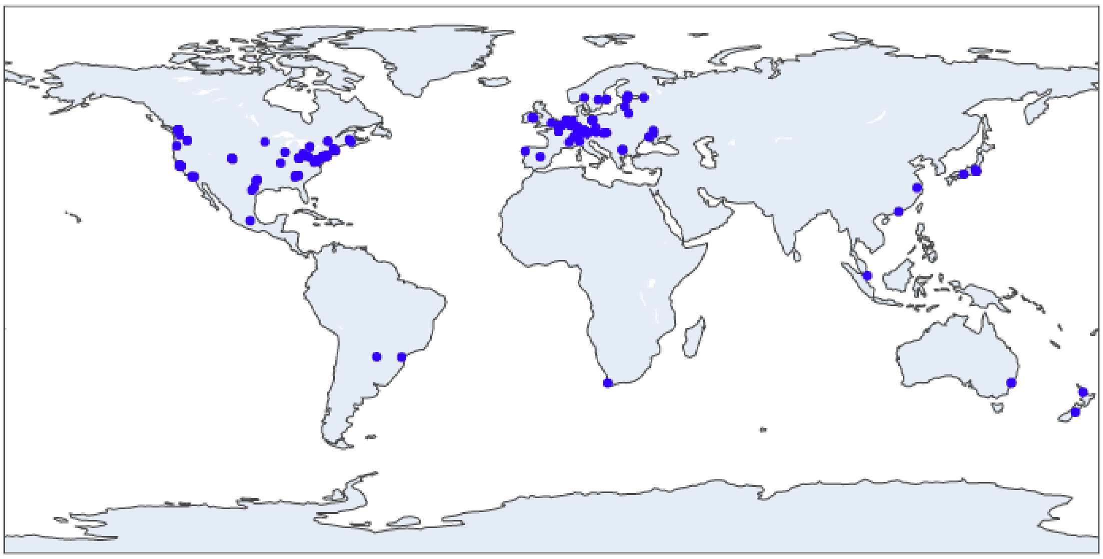


Figure 5.9: The locations of the 250 nodes with the most channels.

Node	Location
02ad6fb8d693dc1e4569bcedefadf5f72a931ae027dc0f0c544b34c1c6f3b9a02b	Clifton (USA)
0331f80652fb840239df8dc99205792bba2e559a05469915804c08420230e23c7c	Ashburn (USA)
03864ef025fde8fb587d989186ce6a4a186895ee44a926bfc370e2c366597a3f8f	Ashburn (USA)
0217890e3aad8d35bc054f43acc00084b25229ecff0ab68debd82883ad65ee8266	New York (USA)
0279c22ed7a068d10dc1a38ae66d2d6461e269226c60258c021b1ddcdfe4b00bc4	Duluth (USA)

Table 5.7: The locations of the top five nodes with the most channels.

5.2.3 Bitcoin Address Reuse

We analysed all funding transaction input addresses and whether already closed, all closing transaction output addresses of all 43,151 unique channels which we recorded during the period of the Lightning Network snapshots. 1,039 out of 74,770 (1.39%) input addresses were reused among other input addresses of channel funding transactions. Table 5.8 shows the most reused input addresses among other input addresses. 4 out of 10,427 (0.04%) output addresses were reused among other output addresses of channel closing transactions. Table 5.9 shows the most reused output addresses among other output addresses. The amount of both, reused output addresses and total output addresses of channel closing transactions is very low because most of the channels were still open and had no closing transaction. We also compared input and output addresses combined. 3,297 out of 85,206 (3.87%) addresses were reused in either input of channel funding transaction or output of channel closing transaction. Table 5.8 shows the most reused input or output addresses among other input or output addresses. The analysis of address reuse implies that some channels belong to the same address and that some channels are successors of previous channels. In general, the low count of reused addresses is most likely caused by the hierarchical deterministic wallet of the LND clients [178].

Address	Reuse Count
bc1qfl93evayrv7dythlnvvkwwz4v0kmwc686ke4gq	82
bc1q99tp7d30man0h86y0wh5yzzv99x2qvuc9dsp8u	42
bc1qa9vz2rlq6e548q4vu9xms46ghfl53u2k7p7k3w	39
bc1qzjap7cv6d4qe3psnytrjvwu5nvtx4yjtdg5lux	26
bc1qh3eppzs6yypgd20g5lmer9zgqqmn6meud6vgfd	22

Table 5.8: The top five addresses with the highest reuse count among input addresses only and among input or output addresses.

Address	Reuse Count
bc1qm9lk7z8v0nmt6wvq4zym7cwqu4a2w8a40rahx2	2
bc1qyadyyz2nrp8j39sruj6rrx5v7n0efaf8spevh8	2
bc1qr2dsl65vxuene3qedd0a9f06tx780vkh4unkrp	2
bc1qlvjwsf2shnk7tdpk0hdl3zfs20d376hmcq5nt5	2

Table 5.9: The top four addresses with the highest reuse count among output addresses only.

5.2.4 Bitcoin Key Reuse

We analysed the involved Bitcoin public keys in each channel and compared each key with all the others, to analyse the reuse of public keys. Only 62 (0.07%) out of 86,302 keys were reused once. This low count is most likely also caused by the hierarchical deterministic wallet of the LND clients [178].

5.2.5 Analysis of Entities

We identified 47,948 entities based on the recorded 43,151 unique channels and investigated the balance distribution of the identified entities and the distribution of entities concerning their total amount received and spent. Additionally, we also analysed entities concerning the amount of their linked addresses as well as the active days, meaning the time passed by between the first and the last transactions of an entity. Below we will present the results of these approaches and mechanisms.

As mentioned, we analysed the balances of the entities. Figure 5.10 shows the entities concerning the total balances. It can be seen that almost all of the entities have a balance of zero. This means that the values are already transferred to other entities. Moreover, we analysed the total amount of all the entities received respectively spent. Figure 5.11 shows the entities concerning the total amount received and figure 5.12 the entities concerning the total amount spent. Again, the majority of the entities have received and spent low amounts and some entities have received and spent very high amounts. After that, we analysed the number of addresses related to each entity. Figure 5.13 shows the distribution of entities concerning count of their related addresses. The majority of all the identified entities only has one single related address. This means that probably most of the actors within the Lightning Network are small entities or that GraphSense has not been able to cluster them so far. Finally, we analysed the active period of each entity. Figure 5.14 shows the entities concerning their active time. It is evident that entities are tendentially active for only a shorter period rather than for a longer one.

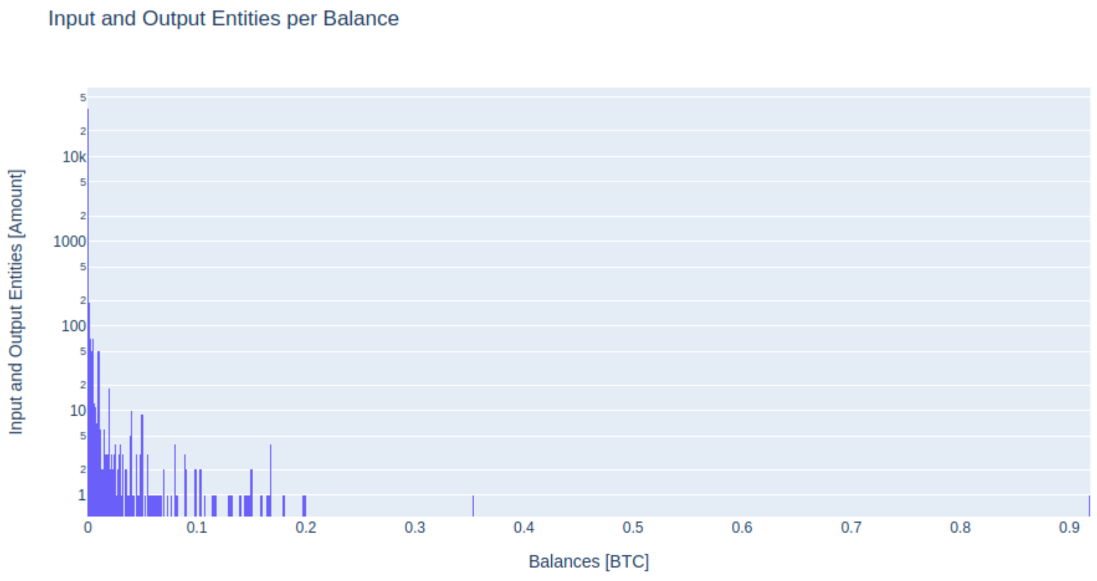


Figure 5.10: The amount of entities per total balance.

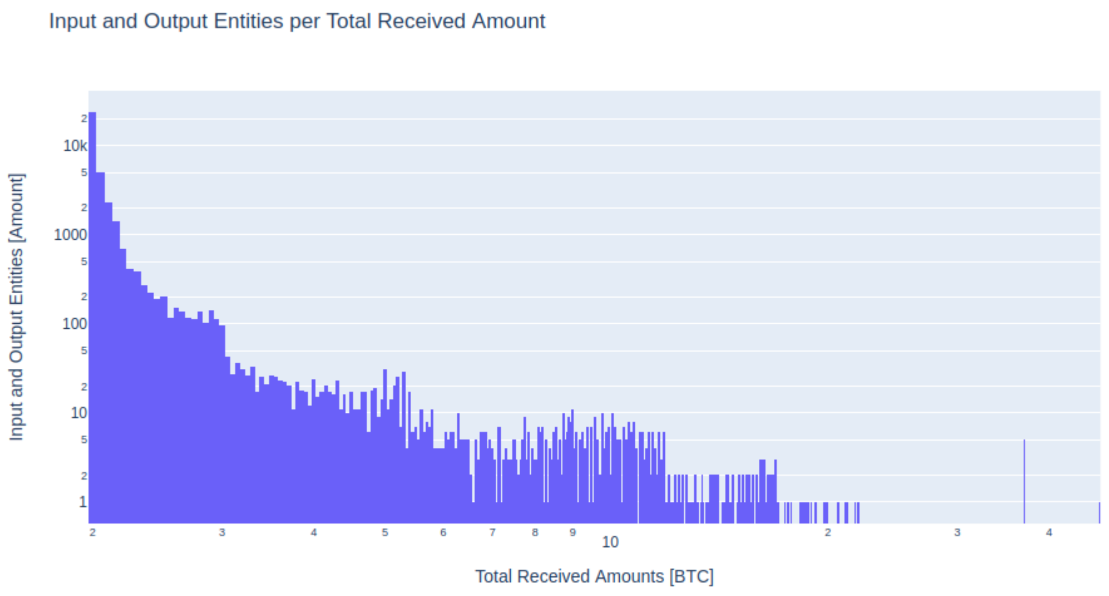


Figure 5.11: The amount of entities per total received amount.

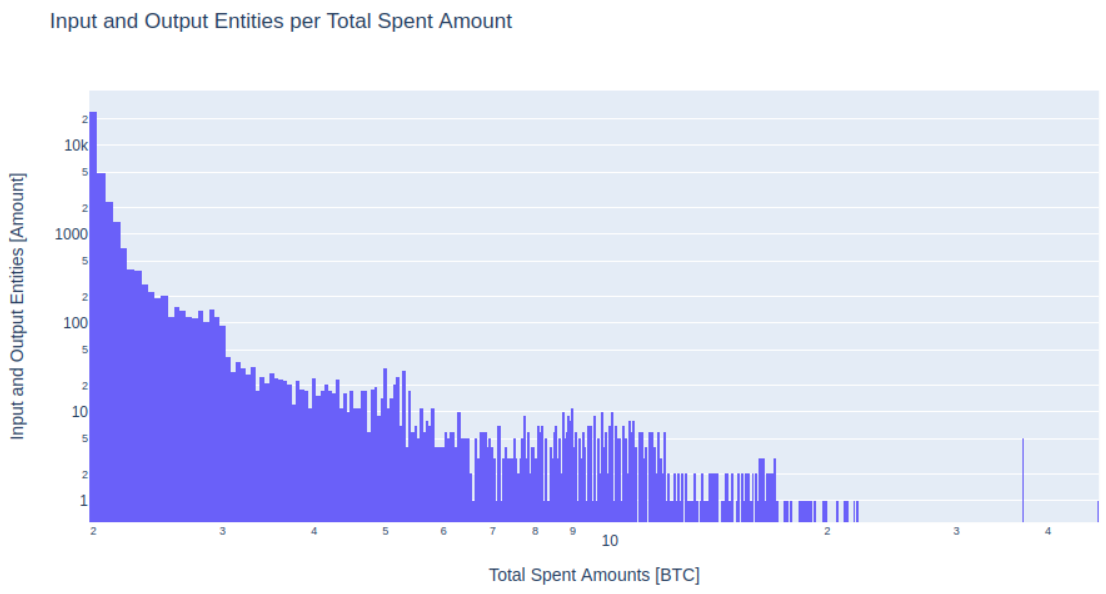


Figure 5.12: The amount of entities per total spent amount.

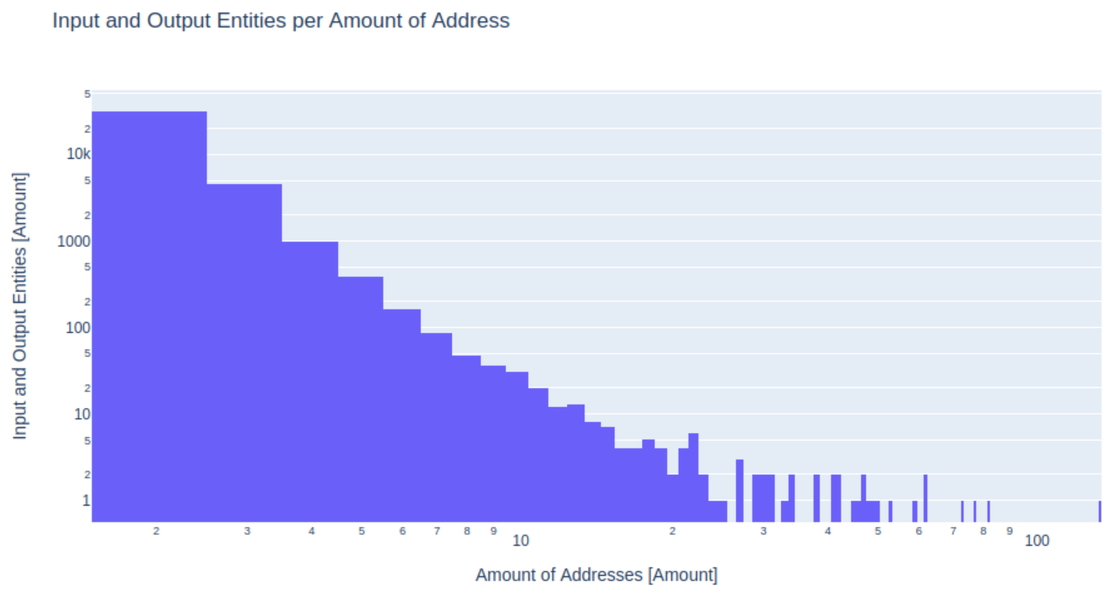


Figure 5.13: The amount of entities per number of related addresses.

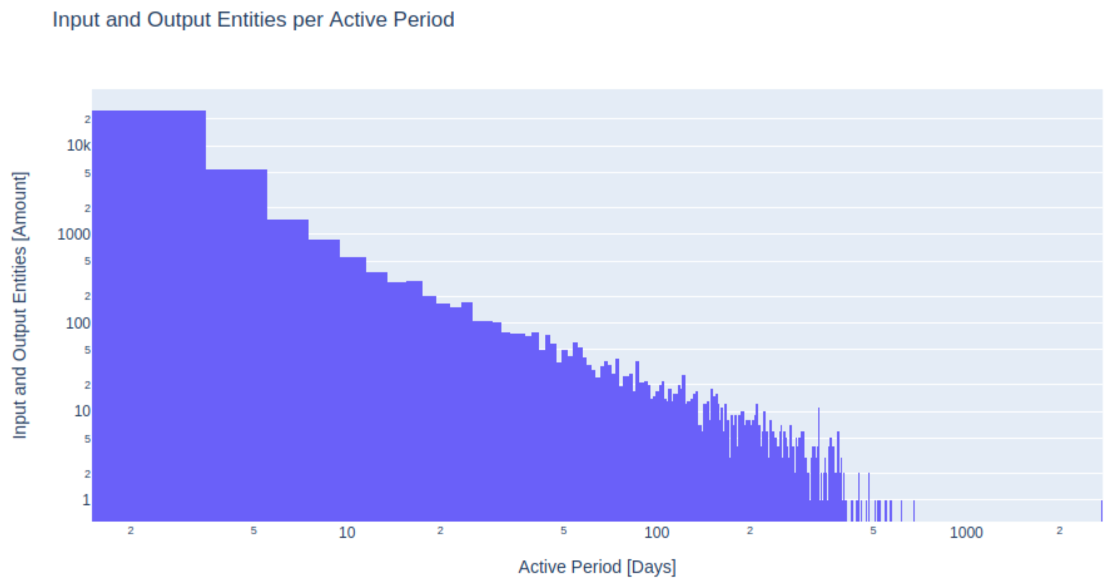


Figure 5.14: The amount of entities per active days.

Further, we analysed the enriched data concerning the amount of channels per entity. Figure 5.15 shows the distribution of entities concerning the amount of channels based on both, the funding transaction input and closing transaction output addresses. The vast majority of the entities is linked to one single channel, but there is an entity that is linked to 145 channels. Table 5.10 shows the top five GraphSense entities with the most linked channels based on both, the funding transaction input and closing transaction output addresses.

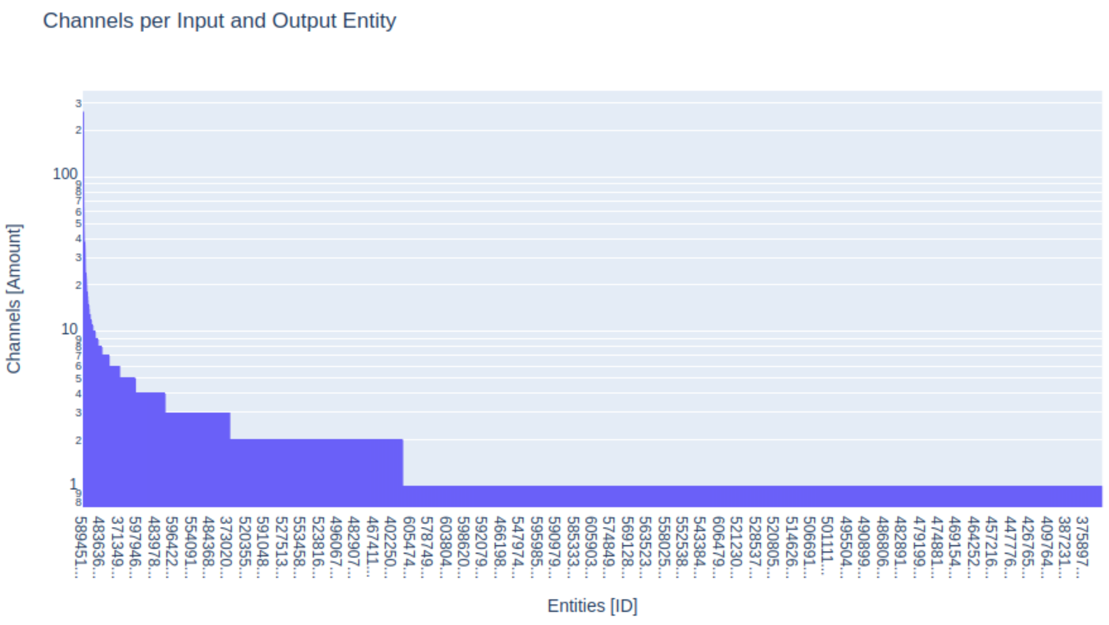


Figure 5.15: The amount of channels per GraphSense entity based on funding transaction input and closing transaction output addresses.

Entity	Channel Count
493979339	145
589451869	100
586775838	78
602018841	60
585848060	48

Table 5.10: The top five GraphSense entities based on funding transaction input and closing transaction output addresses with the most channels.

Next, we analysed the enriched data concerning the balances of all channels per entity. Figure 5.16 shows the distribution of entities concerning the total channel balances based on both, the funding transaction input and closing transaction output addresses. The vast majority of the entities have low balances among their linked channels under control. On the other hand, there are also some entities having high balances among their linked channels under control. There is especially one entity having about 46 BTC among its linked channels under control. Table 5.11 shows the top five GraphSense entities with the highest total channel balances based on both, the funding transaction input and closing transaction output addresses. Three additional entities have the same total channel balance among their linked channels under control like the last listed entity in the table. For a better overview, we only listed one of them in the table below.

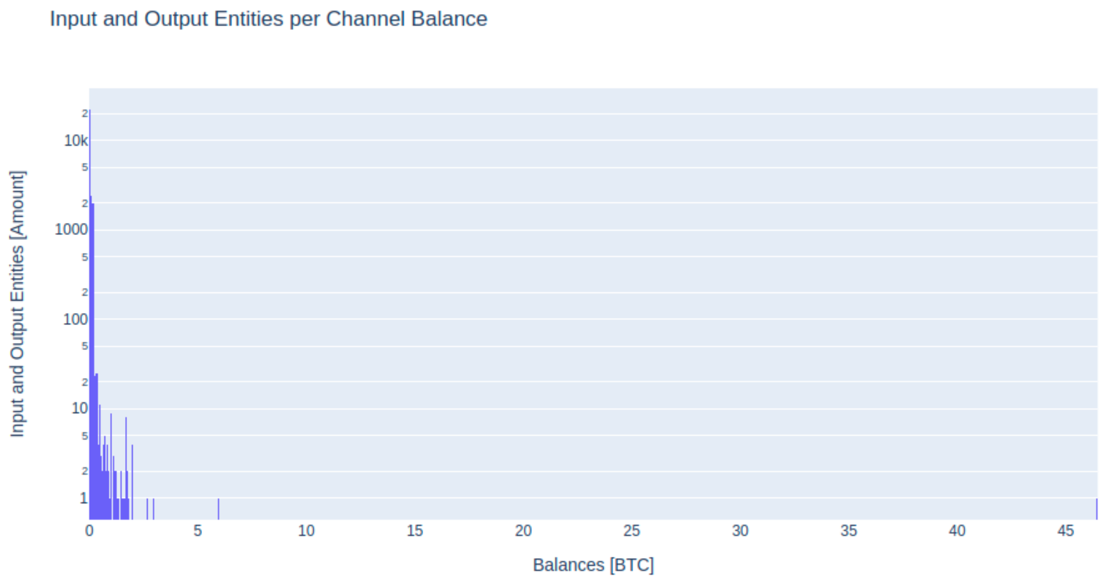


Figure 5.16: The amount of total channel balances per GraphSense entity based on funding transaction input and closing transaction output addresses.

Entity	Total Channel Balance in Satoshi
493979339	4,646,567,364
605916288	595,762,020
586775838	296,210,000
602028994	268,935,456
586731621	200,000,000

Table 5.11: The top five GraphSense entities based on funding transaction input and closing transaction output addresses with the most balances based on all related channels.

5.2.6 Analysis of Mapped Entities to Nodes

We identified 47,948 entities based on the recorded 43,151 unique channels. These channels are related to 6,877 unique nodes. By applying the first step of our heuristic which we discussed in detail in section 4.4, we uniquely mapped 2,260 entities to one single node. By applying the second step in successive processing rounds, we increased the unique entity-node-mapping from 2,260 to 8,148 entities. At the end, we verified that there was no violating data in the result of this heuristic. This means, there was no single entity, which was mapped to different nodes, but as the reader can imagine, there were different entities mapping to the same node. With this information, we could heuristically uniquely map 6,773 (98.5%) of the 6,877 nodes to entities. Table 5.12 shows the heuristical mapping of the top five entities with the most channels to the respective Lightning Network nodes. Table 5.13 shows the heuristical mapping of the top five entities with the highest total channel balances to the respective Lightning Network nodes.

Entity	Node	Node Count / Channels
493979339	03021c5f5f57322740e4ee6936452add19dc7ea7ccf90635f95119ab82a62ae268	145/145
589451869	02ad6fb8d693dc1e4569bcedefadf5f72a931ae027dc0f0c544b34c1c6f3b9a02b	100/100
586775838	02b77853653ff464339740c06e927ddcc5540487847256d342b98909532853b13d	78/78
602018841	036b343eb46c5db996d3d1e2c6cc9742cbfa7e3b4146d4b4b0aef694b6d12960c8	60/60
585848060	021607cfce19a4c5e7e6e738663dfafbbbac262e4ff76c2c9b30dbeefc35c00643	48/48

Table 5.12: The top five GraphSense entities with the most channels heuristically linked to Lightning Network nodes.

Entity	Node	Node Count / Channels
493979339	03021c5f5f57322740e4ee6936452add19dc7ea7ccf90635f95119ab82a62ae268	145/145
605916288	0254ff808f53b2f8c45e74b70430f336c6c76ba2f4af289f48d6086ae6e60462d3	21/21
586775838	02b77853653ff464339740c06e927ddcc5540487847256d342b98909532853b13d	78/78
602028994	03864ef025fde8fb587d989186ce6a4a186895ee44a926bfc370e2c366597a3f8f	3/3
586731621	03abf6f44c355dec0d5aa155bdbdd6e0c8fefe318eff402de65c6eb2e1be55dc3e	3/3

Table 5.13: The top five GraphSense entities with the highest total channel balances heuristically linked to Lightning Network nodes.

Next, we analysed the largest entity with the most linked channels and most linked channel balances which we will show hereinafter.

Largest Entity-Node-Mapping

In both analyses, channel count and total channel balances per GraphSense entity, one entity respectively Lightning Network node was the one with the most channels and the highest total channel balances. All five Bitcoin addresses shown in table 5.8 in section 5.2.3 are part of this GraphSense entity. Table 5.14 shows data about the entity loaded from GraphSense and table 5.15 about the node loaded from '1ML.com'. As shown in table 5.15, this entity respectively node is BlueWallet [219]. BlueWallet is a service which offers its users an easy to use Lightning Network wallet without the need of setting up a node, creating channels or providing liquidity.

Data	Value
Entity ID	493979339
Balance	485,077,398 Satoshi
First Transaction	30th of November 2018 at 00:32:52 a.m. (CET)
Last Transaction	29th of January 2020 at 10:39:22 p.m. (CET)
Number of Addresses	7,632
Incoming Transactions	11,909
Outgoing Transactions	498
Total Received	10,867,156,862 Satoshi
Total Spent	10,382,079,464 Satoshi

Table 5.14: The data of entity '493979339' on 8th February 2020 at about 04:00 p.m. (CET) known to GraphSense.

Data	Value
Node ID	03021c5f5f57322740e4ee6936452add1- 9dc7ea7ccf90635f95119ab82a62ae268
Owner	BlueWallet.io
Capacity	3,894,794,334 Satoshi
Channel Count	362
Connected Node Count	220
IP Address	207.180.244.165
Location	Nuremberg (Germany)

Table 5.15: The data of the Lightning Network node '03021c5f...' on 8th February 2020 at about 04:00 p.m. (CET) known to '1ML.com'.

In the above sections, we presented our results in detail. Before we will conclude our paper in the next chapter, we will summarise the major results.

5.3 Summary

In this chapter, we answered two out of three research questions formulated at the beginning of this paper. We presented our results concerning network size, channel balance distribution and centrality measures, figured out that the Lightning Network is centralized, at least in some parts. Some nodes operate many channels, but most of them operate only a few channels. The same is true for the amount of Bitcoins. Next, we presented results on the occurrence of 2-of-2 multi-signature transactions as they are a major building block of Lightning Network channels. We came to the conclusion that the occurrence of 2-of-2 multi-signature transactions highly correlates with the launch of an implementation of the Lightning Network. Further, we presented our results on identifying Lightning Network channels on the Bitcoin blockchain, statistics about true- and false-positive channel transactions and state verification of probably closed channels. We mentioned that we could map about 30% of on-chain identified possible Lightning Network channels to actual channels from the Lightning Network and that we identified about 4% of all absent channels from the network graph maintained by the LND node were not yet closed. After that, we presented results on analysis concerning privacy. We mentioned that about 74% of all nodes have an alias, which in general makes it possible to link a node to a public service. We presented results concerning IP addresses, geo-information of nodes and Bitcoin address and Bitcoin key reuses. We figured out that the Lightning Network is geographically concentrated in Europe and Northern America and that Bitcoin addresses and keys are only less reused in most cases. Next, we presented our results concerning enriched linked on- and off-chain data like analyses of entities and analyses of entity-node-mappings. In detail, we heuristically mapped 'BlueWallet.io', a node from the Lightning Network, to an entity from GraphSense.

CHAPTER 6

Discussion & Conclusion

In this chapter, we will conclude our paper concerning the most important aspects. We will summarise the key findings, reveal identified limitations of the presented methods and give some ideas for future work. As mentioned in chapter 3, the Lightning Network aims to overcome some major limitations of the Bitcoin blockchain. Beside scaling limitations, the Lightning Network is also intended to offer a better privacy to its users than the Bitcoin blockchain does. As presented in [15], it is a myth that off-chain transactions, the Lighting Network included, offer privacy by default.

For two months we collected off-chain data and performed different analyses on it, as presented in detail in chapter 5.1. When analysing the network and its metrics we also noticed that the Lightning Network is centralised, at least in some major parts, as mentioned also in [220]. The authors state that about 10% of nodes control 80% of the funds. We didn't calculate numbers on that, because as we were focusing on other aspects, but from a security perspective, more centralized systems are more vulnerable to targeted attacks than less centralized systems.

Beside off-chain analyses, we also performed on-chain analysis concerning possible Lightning Network transactions and presented our results in detail in section 5.1. About 0.1% of all transactions were 2-of-2 multi-signature transactions, a major building block of Lightning Network channels. Furthermore, we could map about 30% of potential Lightning Network channel transactions identified on the blockchain to actual Lightning Network channels. For this purpose we followed an approach introduced in [208]. The authors were able to map about 75% of potential identified Lightning Network channel transactions from the blockchain to actual Lightning Network channels. The gap between 30% and 75% could be caused by a higher amount of private channels compared to public channels or by other applications or use cases today which result in similar footprints on the blockchain like those of the interaction with the Lightning Network.

Next, we also mapped the channels and nodes based on the input addresses of channel funding transactions and whether a channel was already closed, on the output addresses of a channel closing transaction to already clustered entities by GraphSense. We presented our results in detail in section 5.2. With the presented method, we could provide additional information from the Lightning Network, such as node alias, IP addresses and connections between nodes to already clustered entities from GraphSense. We also presented a heuristic to map entities to nodes. In this way, we could map 98.5% of all recorded unique nodes within the network unique and heuristically to an entity in GraphSense. E.g. we identified one entity in GraphSense to be related to 'BlueWallet.io', a popular Lightning Network service, as we discussed in section 5.2.6.

To answer the third research question on how off-chain analyses can be integrated into existing cryptoasset analytics tools and methods, we will give some ideas in the section 6.2.

By linking on-chain and off-chain data and mapping it to an enriched dataset in the presented way, one can get an even deeper insight into the Bitcoin blockchain, the Lightning Network and GraphSense, as shown in several facets in chapter 5.

As we have summarised the key findings of our paper, we will introduce the reader into some limitations concerning our methods and approaches next.

6.1 Limitations

We collected the off-chain dataset by a locally operating LND node. As mentioned in 3.1.1, public channels and nodes are sent through the network by each node. On the other hand, private channels are only known to the two interacting nodes and their neighbors. The research team of BitMEX estimates the amount of private channels compared to public channels to about 28% as per January 2020 [217]. The fewer public channels within the Lightning Network are available, the fewer information is sent across the network and so also fewer information can be obtained. This leads to limited possibilities of linking off-chain data to on-chain data by the presented method.

Another limiting fact is the selected routing method within the second layer network. As mentioned in section 3.1.5, Lightning Network uses source routing. This means, the source node of a payment calculates the route to the destination of the payment. Therefore the node has to know all the public nodes and channels within the network which can be used for routing a payment. To ensure this, public nodes and channels are sent through the network. Once the Lightning Network uses local routing, which we discussed in section 3.1.5, it is not necessary to broadcast all the public channels and nodes across the network anymore. This also results in limited possibilities of collection data from the Lightning Network and consequently also in limited possibilities of linking on-chain data to off-chain data by the presented method.

It is not guaranteed that a node knows the complete state of the payment channel network at any time. As we only collected channels and nodes from the Lightning Network via a single locally operating LND node, we might not have recognized all the active channels and nodes within the Lightning Network during the captured period. This would lead to an incomplete linkage of on-chain and off-chain data and an incomplete mapping of entities to nodes.

As we have introduced some limitations concerning our methods and approaches, we will finally propose some ideas for possible areas of future work.

6.2 Possible Areas of Future Work

The knowledge of GraphSense can be increased in more detail. As mentioned in section 5.2.6, there was no single entity that was mapped to different nodes, but there were different entities that mapped to the same node. Knowing this, one could further increase the knowledge of GraphSense by merging those entities to one single entity. Besides, several nodes could be grouped by meta-information, e.g. from '1.ML.com'. For example there is a service called 'LNBIG.com' [221] operating several nodes, in detail 25 nodes at the time of this writing [222]. One can merge these nodes to one off-chain entity and so merge even much more GraphSense entities to one single entity. Also, there are probably more services like that, which could be clustered the same way.

The importance of the linked and enriched data can be increased. This could be achieved by providing additional information to the data, e.g. by using tagged on-chain addresses. One could check involved addresses in a channel against this dataset and in this way tag off-chain data as well.

The accuracy of the collected data and the presented method can be increased. This could be achieved by collecting channels and nodes via different nodes across the network, ideally located on different sites of the network. This will increase the coverage of the recognized part of the Lightning Network. The more nodes are used for data collection, the more channels and nodes will likely be recognized during the regular snapshots and the more on-chain entities can be provided with additional information. Besides, the amount of entities that could be merged, could be increased too.

Investigation on the network protocol level can be increased. As mentioned in section 5.2.2, we identified and located IP addresses of the biggest nodes concerning the channel count. The effort on analysing the network concerning revealing the node operators can be increased. Probably well-known exchanges can be identified to operate big and central nodes within the Lightning Network for example as well.

As mentioned in section 5.1.1, there was a jump in node and channel count between the snapshots on 31st December 2019 at 10:00:01 am and 11:00:01 a.m. (CET). The node count increased by 101 and the channel count by 983 within this single hour. Before this significant jump happened, there was a slower decrease of node and channel count over a few days. Probably one big node or service re-started or one node or service was released by another one. The reasons for such jumps could be investigated.

List of Figures

List of Tables

Bibliography

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *bitcoin.org*, page 9, 2008.

[2] CoinMarketCap. Historical Snapshot - 11 May 2013. https://coinmarketcap.com/historical/, October 2019. Accessed on 2019-10-12.

[3] CoinMarketCap. Cryptocurrency Market Capitalizations. https://coinmarketcap.com/, October 2019. Accessed on 2019-10-12.

[4] BLOCKCHAIN LUXEMBOURG S.A. Confirmed Transactions Per Day. https://www.blockchain.com/charts/n-transactions, February 2020. Accessed on 2020-02-24.

[5] BLOCKCHAIN LUXEMBOURG S.A. Market Capitalization. https://www.blockchain.com/charts/market-cap, February 2020. Accessed on 2020-02-24.

[6] BLOCKCHAIN LUXEMBOURG S.A. Market Price (USD). https://www.blockchain.com/charts/market-price, February 2020. Accessed on 2020-02-24.

[7] Jesse Yli-Huumo, Deokyoon Ko, Sujin Choi, Sooyong Park, and Kari Smolander. Where Is Current Research on Blockchain Technology?—A Systematic Review. *PLOS ONE*, 11(10):1–27, 2016.

[8] VISA. Visa fact sheet. https://www.visa.gr/dam/VCOM/download/corporate/media/visanet-technology/aboutvisafactsheet.pdf, October 2019. Accessed on 2019-10-12.

[9] Bitcoinist.com. Bitcoin Transactions Per Second Approaching All-Time High. https://bitcoinist.com/bitcoin-transactions-per-second-approaching-all-time-high/, February 2019. Accessed on 2020-03-07.

[10] ligthning.network. Lightning Network. https://lightning.network/, July 2019. Accessed on 2019-07-08.

[11] Joseph Poon and Thaddeus Dryja. The Bitcoin Lightning Network:. *lightning.network*, page 59, January 2016.

[12] DCI Institut and Hochschule Fresenius Hamburg. Paid Content Study 2018. `https://www.dci-institute.com/paidcontent-studie2018`, 2018. Accessed on 2019-11-16.

[13] PayPal. PayPal Österreich: Sicher & bequem bezahlen und Geld empfangen. `https://www.paypal.com/at/webapps/mpp/home`, March 2020. Accessed on 2020-03-31.

[14] Ghada Almashaqbeh, Allison Bishop, and Justin Cappos. MicroCash: Practical Concurrent Processing of Micropayments. *arXiv:1911.08520 [cs]*, November 2019. Accessed on 2019-11-23.

[15] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Off The Chain Transactions. Technical Report 360, http://eprint.iacr.org/2019/360, 2019. Accessed on 2019-10-04.

[16] en.bitcoin.it. SHA-256 - Bitcoin Wiki. `https://en.bitcoin.it/wiki/SHA-256`, November 2019. Accessed on 2019-11-02.

[17] en.bitcoin.it. RIPEMD-160 - Bitcoin Wiki. `https://en.bitcoin.it/wiki/RIPEMD-160`, November 2019. Accessed on 2019-11-02.

[18] bitcoinwiki.org. Base58 algorithm. All about cryptocurrency - BitcoinWiki. `https://en.bitcoinwiki.org/wiki/Base58`, March 2019. Accessed on 2019-11-02.

[19] Andreas M. Antonopoulos. 4. Keys, Addresses, Wallets - Mastering Bitcoin [Book]. `https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch04.html`, December 2014. Accessed on 2019-11-02.

[20] Daniel R L Brown. SEC 2: Recommended Elliptic Curve Domain Parameters. *sec.org*, page 37, November 2019. Accessed on 2019-11-02.

[21] Svetlin Nakov. ECDSA: Elliptic Curve Signatures. `https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages`, November 2019. Accessed on 2019-11-08.

[22] en.bitcoin.it. Secp256k1 - Bitcoin Wiki. `https://en.bitcoin.it/wiki/Secp256k1`, November 2019. Accessed on 2019-11-02.

[23] Ashish. Introduction to Zero Knowledge Proof: The protocol of next generation Blockchain. `https://medium.com/coinmonks/introduction-to-zero-knowledge-proof-the-protocol-of-next-generation-blockchain-305b2fc7f8e5`, October 2018. Accessed on 2019-11-23.

[24] Rui Zhang, Rui Xue, and Ling Liu. Security and Privacy on Blockchain. *arXiv:1903.07602 [cs]*, August 2019. Accessed on 2019-11-23.

[25] Yahya Hassanzadeh-Nazarabadi, Alptekin Küpçü, and Öznur Özkasap. LightChain: A DHT-based Blockchain for Resource Constrained Environments. *arXiv:1904.00375 [cs]*, March 2019. Accessed on 2019-12-14.

[26] Ling Ren. Analysis of Nakamoto Consensus. Technical Report 943, eprint.iacr.org, 2019. Accessed on 2019-11-21.

[27] Decryptionary.com. What is Chain Split? Get the definition here. `https://decryptionary.com/dictionary/chain-split/`, February 2020. Accessed on 2020-02-27.

[28] Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. A Survey of Distributed Consensus Protocols for Blockchain Networks. *arXiv:1904.04098 [cs]*, April 2019. Accessed on 2019-10-13.

[29] CoinMarketCap. Bitcoin price, charts, market cap, and other metrics. `https://coinmarketcap.com/`, December 2019. Accessed on 2019-12-07.

[30] bitcoinmining.com. What is the Bitcoin Mining Block Reward? `https://www.bitcoinmining.com/what-is-the-bitcoin-block-reward/`, December 2019. Accessed on 2019-12-07.

[31] Bitnodes.earn.com. Global Bitcoin nodes distribution. `https://bitnodes.earn.com/`, December 2019. Accessed on 2019-12-07.

[32] blockchain.com. Hash Rate. `https://www.blockchain.com/charts/hash-rate`, December 2019. Accessed on 2019-12-07.

[33] Vitalik Buterin. ethereum/wiki. `https://github.com/ethereum/wiki`, October 2019. Accessed on 2019-10-20.

[34] Ethereum Foundation. Home | Ethereum. `https://ethereum.org`, October 2019. Accessed on 2019-10-20.

[35] Demiro Massessi. Public Vs Private Blockchain In A Nutshell. `https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f`, September 2019. Accessed on 2019-11-30.

[36] Flora Sun. UTXO vs Account/Balance Model. `https://medium.com/@sunflora98/utxo-vs-account-balance-model-5e6470f4e0cf`, October 2018. Accessed on 2019-11-30.

[37] Prasanna. Blockchain Trilemma: Explained. `https://cryptoticker.io/en/blockchain-trilemma-explained/`, November 2019. Accessed on 2019-11-02.

[38] Andreas M. Antonopoulos. 6. The Bitcoin Network - Mastering Bitcoin [Book]. https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch06.html, December 2014. Accessed on 2019-12-13.

[39] blockchain.com. Hashrate Verteilung. https://www.blockchain.com/pools, December 2019. Accessed on 2019-12-06.

[40] Binance.com. Staking Pool - Definition. https://www.binance.vision/glossary/staking-pool, December 2019. Accessed on 2019-12-06.

[41] Adah. Types of wallets. http://support.coinmama.com/hc/en-us/articles/360015098654-Types-of-wallets, October 2019. Accessed on 2019-10-25.

[42] Nathan Reiff. 20% of All BTC is Lost, Unrecoverable, Study Shows. https://www.investopedia.com/news/20-all-btc-lost-unrecoverable-study-shows/, June 2019. Accessed on 2019-12-07.

[43] Bitcoin Core Developer. bitcoin/bips. https://github.com/bitcoin/bips, November 2019. Accessed on 2019-11-02.

[44] Narayan Prusty and Paul Valencourt and Samanyu Chopra and Brenn Hill. Blockchain developer's guide. https://www.oreilly.com/library/view/blockchain-developers-guide/9781789954722/, December 2018. Accessed on 2019-10-26.

[45] BitDegree. Token vs Coin: What's the Difference? https://www.bitdegree.org/tutorials/token-vs-coin/, March 2018. Accessed on 2019-12-08.

[46] Fabian Vogelsteller and Vitalik Buterin. EIP-20: ERC-20 Token Standard. https://eips.ethereum.org/EIPS/eip-20, November 2015. Accessed on 2019-12-08.

[47] William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs. EIP-721: ERC-721 Non-Fungible Token Standard. https://eips.ethereum.org/EIPS/eip-721, January 2018. Accessed on 2019-12-08.

[48] Ethereum Foundation. Etherscan ERC-721 Token Tracker Page. https://etherscan.io/tokens-nft, December 2019. Accessed on 2019-12-08.

[49] Tether. Tether. https://tether.to/, December 2019. Accessed on 2019-12-08.

[50] Ethereum Foundation. Etherscan Token Tracker. https://etherscan.io/tokens, December 2019. Accessed on 2019-12-08.

[51] https://gu.cards/. Gods Unchained Cards | Gods Unchained Cards. https://gu.cards/, December 2019. Accessed on 2019-12-08.

[52] Coin Crunch. Guide to Crypto Token Types. https://hackernoon.com/guide-to-crypto-token-types-6ce04edaba72, July 2018. Accessed on 2019-12-14.

[53] S. Voshmgir. *Token Economy: How Blockchains and Smart Contracts Revolutionize the Economy.* Shermin Voshmgir, 2019.

[54] Philipp Sandner, Thomas Nägele, and Jonas Gross. Liechtenstein Blockchain Act: How can nearly any right and therefore any asset be tokenized based on the Token Container Model? https://medium.com/@philippsandner/liechtenstein-blockchain-act-how-can-nearly-any-right-and-therefore-any-asset-be-tokenized-based-389fc9f039b1, October 2019. Accessed on 2019-10-27.

[55] en.bitcoin.it. Atomic swap - Bitcoin Wiki. https://en.bitcoin.it/wiki/Atomic_swap, December 2019. Accessed on 2019-12-08.

[56] Blockgeeks. Learn What Are Bitcoin Forks? [The Ultimate Step-by-Step Guide]. https://blockgeeks.com/guides/bitcoin-forks-guide/, May 2018. Accessed on 2019-10-25.

[57] Lilon UG. Blockchain Oracles. https://blockchainhub.net/blockchain-oracles/, October 2019. Accessed on 2019-10-27.

[58] Julien Thevenard. Decentralised Oracles: a comprehensive overview. https://medium.com/fabric-ventures/decentralised-oracles-a-comprehensive-overview-d3168b9a8841, June 2019. Accessed on 2019-10-27.

[59] SmartContract Chainlink Ltd. Chainlink. https://chain.link/, October 2019. Accessed on 2019-10-27.

[60] Lei ZHANG. Intel® SGX and Blockchain: The iExec End-to-End Trusted Execution Solution. https://medium.com/iex-ec/iexec-end-to-end-sgx-solution-fee1e63297b2, February 2019. Accessed on 2019-11-02.

[61] Joshua Lind and Ittay Eyal and Florian Kelbert and Oded Naor and Peter Pietzuch and Emin Gun Sirer. Teechain: Scalable blockchain payments using trusted execution environments. https://www.researchgate.net/profile/Florian_Kelbert/publication/318528079_Teechain_Scalable_Blockchain_Payments_using_Trusted_Execution_Environments/links/5970a7d4aca272a59f76c112/Teechain-Scalable-Blockchain-Payments-using-Trusted-Execution-Environments.pdf, July 2017. Accessed on 2019-10-26.

[62] Andreas M. Antonopoulos. 7. The Blockchain - Mastering Bitcoin [Book]. https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch07.html, December 2014. Accessed on 2019-10-25.

[63] Bitcoin.org. Transactions Guide - Bitcoin. https://bitcoin.org/en/transactions-guide#introduction, October 2019. Accessed on 2019-10-25.

[64] Andreas M. Antonopoulos. 5. Transactions - Mastering Bitcoin [Book]. https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch05.html, December 2014. Accessed on 2019-10-25.

[65] programmingblockchain.gitbook.io. P2WPKH (Pay to Witness Public Key Hash). https://programmingblockchain.gitbook.io/programmingblockchain/other_types_of_ownership/p2wpkh_pay_to_witness_public_key_hash, October 2019. Accessed on 2019-10-25.

[66] programmingblockchain.gitbook.io. P2WSH (Pay to Witness Script Hash). https://programmingblockchain.gitbook.io/programmingblockchain/other_types_of_ownership/p2wsh_pay_to_witness_script_hash, October 2019. Accessed on 2019-10-25.

[67] en.bitcoin.it. Script. https://en.bitcoin.it/wiki/Script, October 2019. Accessed on 2019-10-26.

[68] Ethereum Foundation. Solidity — Solidity 0.5.12 documentation. https://solidity.readthedocs.io/en/v0.5.12/, October 2019. Accessed on 2019-10-27.

[69] Andreas M. Antonopoulos. ethereumbook/ethereumbook. https://github.com/ethereumbook/ethereumbook, November 2018. Accessed on 2019-11-16.

[70] Nick Szabo. Formalizing and Securing Relationships on Public Networks. *First Monday*, 2(9), September 1997. Accessed on 2019-10-27.

[71] Nick Szabo. Bit Gold | Satoshi Nakamoto Institute. https://nakamotoinstitute.org/bit-gold/, December 2005. Accessed on 2019-10-27.

[72] Pavel Romanenko. 20 Blockchain Use Cases for 2018 You Should Know. https://hackernoon.com/20-blockchain-use-cases-for-2018-you-should-know-f7d2919c191d, October 2019. Accessed on 2019-10-27.

[73] Jordan Clifford. Scriptless Scripts. https://medium.com/scalar-capital/scriptless-scripts-25e18fd52ede, April 2019. Accessed on 2019-11-09.

[74] Nolan Bauerle. What are Blockchain's Issues and Limitations? https://www.coindesk.com/information/blockchains-issues-limitations. Accessed on 2019-10-04.

[75] Digiconomist. Bitcoin Energy Consumption Index. https://digiconomist.net/bitcoin-energy-consumption, October 2019. Accessed on 2019-10-12.

[76] BLOCKCHAIN LUXEMBOURG S.A. Average Number Of Transactions Per Block. https://www.blockchain.com/charts/n-transactions-per-block, October 2019. Accessed on 2019-10-12.

[77] Buy Bitcoin Worldwide. 3 Things to Know about Bitcoin Confirmations (2019 Updated). https://www.buybitcoinworldwide.com/confirmations/, October 2019. Accessed on 2019-10-13.

[78] Muhammad Saad, Jeffrey Spaulding, Laurent Njilla, Charles Kamhoua, Sachin Shetty, DaeHun Nyang, and Aziz Mohaisen. Exploring the Attack Surface of Blockchain: A Systematic Overview. *arXiv:1904.03487 [cs]*, April 2019. Accessed on 2019-10-19.

[79] LearnCryptography.com. Learn Cryptography - 51% Attack. https://learncryptography.com/cryptocurrency/51-attack, October 2019. Accessed on 2019-10-19.

[80] Ittay Eyal and Emin Gun Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable. *ACM*, page 18, June 2018.

[81] Blockchain Luxembourg S.A. Blockchain Size. https://blockchain.info/blocks-size, May 2018. Accessed on 2018-05-07.

[82] Bitcoinfees.info. Bitcoin Transaction Fees. https://bitcoinfees.info/, October 2019. Accessed on 2019-10-13.

[83] Kim P. Credit Card Processing Fees. https://www.creditdonkey.com/credit-card-processing-fees.html, October 2019. Accessed on 2019-10-13.

[84] Michael Borkowski, Philipp Frauenthaler, Marten Sigwart, Taneli Hukkinen, Oskar Hladky, and Stefan Schulte. Cross-Blockchain Technologies: Review, State of the Art, and Outlook. *borkowski.at*, page 5, November 2019. Accessed on 2019-11-15.

[85] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of Clients in Bitcoin P2P Network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*, pages 15–29, Scottsdale, Arizona, USA, 2014. ACM Press. Accessed on 2019-10-26.

[86] en.bitcoin.it. Privacy. https://en.bitcoin.it/wiki/Privacy, October 2019. Accessed on 2019-10-26.

[87] en.bitcoin.it. CoinJoin - Bitcoin Wiki. https://en.bitcoin.it/wiki/CoinJoin, November 2019. Accessed on 2019-11-23.

[88] en.bitcoin.it. Confidential transactions - Bitcoin Wiki. `https://en.bitcoin.it/wiki/Confidential_transactions`, November 2019. Accessed on 2019-11-23.

[89] Amitabh Saxena, Janardan Misra, and Aritra Dhar. Increasing Anonymity in Bitcoin. In *Challenges and Opportunities Associated with a Bitcoin-Based Transaction Rating System*, March 2014.

[90] Xinxin Fan. Faster Dual-Key Stealth Address for Blockchain-Based Internet of Things Systems. *arXiv:1806.00951 [cs]*, June 2018. Accessed on 2019-11-21.

[91] Tom Elvis Jedusor. mimblewimble/docs. `https://github.com/mimblewimble/docs`, July 2016. Accessed on 2019-11-23.

[92] Kalpana Singh, Nicolas Heulot, and Elyes Ben Hamida. Towards Anonymous, Unlinkable, and Confidential Transactions in Blockchain. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1642–1649, July 2018. ISSN: null.

[93] Yuki Yasusaka, Chiemi Watanabe, and Hiroyuki Kitagawa. Privacy-Preserving Pre-Consensus Protocol for Blockchains. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–8, February 2019. ISSN: 2375-933X.

[94] Aniket Kate, Yizhou Huang, and Ian Goldberg. Distributed Key Generation in the Wild. Technical Report 377, Cryptology ePrint Archive, Report 2012/377, 2012. Accessed on 2019-11-02.

[95] Torus. TORUS | Home. `https://tor.us/`, November 2019. Accessed on 2019-11-02.

[96] Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems. Technical Report 134, eprint.iacr.org, 2002. Accessed on 2019-11-29.

[97] Aaron van Wirdum. The Next Step to Improve Bitcoin's Flexibility, Scalability and... `https://bitcoinmagazine.com/articles/the-next-step-to-improve-bitcoin-s-flexibility-scalability-and-privacy-is-called-mast-1476388597`, October 2016. Accessed on 2019-12-21.

[98] Johnson Lau. bitcoin/bips. `https://github.com/bitcoin/bips`, April 2016. Accessed on 2019-12-21.

[99] Aaron van Wirdum. Taproot Is Coming: What It Is, and How It Will Benefit Bitcoin. `https://bitcoinmagazine.com/articles/taproot-coming-`

what-it-and-how-it-will-benefit-bitcoin, January 2019. Accessed on 2019-12-21.

[100] Clavestone. Bitcoin: Multisig vs. Shamir's Secret Sharing Scheme. https://medium.com/clavestone/bitcoin-multisig-vs-shamirs-secret-sharing-scheme-ea83a888f033, March 2019. Accessed on 2019-11-29.

[101] Lucas Nuzzi. Schnorr Signatures & The Inevitability of Privacy in Bitcoin. https://medium.com/digitalassetresearch/schnorr-signatures-the-inevitability-of-privacy-in-bitcoin-b2f45a1f7287, March 2019. Accessed on 2019-11-08.

[102] Bill Buchanan. Ring Signatures And Anonymisation. https://medium.com/coinmonks/ring-signatures-and-anonymisation-c9640f08a193, December 2018. Accessed on 2019-10-26.

[103] Hcash. Ring Signatures, Blockchain Security, and Privacy Protection Technologies. https://medium.com/@media_30378/ring-signatures-blockchain-security-and-privacy-protection-technologies-50c7b90db30e, February 2019. Accessed on 2019-12-08.

[104] Monero Research Lab. Monero: Home. https://getmonero.org/index.html, October 2019. Accessed on 2019-10-26.

[105] Coincentral. What is Monero? An In-depth Guide. https://hackernoon.com/what-is-monero-an-in-depth-guide-5d43f1917178, October 2019. Accessed on 2019-10-26.

[106] ELECTRIC COIN COMPANY. Privacy-protecting digital currency. https://z.cash/, October 2019. Accessed on 2019-10-26.

[107] ELECTRIC COIN COMPANY. What are zk-SNARKs? https://z.cash/technology/zksnarks/, October 2019. Accessed on 2019-10-27.

[108] ELECTRIC COIN COMPANY. How It Works. https://z.cash/technology/, October 2019. Accessed on 2019-10-26.

[109] grin.mw. Grin. https://grin.mw/, November 2019. Accessed on 2019-11-23.

[110] Beam Development Limited. BEAM-Mimblewimble-based Privacy Coin. https://beam.mw/, November 2019. Accessed on 2019-11-23.

[111] Tom Elvis Jedusor. mimblewimble/grin. https://github.com/mimblewimble/grin, April 2019. Accessed on 2019-11-23.

[112] Beam Development Limited. BEAM-Beampedia/Equihash. https://beam.mw/beampedia-item/equihash, November 2019. Accessed on 2019-11-23.

[113] Zhangshuang Guan, Zhiguo Wan, Yang Yang, Yan Zhou, and Butian Huang. BlockMaze: An Efficient Privacy-Preserving Account-Model Blockchain Based on zk-SNARKs. Technical Report 1354, eprint.iacr.org, 2019. Accessed on 2019-11-30.

[114] en.bitcoin.it. Block size limit controversy - Bitcoin Wiki. https://en.bitcoin.it/wiki/Block_size_limit_controversy, November 2019. Accessed on 2019-11-02.

[115] TradeBlock. Analysis of Bitcoin Transaction Size Trends. https://tradeblock.com/blog/analysis-of-bitcoin-transaction-size-trends, October 2015. Accessed on 2019-11-02.

[116] Nikolai Kuznetsov. SegWit, Explained. https://cointelegraph.com/explained/segwit-explained, September 2019. Accessed on 2019-11-02.

[117] Jimmy Song. Understanding Segwit Block Size. https://medium.com/@jimmysong/understanding-segwit-block-size-fd901b87c9d4, August 2017. Accessed on 2019-11-02.

[118] Tiaan Wolmarans. Sharding and Scaling On Blockchain. https://hackernoon.com/sharding-and-the-scaling-of-a-blockchain-xz1kq30j0, August 2019. Accessed on 2019-11-15.

[119] Juri Mattila. THE BLOCKCHAIN PHENOMENON. *ResearchGate*, page 26, May 2016. Accessed on 2019-10-20.

[120] Pool Of Stake. Proof of Stake and Scalability. https://medium.com/@poolofstake/proof-of-stake-and-scalability-2ecaa8fd5d7c, April 2018. Accessed on 2019-12-06.

[121] Chaya Ganesh, Claudio Orlandi, and Daniel Tschudi. Proof-of-Stake Protocols for Privacy-Aware Blockchains. Technical Report 1105, eprint.iacr.org, 2018. Accessed on 2019-12-06.

[122] Thomas Kerber, Markulf Kohlweiss, Aggelos Kiayias, and Vassilis Zikas. Ouroboros Crypsinous: Privacy-Preserving Proof-of-Stake. Technical Report 1132, eprint.iacr.org, 2018. Accessed on 2019-12-06.

[123] Jelurida Swiss SA. Nxt | Jelurida. https://www.jelurida.com/nxt, October 2019. Accessed on 2019-10-20.

[124] Cardano Foundation. Home. https://www.cardano.org/en/home/, November 2019. Accessed on 2019-11-21.

[125] Cardano Foundation. Ouroboros Proof of Stake Algorithm. https://www.cardano.org/en/ouroboros/, November 2019. Accessed on 2019-11-21.

[126] Ethereum Foundation. Ethereum 2.0 (Serenity) Phases - EthHub. https://docs.ethhub.io/ethereum-roadmap/ethereum-2.0/eth-2.0-phases/, October 2019. Accessed on 2019-10-20.

[127] Block.one. EOSIO - Blockchain software architecture. https://eos.io/, July 2019. Accessed on 2019-07-08.

[128] Apla. Proof-of-Authority consensus — Apla Blockchain Platform Guide documentation. https://apla.readthedocs.io/en/latest/concepts/consensus.html, October 2019. Accessed on 2019-10-20.

[129] GoChainGo. Proof of Reputation. https://medium.com/gochain/proof-of-reputation-e37432420712, February 2018. Accessed on 2019-10-20.

[130] Vishwas B. Anand. Proof-Of-Energy (PoE). https://medium.com/experiencebihar/proof-of-energy-poe-32374e05f213, December 2018. Accessed on 2019-10-20.

[131] Peter Holzer. PeHo89/PoREP. https://github.com/PeHo89/PoREP, October 2019. Accessed on 2019-10-25.

[132] Jake Frankenfield. Proof of Elapsed Time (Cryptocurrency). https://www.investopedia.com/terms/p/proof-elapsed-time-cryptocurrency.asp, October 2019. Accessed on 2019-10-20.

[133] Todorin Cryptocurrency • 2 Years Ago. Proof-of-play. https://steemit.com/cryptocurrency/@todor/proof-of-play, July 2017. Accessed on 2019-10-20.

[134] Coordicide Team - IOTA Foundation. The coordicide. https://files.iota.org/papers/Coordicide_WP.pdf, October 2019. Accessed on 2019-10-25.

[135] Serguei Popov and William J. Buchanan. FPC-BI: Fast Probabilistic Consensus within Byzantine Infrastructures. *arXiv:1905.10895 [cs, math]*, July 2019. Accessed on 2019-10-25.

[136] Sheffield Nolan. pBFT— Understanding the Consensus Algorithm. https://medium.com/coinmonks/pbft-understanding-the-algorithm-b7a7869650ae, November 2018. Accessed on 2019-11-02.

[137] Parikshit Hooda. practical Byzantine Fault Tolerance(pBFT). https://www.geeksforgeeks.org/practical-byzantine-fault-tolerancepbft/, January 2019. Accessed on 2019-11-02.

[138] Libra Association. Libra-Whitepaper. https://libra.org/de-DE/white-paper/, November 2019. Accessed on 2019-11-02.

[139] The Ontology Team. HotStuff: the Consensus Protocol Behind Facebook's LibraBFT. https://medium.com/ontologynetwork/hotstuff-the-consensus-protocol-behind-facebooks-librabft-a5503680b151, September 2019. Accessed on 2019-11-02.

[140] IOTA Foundation. The Next Generation of Distributed Ledger Technology. https://www.iota.org/. Accessed on 2019-07-03.

[141] Serguei Popov. The tangle. https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf, October 2019. Accessed on 2019-10-18.

[142] Leslie Ankney. IOTA: No transaction fees and infinite scalability — How does it work? https://medium.com/@CryptoLeslie/iota-no-transaction-fees-and-infinite-scalability-how-does-it-work-b0107a9f988d, September 2017. Accessed on 2019-10-19.

[143] Alon Gal. The Tangle: an Illustrated Introduction. https://blog.iota.org/the-tangle-an-illustrated-introduction-4d5eae6fe8d4, April 2018. Accessed on 2019-10-18.

[144] Stephen O'Neal. Who Scales It Best? Inside Blockchains' Ongoing Transactions-Per-Second Race. https://cointelegraph.com/news/who-scales-it-best-inside-blockchains-ongoing-transactions-per-second-race, January 2019. Accessed on 2019-10-18.

[145] IOTA Foundation. The Coordicide. Realizing IOTA's vision of a permissionless and scalable distributed ledger technology. https://coordicide.iota.org, October 2019. Accessed on 2019-10-19.

[146] Hedera Hashgraph. Hello future. http://www.hedera.com, October 2019. Accessed on 2019-10-18.

[147] Nano Foundation. Nano | Digital money for the modern world. https://nano.org, November 2019. Accessed on 2019-11-15.

[148] Locus Chain Foundation. Locus Chain. https://locuschain.com/en/index, December 2019. Accessed on 2019-12-06.

[149] James Aspnes and Gauri Shah. Skip graphs. *ACM Transactions on Algorithms*, 3(4):37, November 2007.

[150] Holo Ltd. Holochain. https://holochain.org/, October 2019. Accessed on 2019-10-19.

[151] Holo Ltd. How does Holochain manage consensus and data integrity? | Holo FAQ. https://holo.host/faq/how-does-holochain-manage-consensus-data-integrity/, October 2019. Accessed on 2019-10-20.

[152] Holo Ltd. Holochain-Core-Concepts - Holochain Docs. https://developer.holochain.org/docs/concepts/, October 2019. Accessed on 2019-10-19.

[153] Web 3.0 Technologies Foundation. Polkadot-Network. https://polkadot.network/, November 2019. Accessed on 2019-11-15.

[154] Web 3.0 Foundation. Introducing polkadot. https://polkadot.network/Polkadot-lightpaper.pdf, May 2019. Accessed on 2019-05-15.

[155] Linux Foundation. Hyperledger Fabric. https://www.hyperledger.org/projects/fabric, November 2019. Accessed on 2019-11-29.

[156] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *Proceedings of the Thirteenth EuroSys Conference on - EuroSys '18*, pages 1–15, 2018. Accessed on 2019-11-29.

[157] Linux Foundation. ReadTheDocs-Hyperledger. https://hyperledger-fabric.readthedocs.io/en/release-1.4/, November 2019. Accessed on 2019-11-29.

[158] Ethereum Foundation. Raiden Network. https://raiden.network/, October 2019. Accessed on 2019-10-18.

[159] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual Payment Hubs over Cryptocurrencies. Technical Report 635, https://eprint.iacr.org/, 2017. Accessed on 2019-11-15.

[160] Rami Khalil, Alexei Zamyatin, Guillaume Felley, Pedro Moreno-Sanchez, and Arthur Gervais. Commit-Chains: Secure, Scalable Off-Chain Payments. Technical Report 642, eprint.iacr.org, 2018. Accessed on 2019-11-15.

[161] Georgios Konstantopoulos. loomnetwork/plasma-paper. https://github.com/loomnetwork/plasma-paper, January 2019. Accessed on 2019-11-15.

[162] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling Blockchain Innovations with Pegged Sidechains. *https://blockstream.com/*, page 25, October 2014. Accessed on 2019-11-15.

[163] Katalyse.io. What are Sidechains and Childchains? https://hackernoon.com/what-are-sidechains-and-childchains-7202cc9e5994, August 2018. Accessed on 2019-12-08.

[164] RSK Labs. RSK-Home. https://rsk.co, December 2019. Accessed on 2019-12-08.

[165] RSK Labs. RSK-Developers-Portal. https://developers.rsk.co/, December 2019. Accessed on 2019-12-08.

[166] Blockstream Corporation Inc. Liquid Network FAQs — documentation. https://docs.blockstream.com/liquid/faq.html, December 2019. Accessed on 2019-12-13.

[167] Blockstream Corporation Inc. Liquid. https://blockstream.com/liquid/, December 2019. Accessed on 2019-12-13.

[168] Blockstream Corporation Inc. Technical Overview — documentation. https://docs.blockstream.com/liquid/technical_overview.html, December 2019. Accessed on 2019-12-13.

[169] Jacob Blacklock and Shi Lei. Blockchain Laws and Regulations | China | GLI. https://www.globallegalinsights.com/, December 2019. Accessed on 2019-12-07.

[170] Ahmed Banafa. Quantum Computing vs Blockchain Cryptography - Facts, Myths, and Synergies. https://hackernoon.com/quantum-computing-and-blockchain-facts-and-myths-l71w28d2, December 2019. Accessed on 2019-12-07.

[171] en.bitcoin.it. Quantum computing and Bitcoin - Bitcoin Wiki. https://en.bitcoin.it/wiki/Quantum_computing_and_Bitcoin, December 2019. Accessed on 2019-12-07.

[172] Edmund-Philipp Schuster. Cloud Crypto Land. SSRN Scholarly Paper ID 3476678, Social Science Research Network, Rochester, NY, November 2018. Accessed on 2019-11-28.

[173] Kate Rooney. Bitcoin is the 'mother of all scams' and blockchain is most hyped tech ever, Roubini tells Congress. https://www.cnbc.com/2018/10/11/roubini-bitcoin-is-mother-of-all-scams.html, October 2018. Accessed on 2019-12-13.

[174] Yun Li. Warren Buffett says bitcoin is a 'gambling device' with 'a lot of frauds connected with it'. https://www.cnbc.com/2019/05/04/warren-buffett-says-bitcoin-is-a-gambling-device-with-a-lot-of-frauds-connected-with-it.html, May 2019. Accessed on 2019-12-13.

[175] Omar Wagih. Outsourcing Route Computation With Trampoline Payments. https://bitcointechweekly.com/front/outsourcing-route-computation-with-trampoline-payments/, April 2019. Accessed on 2019-11-10.

[176] en.bitcoin.it. Hash Time Locked Contracts - Bitcoin Wiki. https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts, November 2019. Accessed on 2019-11-02.

[177] Dennis Reimann. The Payment Channel Lifecycle – Lightning Network explained. https://dennisreimann.de/articles/lightning-network-payment-channel-lifecycle.html, May 2019. Accessed on 2019-11-02.

[178] lightningnetwork/lnd. https://github.com/lightningnetwork/lnd, October 2019. Accessed on 2019-10-04.

[179] Cezary Dziemian. [Lightning-dev] Both-side funded channels. https://www.mail-archive.com/lightning-dev@lists.linuxfoundation.org/msg00668.html, October 2018. Accessed on 2019-12-13.

[180] Dennis Reimann. How channel capacity and liquidity affect payment routing – Lightning Network explained. https://dennisreimann.de/articles/lightning-network-routing-payments-liquidity.html, May 2019. Accessed on 2019-11-03.

[181] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, Giulia Fanti, and Pramod Viswanath. Routing Cryptocurrency with the Spider Network. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks - HotNets '18*, pages 29–35, Redmond, WA, USA, 2018. ACM Press. Accessed on 2019-11-16.

[182] Ben Congdon. bcongdon/awesome-lightning-network. https://github.com/bcongdon/awesome-lightning-network, October 2019. Accessed on 2019-11-02.

[183] ACINQ SAS. ACINQ/eclair. https://github.com/ACINQ/eclair, October 2019. Accessed on 2019-11-02.

[184] The MIT Digital Currency Initiative. mit-dci/lit. https://github.com/mit-dci/lit, October 2019. Accessed on 2019-11-02.

[185] Elements by Blockstream. ElementsProject/lightning. https://github.com/ElementsProject/lightning, November 2019. Accessed on 2019-11-02.

[186] Rust Bitcoin Community. rust-bitcoin/rust-lightning. https://github.com/rust-bitcoin/rust-lightning, October 2019. Accessed on 2019-11-02.

[187] Lightning Labs, Inc. lightningnetwork/lightning-onion. https://github.com/lightningnetwork/lightning-onion, October 2019. Accessed on 2019-11-02.

[188] Nayuta. nayutaco/ptarmigan. https://github.com/nayutaco/ptarmigan, November 2019. Accessed on 2019-11-02.

[189] BitcoinVisuals.com. Bitcoin Visuals - Extensive Charts and Statistics. https://bitcoinvisuals.com/, November 2019. Accessed on 2019-11-10.

[190] Kai Sedgwick. Lightning Network User Confused by Protocol Lost 30,000 USD. https://news.bitcoin.com/mishap-sees-user-lose-30000-btc-on-lightning-network/, October 2019. Accessed on 2019-12-14.

[191] Weizhao Tang, Weina Wang, Giulia Fanti, and Sewoong Oh. Privacy-Utility Trade-offs in Routing Cryptocurrency over Payment Channel Networks. *arXiv:1909.02717 [cs]*, September 2019. Accessed on 2019-11-16.

[192] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and Privacy with Payment-Channel Networks. Technical Report 820, Cryptology ePrint Archive, Report 2017/820, 2017. Accessed on 2019-11-03.

[193] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability. In *Proceedings 2019 Network and Distributed System Security Symposium*, San Diego, CA, 2019. Internet Society. Accessed on 2019-10-27.

[194] Matteo Maffei. Security and privacy for payment channel networks - lecture series in distributed ledgers and smart contracts - darmstadt, april 5th 2019. https://secpriv.tuwien.ac.at/fileadmin/t/secpriv/Slides/Darmstadt-Blockchain-2019.pdf, April 2019. Accessed on 2019-11-03.

[195] Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, Alejandro Ranchal-Pedrosa, Cristina Pérez-Solà, and Joaquin Garcia-Alfaro. On the Difficulty of Hiding the Balance of Lightning Network Channels. Technical Report 328, eprint.iacr.org, 2019. Accessed on 2019-11-10.

[196] Cristina Pérez-Solà, Alejandro Ranchal-Pedrosa, Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, and Joaquin Garcia-Alfaro. LockDown: Balance Availability Attack against Lightning Network Channels. Technical Report 1149, eprint.iacr.org, 2019. Accessed on 2019-11-10.

[197] EOS Congested Says Coinbase, Meet AVA Ethereum Fork to Give Free ATH, "We Need Another Two Weeks" For Ethereum 2 0 Testnet Says Dev, Press Releases, Advertising, News Tips, and Positions. Lightning Network DDoS Sends 20% of Nodes Down. https://www.trustnodes.com/2018/03/21/lightning-network-ddos-sends-20-nodes, March 2018. Accessed on 2019-11-10.

[198] István András Seres, László Gulyás, Dániel A. Nagy, and Péter Burcsi. Topological Analysis of Bitcoin's Lightning Network. *arXiv:1901.04972 [cs]*, April 2019. Accessed on 2019-11-10.

[199] Olaoluwa Osuntokun. [Lightning-dev] AMP: Atomic Multi-Path Payments over Lightning. `https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-February/000993.html`, February 2018. Accessed on 2019-11-09.

[200] Dmytro Piatkivskyi and Mariusz Nowostawski. Split Payments in Payment Networks. In Joaquin Garcia-Alfaro, Jordi Herrera-Joancomartí, Giovanni Livraga, and Ruben Rios, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, volume 11025, pages 67–75. Springer International Publishing, Cham, 2018. Accessed on 2019-11-09.

[201] SEBASTIÁN RECA. Rebalancing: The Key to the Lightning Network. `https://blog.muun.com/rebalancing-in-the-lightning-network/`, October 2018. Accessed on 2019-11-09.

[202] FLORENCIA RAVENNA. Splices and Liquidity in the Lightning Network. `https://blog.muun.com/splices-and-liquidity-in-the-lightning-network/`, July 2019. Accessed on 2019-11-09.

[203] blockonomi.com. What Are Watchtowers in Bitcoin's Lightning Network? `https://blockonomi.com/watchtowers-bitcoin-lightning-network/`, February 2019. Accessed on 2019-11-10.

[204] Breez. breez/LightningRod. `https://github.com/breez/LightningRod`, November 2019. Accessed on 2019-11-10.

[205] Roy Sheinfeld. Introducing Lightning Rod. `https://medium.com/breez-technology/introducing-lightning-rod-2e0a40d3e44a`, October 2019. Accessed on 2019-11-10.

[206] Chen Pan, Shuyang Tang, Zhiqiang Liu, Yu Long, Zhen Liu, and Dawu Gu. Gnocchi : A Multiplexed Payment Channel Scheme. In *Gnocchi: A Multiplexed Payment Channel Scheme*, 2018.

[207] Ferenc Beres, Istvan Andras Seres, and Andras A. Benczur. A Cryptoeconomic Traffic Analysis of Bitcoins Lightning Network. *arXiv:1911.09432 [cs]*, November 2019. Accessed on 2019-11-28.

[208] Mariusz Nowostawski and Jardar Tøn. Evaluating Methods for the Identification of Off-Chain Transactions in the Lightning Network. *Applied Sciences*, 9(12):2519, June 2019. Accessed on 2019-10-04.

[209] Avi Mizrahi. Report: Lightning Network Still Way off Being Ready for Commercial Use. https://news.bitcoin.com/report-lightning-network-still-way-off-being-ready-for-commercial-use/, December 2018. Accessed on 2019-12-14.

[210] Jamie Redman. Business Owner's Seething Critique of the Lightning Network Goes Viral. https://news.bitcoin.com/business-owners-seething-critique-of-the-lightning-network-goes-viral/, March 2019. Accessed on 2019-12-14.

[211] Harry Kalodner, Steven Goldfeder, Alishah Chator, Malte Möser, and Arvind Narayanan. BlockSci: Design and applications of a blockchain analysis platform. *arXiv:1709.02489 [cs]*, September 2017. Accessed on 2019-12-14.

[212] Princeton CITP. citp/BlockSci. https://github.com/citp/BlockSci, December 2019. Accessed on 2019-12-14.

[213] sqlite.org. SQLite Version 3 Overview. https://www.sqlite.org/version3.html, February 2020. Accessed on 2020-02-07.

[214] Austrian Institute of Technology GmbH. GraphSense. https://github.com/graphsense. Accessed on 2019-10-11.

[215] en.bitcoin.it. Common-input-ownership heuristic - Bitcoin Wiki. https://en.bitcoin.it/wiki/Common-input-ownership_heuristic, March 2020. Accessed on 2020-03-18.

[216] Lightning Labs, Inc. lightningnetwork/lnd. https://github.com/lightningnetwork/lnd/releases/tag/v0.4-beta, March 2018. Accessed on 2020-02-07.

[217] BitMEX Research. Lightning Network (Part 7) – Proportion Of Public vs Private Channels | BitMEX Blog. https://blog.bitmex.com/lightning-network-part-7-proportion-of-public-vs-private-channels/, January 2020. Accessed on 2020-02-08.

[218] apilayer.com. ipstack - Free IP Geolocation API. https://ipstack.com/, February 2020. Accessed on 2020-02-22.

[219] BlueWallet Services S.R.L. Lightning Wallet - Bluewallet Bitcoin wallet for iOS and Android. https://bluewallet.io/lightning/, February 2020. Accessed on 2020-02-10.

[220] Jian-Hong Lin, Kevin Primicerio, Tiziano Squartini, Christian Decker, and Claudio J. Tessone. Lightning Network: a second path towards centralisation of the Bitcoin economy. *arXiv:2002.02819 [physics]*, February 2020. Accessed on 2020-02-22.

[221] LNBIG.com. LNBIG - The Network of LND Servers With Greater Liquidity. https://lnbig.com/#/our-nodes, February 2020. Accessed on 2020-02-29.

[222] LNBIG.com. LNBIG - The Network of LND Servers With Greater Liquidity. https://lnbig.com/#/, February 2020. Accessed on 2020-02-29.