Incentives and Cryptographic Protocols for Bitcoin-like Blockchains

Ersoy, O.

DOI
10.4233/uuid:d467c061-ef88-4b70-aaf2-5233923282eb

Publication date
2021

Document Version
Final published version

Citation (APA)
Ersoy, O. (2021). *Incentives and Cryptographic Protocols for Bitcoin-like Blockchains*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:d467c061-ef88-4b70-aaf2-5233923282eb

Incentives and Cryptographic Protocols for Bitcoin-like Blockchains

INCENTIVES AND CRYPTOGRAPHIC PROTOCOLS FOR BITCOIN-LIKE BLOCKCHAINS

Dissertation

for the purpose obtaining the degree of doctor
at Delft University of Technology
by the authority of Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen
chair of the Board for Doctorates
to be defended by publicly on
Tuesday 7 September 2021 at 12.30 o'clock

by

Oğuzhan ERSOY

Master of Science in Electrical & Electronics Engineering,
Boğaziçi University, Turkey,
born in Erzurum, Turkey.

Keywords: Blockchain, Bitcoin, payment channel networks.
Printed by: Gildeprint
Cover: Oğuzhan Ersoy

ISBN 978-94-6384-248-8

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

To my family

CONTENTS

SUMMARY

Bitcoin is a widely acknowledged digital currency that is designed in a decentralized manner. The recognition of Bitcoin has introduced the notion of cryptocurrencies and, in general, blockchain technology. Blockchain, within less than a decade, has become one of the most exciting technological developments. Among several exciting use cases and projects, there has been an inevitable hype in the industry as well. While in the research community, it has opened an interdisciplinary research field among cryptography, distributed systems, and economics.

Notwithstanding the interest and great effort, blockchain is still a new and evolving technology, and numerous challenges need to be addressed. To name a few, security, privacy, scalability, smart contracts, and economic aspects with their manifold sub-challenges can be mentioned. Among the research challenges, in this thesis, we investigate three crucial ones for the long-term functionality of the Bitcoin-like blockchains, which are security, scalability, and economic aspects. Our works can be divided into two subjects: transaction propagation and payment channel networks.

Transaction propagation or advertisement refers to the dissemination of newly created transactions of clients in the mining network. In this thesis, we investigate the lack of incentives for transaction propagation and provide an incentive mechanism for peer-to-peer mining networks. Moreover, we focus on the inefficient routing of the transactions and propose a smart routing mechanism.

Payment channel networks (PCN) are promising layer-2 protocols aiming to improve the scalability of blockchains. In this thesis, we present three works on the PCNs. Firstly, we investigate the incentives to participate in multi-hop payments and propose a profit strategy that would encourage the use of PCNs. Secondly, we propose the first Bitcoin-compatible virtual channel constructions on payment channels that improve the efficiency and availability of multi-hop payments. Finally, we introduce the first post-quantum PCN utilizing our post-quantum adaptor signature scheme. Our works mainly focus on Bitcoin and its PCN, Lightning Network, yet they can be applied to the blockchains and cryptocurrencies having similar characteristics.

SAMENVATTING

Bitcoin is een algemeen erkende digitale valuta die op een gedecentraliseerde manier is ontworpen. Met de komst van Bitcoin is het concept van cryptocurrencies en, in het algemeen, blockchain-technologie geïntroduceerd. Blockchain is in minder dan een decennium een van de meest opwindende technologische ontwikkelingen geworden. Naast een aantal interessante toepassingen en projecten, is er ook een onvermijdelijke hype in de industrie ontstaan. In de wetenschap heeft het een interdisciplinair onderzoeksgebied geopend tussen cryptografie, gedistribueerde systemen en economie.

Ondanks de interesse en grote inspanningen staat blockchain nog steeds in zijn kinderschoenen en dienen er nog vele uitdagingen overwonnen te worden. Voorbeelden hiervan zijn op gebied van beveiliging, privacy, schaalbaarheid, slimme contracten en economie, met allerlei onderliggende uitdagingen. In dit proefschrift onderzoeken wij de drie onderwerpen die het meest cruciaal zijn voor de lange termijn functionaliteit van Bitcoin-achtige blockchains. Dit zijn veiligheid, schaalbaarheid en de economische aspecten. Ons werk kan onderverdeeld worden in twee onderwerpen: transactiepropagatie en payment channel-netwerken.

Transactiepropagatie of adverteren verwijst naar de verspreiding van nieuw gecreëerde transacties van klanten in het mining-netwerk. In dit proefschrift onderzoeken wij het gebrek aan motivatie voor het propageren van transacties in peer-to-peer mining-netwerken en stellen wij een mechanisme voor om propagatie te stimuleren. Daarnaast focussen wij op de inefficiënte routering van de transacties en stellen we een slim routeringsmechanisme voor.

Payment channel-netwerken (PCN) zijn veelbelovende laag-2-protocollen die gericht zijn op het verbeteren van de schaalbaarheid van blockchains. In dit proefschrift presenteren wij drie werken over de PCN's. Ten eerste onderzoeken wij de prikkels om deel te nemen aan multi-hop betalingen en stellen wij een winststrategie voor die het gebruik van PCN's zou stimuleren. Ten tweede stellen we de eerste Bitcoin-compatibele virtuele channel constructies op payment channels voor, die de efficiëntie en beschikbaarheid van multi-hop betalingen verbeteren. Ten slotte introduceren wij de eerste post-quantum PCN met behulp van ons post-quantum adapter-handtekeningschema. Onze werken zijn voornamelijk gericht op Bitcoin en zijn PCN, Lightning Network, maar ze kunnen ook worden toegepast op de blockchains en cryptocurrencies met vergelijkbare kenmerken.

1

INTRODUCTION

The pioneering paper of Bitcoin [1] introduced the first commonly used and accepted digital currency, or *cryptocurrency*, that is controlled by a decentralized peer-to-peer network rather than a centralized authority. Twelve years after its implementation, as of January 2021, Bitcoin has peaked with a market cap of 600 Billion Euros, and the total market capitalization of the cryptocurrencies has reached 900 Billion Euros [2]. The recognition of Bitcoin led to the broader development of blockchain technology and various application areas, including finance [3, 4], supply chain [5–7], and energy [8, 9].

Despite the interest in Bitcoin and other cryptocurrencies, several research challenges need to be addressed for their functionality and practical use in real life. Among these challenges, security, incentive-compatibility, and scalability are considered in this thesis for the following reasons. First of all, the security of the ledger against malicious activities is one of the essential properties required from a cryptocurrency. Secondly, since the cryptocurrencies are run by rational miners, each operation done by the miners should be incentive-compatible [10, 11], i.e., for each operation, following the honest protocol behavior should be more profitable than deviating from it. Thirdly, the early practice of the primordial cryptocurrencies showed that they are not scalable in terms of the transaction throughput compared to the traditional payment systems [12]. Overall, it can be said that all three aspects are compulsory requirements for the functionality and practicality of a decentralized cryptocurrency.

This chapter is an introduction to blockchain technology and Bitcoin that form the basis of the thesis. In Section 1.1, we explain the blockchain terminology and its components. In Section 1.2, we explain the off-layer and the payment channel network that is one of the promising scalability solutions investigated in this thesis. Section 1.3 consists of the existing works and the open questions on the aforementioned three dimensions: security, incentive-compatibility, and scalability. Finally, we present the problems that are investigated in this thesis in Section 1.4 and our contributions in Section 1.5.

1.1. BLOCKCHAIN TECHNOLOGY AND BITCOIN

In this thesis, we define blockchain as a decentralized and tamper-resistant ledger that is updated and secured in a distributed structure among untrusted parties. The ledger consists of ordered blocks, which can be composed of coin transactions like in Bitcoin or smart contracts as in Ethereum [13]. The tamper-resistance, also referred as immutability, property of the ledger is provided with the cryptographic hash functions [14]. Hash functions are used to chain the blocks and protect the ordering of the blocks and the integrity of the data. Specifically, each block includes the hash of the previous one that allows the detection of removal or addition of a block as well as the partial change of it. Also, authentication of the ownership on data is provided via digital signature schemes.

The parties who run the blockchain are named as *miners* and the parties who make transactions are *clients*. Clients encode their transactions regarding the *scripting language* of the blockchain, and send to the miners. Miners check the correctness of the transactions and add them to their candidate blocks. Then, the miners update the ledger with the new block selected with respect to a *consensus mechanism* that allows them to agree on the same ledger view. The *access mechanism* of the blockchain defines who can be a miner or a client. In Figure 1.1, the work flow of a blockchain is illustrated.

A blockchain system can be represented by four layers [12]: *hardware*, *network*, *blockchain*, and *off-chain*. The layers are illustrated in Figure 1.2, inspired from [12]. The hardware layer refers to the execution environment, which can be trusted like Intel Software Guard Extensions (SGX) [15], and the network layer is the peer-to-peer network that parties exchange messages. Blockchain layer, also layer-1, consists of the ledger and its characteristics: transaction structure and scripting language, access control and consensus mechanisms. Finally, the off-chain layer, or layer-2, hosts protocols that do not require publication of each transaction on the blockchain, yet their security relies on the blockchain. In this section, we explain the characteristics of the blockchain layer.

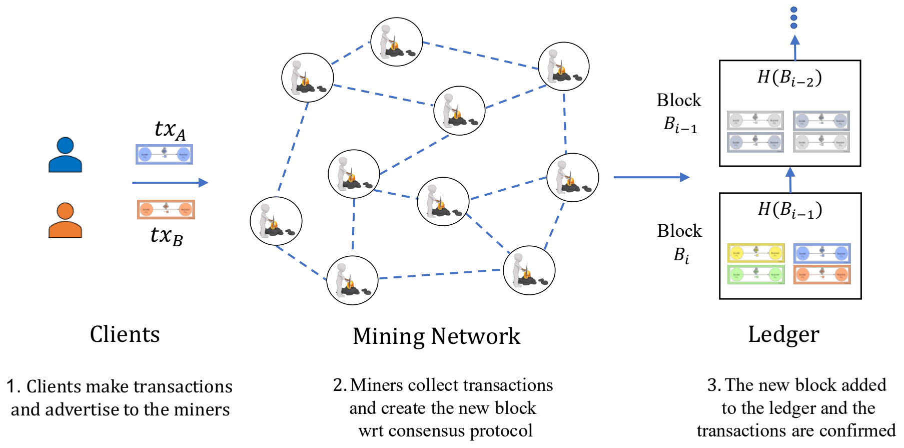

tx_A
tx_B
Block B_{i-1}
Block B_i
$H(B_{i-2})$
$H(B_{i-1})$
Clients
Mining Network
Ledger
1. Clients make transactions and advertise to the miners
2. Miners collect transactions and create the new block wrt consensus protocol
3. The new block added to the ledger and the transactions are confirmed

Figure 1.1: The work flow of a blockchain[1].

[1] Miner image credits: pixabay.com

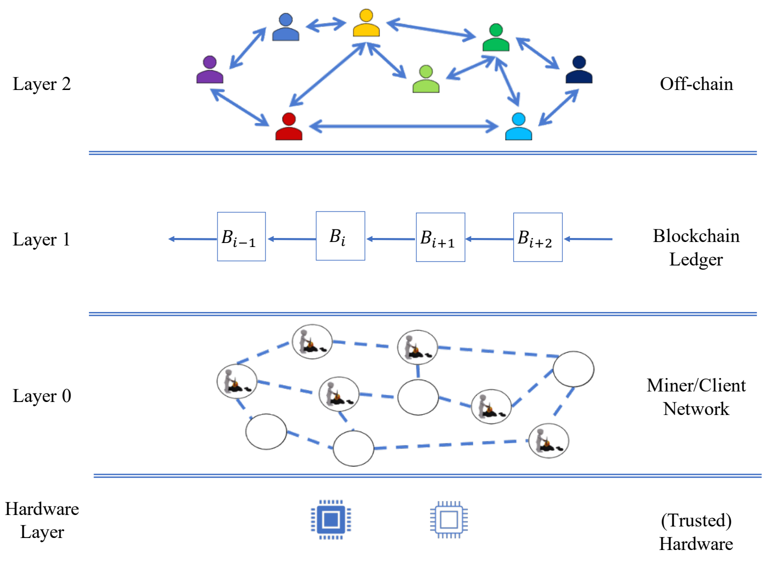


Figure 1.2: The illustration of blockchain layers (inspired from [12])[2].

Transaction Structure and Scripting Language

The transactions of a blockchain are structured with respect to the notion of balance and the scripting language. There are two main balance mechanisms: *unspent transaction output* (UTXO) model used in Bitcoin or *account-based* model used in Ethereum. In the UTXO model, the coins are stored in the transaction addresses, and transactions move coins from input addresses to output addresses. Whereas in the account-based model, parties (or smart contracts) have accounts where transactions move a value from one account to another. The advantage of the UTXO model is the parallelization of transactions where the same client can spend two different UXTOs in separate transactions, and their execution order is irrelevant [16]. However, unlike the account-based one, the UTXO model is stateless and not suitable for the complex contracts that involves multiple parties and state information [17]. For these reasons, the UTXO model is used in cryptocurrencies like Bitcoin that is designed for coin transactions, whereas the account-based model is used in Ethereum, which runs smart contracts.

Transactions are encoded with the scripting language of the blockchain that determines the capabilities of the blockchain. The scripting language can be either a set of predefined and limited operational codes (scripts) like in Bitcoin, or a Turing-complete language as in Ethereum. The Turing-completeness enables to write complex smart contracts, yet it also requires rigorous implementation analysis to avoid attacks like in the Decentralized Autonomous Organization (DAO) case [18].

Bitcoin utilizes the UTXO model and has a restricted scripting language. Each transaction consists of a list of input and output addresses where the input addresses are part of the UTXO set, and the outputs include the out-going addresses of the coins together with

[2]Miner image credits: pixabay.com

their spending conditions. The scripts define the output spending conditions, and the commonly used ones are *Pay-to-PubKeyHash* (P2PKH) and *Pay-to-ScriptHash* [19] that condition on a public key or a hash value. To spend a transaction output, the spending party needs to provide a witness that satisfies the output's spending condition, e.g., a signature that matches the public key hash for the P2PKH.

ACCESS MECHANISM

The access mechanism of a blockchain defines which parties are allowed to perform certain functions. More specifically, there are two classifications concerning the write access for the miners and the read access for the clients [20]. The first classification is about the authentication of the miners: permissioned or permissionless blockchains. In a permissioned blockchain, there is an authority which can be the consortium of the existing miners, that decides who can be miners. Whereas, in the permissionless blockchain, anyone can join the system and start mining it as long as it follows the consensus mechanism. Note that the access mechanism impacts the consensus mechanism, which we will discuss in the following section.

The second classification concerns the client-side: public and private blockchains. In public blockchains, anyone can see what is written on the blockchain, and in private blockchains, only authenticated parties can access the data. The public blockchains are transparent and publicly verifiable because of the read access to everyone, whereas private ones provide confidentiality against parties who do not have access to read. On the one hand, in industrial use cases, permissioned and private blockchains would be preferred since the companies do not share their confidential data. Examples of permissioned blockchains are Hyperledger Fabric [21] and Corda [22]. On the other hand, most of the cryptocurrencies, including Bitcoin, are public and permissionless blockchains with the aim of open write and read access properties.

CONSENSUS

The consensus mechanism is an essential component of a blockchain that enables parties to have a non-conflicting view of the ledger. At each update of the blockchain, miners try to construct the new block that includes new and valid transactions according to the scripting language. Once a new block is constructed, the candidate ledger, including the new block, is shared with the rest of the miners, and the miners accept the new block if its transactions are valid and the block is created with respect to the consensus mechanism. If multiple ledger candidates are available at the same time, miners choose the valid one regarding the chain selection mechanism, such as the longest-chain rule used in Bitcoin and the GHOST rule [23] used in Ethereum.

A well-known consensus mechanism that has been used in peer-to-peer mechanisms for decades is Byzantine fault tolerance (BFT) protocols [24–26]. BFT protocols enable consensus among a group of replicas in a distributed setting where some replicas can be faulty or Byzantine. In BFT protocols, the consensus is achieved by the acknowledgments of the strong majority. More specifically, each party has one vote defined over the party's public-private key pair, and the consensus is reached if more than two-out-of-three parties sign and acknowledge the same view. BFT protocols can be used in permissioned blockchains since parties are known and authenticated to each other regarding their key pair. However, they are not suitable for permissionless blockchains because the network is dynamic where the parties may not know each other's identities, and it is almost costless to create fake/Sybil [27] identities by creating new key pairs.

To enable consensus in a permissionless setting, a new type of consensus protocols has been introduced: *proof-of-X*. The idea in a proof-of-X protocol is designing a voting mechanism that is resistant to the Sybil attacks. In other words, it should not be almost free or costless to create new votes. Some of the commonly used proof-of-X protocols are *proof-of-work* [1], *proof-of-stake* [28–34], *proof-of-space* or *space-time* [35–37]. Proof-of-work is a vote-per-CPU protocol where each miner has a vote proportional to its CPU power. The CPU power is replaced by the stakes in proof-of-stake and by the allocated space or storage in proof-of-space mechanisms. We now explain the proof-of-work mechanism used in the majority of the total capitalization in cryptocurrecies [38].

The proof-of-work (PoW) consensus protocol is introduced in Bitcoin, and it is based on a cryptographic hashing puzzle [1]. Each miner tries to find a nonce value concatenated with the block so that the hash of the block is less than a predefined target value. Since the hash operation is assumed to be one-way function, it is infeasible to calculate a nonce value that satisfies the requirement. That is why each miner has a proportional probability with respect to its CPU power to solve the puzzle. The one who solves the puzzle first propagates the new block in the network. Then, the other miners verify the PoW by re-computing the hash of the block. After the verification, the miners append the block to their ledger and start mining the next one.

The target value in a PoW determines the difficulty of the puzzle and, thereby, the block generation frequency. If the target value is relatively large, then it will be easier to solve the puzzle since the chance of finding a hash value less than the target would be higher for each attempt. In Bitcoin, the target value is adjusted every two weeks such that the average time to create a new block would be 10 minutes. In addition, Bitcoin's consensus protocol restricts the block size to be at most 1 MB, which can be increased up to 4 MB [39]. The block size and frequency constraints limit Bitcoin's transaction throughput to 7 transactions per second. A promising way to improve the throughput is the off-chain protocols, which are explained in the following section.

1.2. Off-chain Layer and Payment Channel Networks

Off-chain or *layer-2* protocols refer to the protocols that do not necessarily publish all of the transactions on the blockchain [12]. We refer *on-chain* transactions for the ones published on the blockchain and *off-chain* transactions for the rest. The off-chain transactions are only validated by the involved parties, and they are not verified by the miners nor stored in the blocks. By removing the miner and blockchain involvement for these transactions, the off-chain protocols aim to improve the scalability of the overall system. The well-known layer-2 protocols are *channel networks* and *commit-chains*.

In a channel network, parties create a channel by locking collaterals with an on-chain transaction, and then they can update the channel arbitrarily many times without interacting with the blockchain. Also, the ones who do not have a direct channel can be connected through the path of channels between them. There are two types of channels: *Payment channels* as in Lightning Network [40] which are used for sending or receiving payments, and *state channels* like in Raiden [41] that can handle smart contracts. There are also centralized networks, called *hubs*, where each party connects to the central party, namely *tumbler*, by creating a channel [42]. Compared to a channel network, an advantage of a hub network is the efficient routing of payments since any two parties are connected via the tumbler. The drawback is the centralization that can cause a single point of failure or censorship of the payments by the tumbler.

The other off-chain protocol, commit-chain, also has a central party called the *operator*. In a commit-chain, parties can join the system by contacting the operator, which does not require an interaction with the blockchain. Then, the parties can start using the commit-chain for the off-chain transactions. The operator periodically publishes a checkpoint on the blockchain where the parties can check their funds. The examples of commit-chains are NOCUST [43] and Plasma [44]. The advantages of commit-chains compared to channels are (i) parties can join the system without interacting with the blockchain, and (ii) the operator is not required to lock collateral for routing a payment between two parties. The drawbacks are the same centralization concerns as mentioned for channel hubs and the necessity of highly expressive scripting language to instantiate a commit-chain [12]. In other words, the commit-chains work on smart contract enabled blockchains, whereas the channels can build on Bitcoin-like blockchains with a restricted scripting language.

Among the off-chain protocols, channel networks are promising and popular solutions because of their decentralized structure. In this manner, Lightning Network, the payment channel network of Bitcoin, is the first deployed layer-2 protocol. As of January 2021, it has more than 35 thousand channels with the total locked coins worth 29 million Euros [45]. Now, we explain the payment channel networks and, specifically, Lightning Network in more detail.

PAYMENT CHANNEL NETWORKS

A payment channel allows two parties to exchange arbitrarily many transactions without publishing them on the blockchain. Parties first open a channel by publishing an on-chain transaction that locks both parties' coins into the channel. The locked coins define the initial balances of the parties in the channel. Then, as long as there are enough balances, parties can send and receive coins by exchanging authenticated messages between themselves. It allows them to realize high transaction throughput and almost-instant transaction confirmations. Eventually, parties can close the channel with another on-chain transaction. Note that opening and closing a channel costs two on-chain transactions. Therefore, it would not be beneficial to create a channel between parties who do not make more than a single transaction. To facilitate off-chain transactions between parties who do not have a direct channel, *multi-hop* payment can be used in a network of channels. In a payment channel network (PCN), parties can be connected via a route of channels that can carry out the payment.

Lightning network [40] is the payment channel network built on Bitcoin. The three stages of a channel in the Lightning Network, namely channel opening, update, and closing, are illustrated in Figure 5.2. To open a channel, parties publish the funding transaction where the inputs consists of UTXO addresses of the parties for their locked coins and the output is the total coins that can spend by both parties' signatures. In channel updates, parties create and exchange signatures of the new commit transactions that spend the output of the funding transaction to the latest state of the channel, and revoke the previous ones. Finally, parties close the channel by publishing a closing transaction regarding the latest state and claim their coins on the blockchain.

Lightning Network uses a *revocation-by-punishment* mechanism for the channel updates. In more detail, every time parties update the channel state, they revoke the previous commit transactions by sharing some secret values with each other that allow them to punish the misbehavior of publishing an old one. The built-in mechanism in Lightning Network requires to have two commit transactions, one for each party. In other words, there are two commit transactions of each channel state and each party can only publish his own transaction. Thereby, if an old state is published, the one who published the corresponding commit transaction can be identified and punished accordingly.

Recently, a new revocation mechanism is proposed in [46] that requires only one commit transaction per channel. It utilizes a new type of signature scheme called *adaptor signature* [47]. An adaptor signature is a two-step algorithm built over a digital signature scheme where first a partial signature is generated, and then its completion to a full signature reveals a secret value to the parties who have the partial signature. The revealing secret mechanism allows parties to embed a condition into the signature. Meanwhile, for any party who does not know the partial signature, the completed signature looks like an ordinary signature of the underlying digital signature scheme. Thus, if an adaptor signature is published on the blockchain, the miners will only see and verify the full signature. The advantage of the adaptor signature is that the embedded condition is not restricted to the blockchain's scripting language, and it does not require any computational or storage

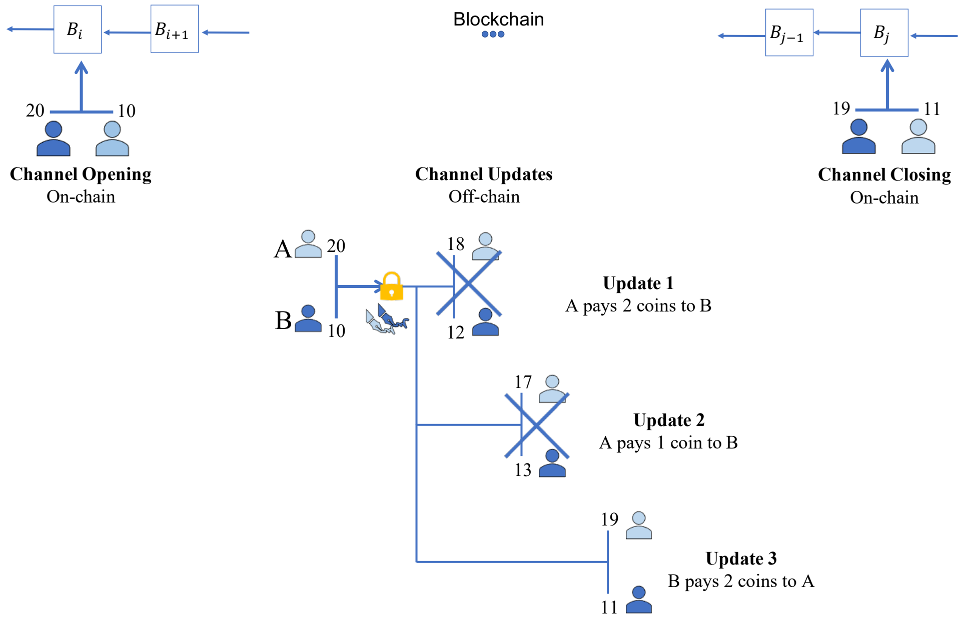


Figure 1.3: Channel opening, update and closing steps in the Lightning Network.

cost because the miners would not see it. In [46], the authors use adaptor signatures to embed different revealing conditions to each party's partial signature of the commit transaction. Thereby the full signature would reveal a unique and identifiable secret of the completing party. In addition to the channel structure, adaptor signatures are also used for multi-hop payments [48].

In Lightning Network, multi-hop payments are realized via the *hash-time lock contract* (HTLC) protocol based on the preimage verification script of Bitcoin. The script is constructed over a hash value, and it requires the reveal of a preimage of the corresponding hash. Briefly, first, the sender and receiver of the payment agree on a hash value, and the sender chooses the payment route. Then, each channel in the route locks the payment amount plus a fee regarding a preimage condition of a given hash value. Here, the fee is paid to the owners of the intermediary channels for their participation in the payment. After all the channels are locked, the receiver reveals the preimage, and all channels are updated accordingly. If the preimage is not revealed, then the locked collaterals are released after a predetermined time.

The PCN protocols are recent developments in blockchain technology. Thus, there are several open questions, especially regarding the multi-hop payments. In the following section, we explain the ones addressed in this thesis.

1.3. RESEARCH DIRECTIONS AND OPEN QUESTIONS

In this section, we discuss the research and open questions in Bitcoin-like blockchains regarding security, incentive-compatibility, and scalability aspects.

SECURITY

We investigate the security of a blockchain in two parts: the security of the ledger functionality against miners' behavior and the security of the cryptographic algorithms used in the blockchain. A ledger should satisfy the security properties of *persistence* and *liveness* [49, 50]. The persistence concerns the consistency of the transactions and immutability of the ledger, and the liveness is related to the availability and growth of the ledger in terms of the addition of new transactions. Persistence implies that if a transaction is accepted as stable by an honest party, then the remaining honest parties cannot have a transaction that conflicts with the stable one. Liveness means that if all honest parties want to add a transaction to the ledger, it will eventually be added and stabilized.

One of the significant security concerns regarding persistence is *double spending attack* that spends the same input in two different transactions. It can happen by creating a *fork* where there are two views of the chain acceptable by the miners. As a precaution, the transactions are assumed to be valid only after several consecutive blocks are added, e.g., the rule of thumb for Bitcoin is six blocks. Nevertheless, it is a known fact that a dishonest majority can create forks even after several blocks are already added. In addition to the dishonest majority attack, also known as 51% attack, there are minority attacks that rely on the miners' rationality and the potential malicious behaviors that aim to maximize their profit, namely *selfish mining* attacks [51–53]. Selfish mining attacks show that, unlike the conventional wisdom [51], miners holding less than the majority of the power can still benefit from deviating the honest protocol by not sharing newly discovered blocks with the rest. These attacks can be mitigated by timestamps or checking the freshness of the block [54–56].

In addition to the security of the ledger functionality, a blockchain's immutability and authentication heavily rely on the cryptographic algorithms used in the protocol, which are hash functions and signature schemes. The most commonly used hash functions and signature schemes such as SHA-2 [57], SHA-3 [58] and ECDSA [59] have shown their resilience against the classical cryptanalysis techniques and have been used in many other applications as well. However, for long-term security, the quantum algorithms and the evolution of quantum computers should be taken into account. With the presence of quantum computers, Shor's algorithm [60] can break most of the existing public-key algorithms, including RSA [61], ECDSA [59]. Also, Grover's algorithm [62] can be used to detect hash collisions. Yet, the traditional hash functions are considered to be resistant to quantum attacks by simply increasing the hash size [63]. To improve security concerning signature schemes, blockchains with lattice-based post-quantum signature schemes have been proposed [64, 65]. However, there is still a need for a post-quantum secure adaptor signature scheme. As described before, adaptor signatures are special type of signatures used in several blockchain protocols such as PCNs [46, 48], hubs [66] and atomic swaps between different cryptocurrencies [67]. Note that a post-quantum secure adaptor signature scheme can be used to design a post-quantum secure PCN, which is also an open problem.

INCENTIVE-COMPATIBILITY

Blockchains, especially permissionless ones, are run by rational miners who try to maximize their profits and, thereby, each blockchain operation should be incentive-compatible [10, 11]. An operation is incentive-compatible if the miners maximize their profit by following the honest behavior on the operation [68]. The most commonly used incentive tools in blockchains are block rewards and transaction fees. The block reward provides the block creation incentive, and the fee of a transaction incentivizes the miners to add it to the block. The challenges regarding the incentive-compatible block creation procedure and selfish mining attacks are already discussed within the security section.

Another essential operation done by the miners is the transaction advertisement. Most of the blockchains, including Bitcoin, do not provide an incentive for the advertisement [69]. There exist two proposals for incentive-compatible transaction advertisement mechanisms that give some of the transaction fees to the propagating nodes [69, 70]. However, these proposals are limited to specific scenarios or network topologies, and a generic mechanism suitable for any blockchain network is an open question.

SCALABILITY

In a permissionless blockchain where every miner needs to verify and store every transaction, the throughput is limited by the following factors: the network latency, consensus protocol, and the block size. For example, Bitcoin can handle 7 transactions per second (tps) because of the limited block size and generation time of 10 mins, whereas Visa can have more than a thousand tps [71]. A straightforward solution by increasing the block capacity or the generation frequency can cause different views of the ledger in the network because of the network latency and damage the persistence of the ledger [49].

There have been several proposals to improve the scalability of the blockchains, including new consensus protocols, *sharding* and *side-chains* [72]. The alternative consensus protocols are either an extension over the Bitcoin's PoW protocol [73, 74] or based on new mechanisms like PoS [32, 75]. For example, Bitcoin-NG [74] presents an improvement over the proof-of-work of Bitcoin that separates the transactions from the proof. Sharding aims to improve the scalability by splitting the network into smaller groups [76], and side-chains with a more hierarchical structure [77]. Furthermore, a recently proposed alternative scalability solutions are *layer-2* or *off-chain* protocols [12]. Layer-2 protocols improve the scalability of the overall system by not publishing every transaction on the blockchain. Among all these proposals, layer-2 protocols are promising and receiving great interest from the research community because of the following reasons: (i) since the transactions are not published on the blockchain, the throughput is not restricted by the limitations of the consensus protocol, (ii) they rely on the consensus protocol of the blockchain, (iii) they can be deployed on the most of the existing blockchains without requiring any changes. For the Bitcoin-like cryptocurrencies, as a layer-2 solution, payment channel networks (PCNs) have been proposed recently. However, there are several open problems in the existing PCNs.

One of the open problems in PCNs is that the existing proposals are vulnerable to quantum algorithms, and they are not post-quantum secure, which is already mentioned in the security section. In addition, the characteristics of the multi-hop payment mechanisms are not well studied. Multi-hop payments create a payment route among the parties who do not have a direct channel by interacting with the intermediary parties on the route. In this regard, we identify two problems: the first one is related to the intermediary parties' incentives, and the second one is about the necessity of the involvement of the intermediaries in each multi-hop payments. The latter one can be mitigated using *virtual channels* [78]. However, the existing proposals utilize Turing-complete smart contract constructions and are not compatible with Bitcoin-like blockchains. We elaborate on the problems in the following section.

1.4. PROBLEM STATEMENT

Blockchain technology has been evolving in the interdisciplinary research community of computer science, mathematics, and economics. This thesis aims to address highly impactful problems regarding security, incentive-compatibility, and scalability aspects. Here, we present the research questions of the thesis, which can be explained in two groups: transaction propagation and payment channel networks.

Our works mainly focus on Bitcoin, yet it is possible to extend them to other blockchains having similar structures. More specifically, the problems related to transaction advertisement apply to the permissionless blockchains having peer-to-peer mining networks. Also, our protocols for payment channel networks can be adopted by the cryptocurrencies having UTXO model and scripting capabilities of timelocks and signature schemes. We now explain the existing problems and our research questions.

Transaction Propagation

Transaction propagation or advertisement refers to the dissemination of newly created transactions of clients in the mining network. Miners must be aware of the transactions because only the miners who have the transaction can add it to their blocks in the block creation procedure. To provide an incentive for the propagation, it is possible to give some of the transaction fees to the propagating nodes. However, it is challenging to provide a Sybil-proof system in a peer-to-peer mining network that only benefits the real propagators. The existing works are limited to specific network topologies of the mining networks, which brings the first research question:

Q1: *How to design an incentive-compatible and Sybil-proof transaction propagation protocol that is suitable for well-connected peer-to-peer mining networks?*

Another problem in transaction propagation is the multiple broadcast of the same transaction. A miner would receive each transaction twice: first during the transaction propagation, then after the block propagation which includes the transaction. On the one hand, block propagation is essential since each miner should be aware of blocks to validate the ledger. On the other hand, it is unnecessary to advertise the transaction to all miners but the one who creates the next block, namely the round leader. An improved version of proof-of-work given in Bitcoin-NG protocol allows to identify the round leader before the transaction block are created. This brings our second question:

Q2: *How to get rid of the redundancy of the transaction propagation if the miner of the block is known in advance?*

Payment Channel Networks

Off-chain protocols, especially PCNs, are one of the prominent solutions for scalability issues. In a PCN, parties with a direct channel can send and receive coins by only exchanging authenticated messages, and parties who do not have a direct channel can utilize multi-hop payments by creating a payment route. Multi-hop payments require intermediary parties' involvement in the route, and they receive a fee for their participation. The fees values are determined by the owner of the channels and publicly known, and a sender of a multi-hop payment would choose the route with the cheapest total fee. Note that the fees are the only incentives of the intermediary parties for participating in a multi-hop payment, and consequently, they are essential for the functionality of the network. Regarding the fees, we present our third question:

Q3: *How to determine the fee of a channel that maximizes the profit and encourages the owner of the channel to create new channels yielding to the growth of the network?*

Secondly, since the intermediary parties are involved in the multi-hop payments, they need to be online and update their channels in the route. A recently proposed concept of virtual channels aims to mitigate this issue. For a scenario where sender and receiver are separated with an intermediary, a virtual channel can be opened with an off-chain agreement between the three parties. Once the virtual channel is opened, the end-parties can send and receive payment without involving the intermediary party. The existing constructions for virtual channels rely on expressive scripting language and smart contracts. Our fourth question is about Bitcoin-compatible virtual channel constructions:

Q4: *How to design a virtual channel between two parties separated with an intermediary that is compatible with Bitcoin-like blockchains?*

Finally, the existing PCNs do not have long-term security in the presence of quantum computers. This is because the signature schemes used in these PCNs are vulnerable to quantum attacks. Therewithal, recent developments in the PCNs utilize a new type of signature scheme, namely adaptor signature, which brings our fifth question:

Q5: *How to design a post-quantum secure adaptor signature, and thereby payment channel network?*

1.5. CONTRIBUTION OF THE THESIS

Each technical chapter of the thesis consists of an integral copy of a paper. The chapters are independent of each other and can be read separately. We preserve the technical details of the papers as their original publication with possible minor changes. This may lead to various notations and overlapping preliminaries and related work sections in different chapters. The outline of the thesis is given as follows:

CHAPTER 2
TRANSACTION PROPAGATION ON PERMISSIONLESS BLOCKCHAINS: INCENTIVE AND ROUTING MECHANISMS

In this chapter, we address the research questions **Q1** and **Q2** and present our contributions for transaction propagation regarding incentive-compatible and bandwidth-efficient routing mechanisms. First, we formulate the propagation problem regarding incentive-compatibility and Sybil-proofness. For Bitcoin-like blockchains where only the current round leader is rewarded for the block, we present the first Sybil-proof and incentive-compatible propagation mechanism suitable for any network that is 2- or more connected. We also show that it is not possible to have a Sybil-proof mechanism if the mining network is a 1-connected network. In addition to incentives, we present a routing mechanism that reduces the redundant propagation cost from the network size to a factor of shortest path length, which can be up to 99% improvement. The routing mechanism requires the round leader to be known in advance before adding the transactions, which is suitable for the improved proof-of-work mechanism given in Bitcoin-NG. The chapter is an integral copy of the paper "*Transaction Propagation on Permissionless Blockchains: Incentive and Routing Mechanisms*" by Ersoy, O., Ren, Z., Erkin, Z., and Lagendijk, R. L. in CVCBT 2018, IEEE, pp. 20–30.

Chapter 3
How to Profit from Payment Channels

In this chapter, we address the research question **Q3** that concerns the fees in multi-hop payments, and we aim to improve the effectiveness of PCNs by increasing the benefits of the participations. Our work focuses on the Lightning Network, yet it gives insights for the other PCNs as they have similar fee structures. In this work, we provide the formula of the profit optimization problem where a party creates new channel connections and assigns a fee value for each of them to maximize its profit. We show that our problem is NP-hard by reducing it to the maximizing betweenness centrality problem in graphs. We propose a heuristic strategy based on graph-theoretical notions that maximizes the gain of the user by creating new connections with a greedy algorithm. We simulate the strategy using a snapshot of the Lightning Network to quantify the impact of the gain and show that it is at least a factor of two better than the default strategy. Our analyses also show that even if the party does not create new channels, the profit can be significantly improved by only applying our fee selection mechanism. This chapter is an integral copy of the paper "*How to Profit from Payments Channels*" by Ersoy, O., Roos, S., and Erkin, Z. in Financial Cryptography 2020, pp. 284–303.

Chapter 4
Bitcoin-Compatible Virtual Channels

In this chapter, we address the research question **Q4**, and we propose the first virtual channel constructions that are compatible with Bitcoin. We formalize the ideal functionality of the virtual channels, present two constructions that realize the ideal functionality and implement a prototype of our proposals. Our constructions differ on the validity property that provides the guarantee on the channel's availability for a predefined period of time. In the virtual channels with validity construction, the intermediary party cannot offload the virtual channel before the validity time, whereas, in without validity construction, the intermediary party can offload the channel without requiring the involvement of other parties at any time. The chapter is an integral copy of the paper "*Bitcoin-Compatible Virtual Channels*" by Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P. and Riahi, S. in IEEE Symposium on Security and Privacy 2021 (in press). This work is primarily conducted during an internship at TU Wien. The author of the thesis mainly worked on designing the virtual channels with validity construction together with the researchers from TU Wien.

Chapter 5
Post-Quantum Adaptor Signatures and Payment Channel Networks

In this chapter, we address the research question **Q5** regarding post-quantum security. We propose the first post-quantum adaptor signature, which relies on the standard lattice assumptions. Using our adaptor signature, we construct post-quantum secure PCN and atomic swap protocols. Our constructions can be utilized in Bitcoin-like blockchains having UTXO model and timelock scripts by replacing their signature scheme with our underlying post-quantum signature scheme. The chapter is an integral copy of the paper "*Post-Quantum Adaptor Signatures and Payment Channel Networks*" by Esgin, M. F., Ersoy, O., and Erkin, Z. in ESORICS 2020, pp. 378–397. The author of the thesis mainly worked on formulating adaptor signature definition into the post-quantum setting and integrating our protocol into atomic swaps and payment channel networks together with a researcher from Monash University.

The following papers are published during the Ph.D. study but not included in the thesis since they do not contribute to the overall story of the thesis or they significantly overlap with a chapter of the thesis:

1. **Ersoy, O.**, Genç, Z. A., Erkin, Z., and Conti, M. Practical Exchange for Unique Digital Goods. In *IEEE DAPPS* (2021), IEEE, (in press).
2. Aumayr, L., **Ersoy, O.**, Erwig, A., Faust, S., Hostakova, K., Maffei, M., Moreno-Sanchez, P., and Riahi, S. Generalized bitcoin-compatible channels. *IACR Cryptol. ePrint Arch. 2020* (2020), 476.
3. **Ersoy, O.**, Erkin, Z., and Lagendijk, R. L. TULIP: A fully incentive compatible blockchain framework amortizing redundant communication. In *EuroS&P Workshops* (2019), IEEE, pp. 396–405.
4. **Ersoy, O.**, Erkin, Z., and Lagendijk, R. L. Decentralized incentive-compatible and sybil-proof transaction advertisement. In *MARBLE* (2019), Springer, pp. 151–165.
5. el Maouchi, M., **Ersoy, O.**, and Erkin, Z. DECOUPLES: a decentralized, unlinkable and privacy-preserving traceability system for the supply chain. In *SAC* (2019), ACM, pp. 364–373.
6. van der Laan, B., **Ersoy, O.**, and Erkin, Z. MUSCLE: authenticated external data retrieval from multiple sources for smart contracts. In *SAC* (2019), ACM, pp. 382–391.
7. el Maouchi, M., **Ersoy, O.**, and Erkin, Z. Trade: A transparent, decentralized traceability system for the supply chain. In *Proceedings of 1st ERCIM Blockchain Workshop 2018* (2018), European Society for Socially Embedded Technologies (EUSSET).

REFERENCES

[1] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system,* (2008).

[2] CoinMarketCap, *Bitcoin market capitalization,* (2019), available at: https://coinmarketcap.com/currencies/bitcoin/.

[3] P. C. Treleaven, R. G. Brown, and D. Yang, *Blockchain technology in finance,* Computer **50**, 14 (2017).

[4] A. Tapscott and D. Tapscott, *How blockchain is changing finance,* Harvard Business Review **1**, 2 (2017).

[5] R. Casado-Vara, J. Prieto, F. de la Prieta, and J. M. Corchado, *How blockchain improves the supply chain: case study alimentary supply chain,* in *FNC/MobiSPC*, Procedia Computer Science, Vol. 134 (Elsevier, 2018) pp. 393–398.

[6] D. Dujak and D. Sajter, *Blockchain applications in supply chain,* in *SMART supply network* (Springer, 2019) pp. 21–46.

[7] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, *Blockchain technology and its relationships to sustainable supply chain management,* International Journal of Production Research **57**, 2117 (2019).

[8] C. Burger, A. Kuhlmann, P. Richard, and J. Weinmann, *Blockchain in the energy transition. a survey among decision-makers in the german energy industry,* DENA German Energy Agency **60** (2016).

[9] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum, and A. Peacock, *Blockchain technology in the energy sector: A systematic review of challenges and opportunities,* Renewable and Sustainable Energy Reviews **100**, 143 (2019).

[10] G. W. Peters and E. Panayi, *Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money,* in *Banking Beyond Banks and Money* (Springer, 2016) pp. 239–278.

[11] Y. Sompolinsky and A. Zohar, *Bitcoin's underlying incentives,* Commun. ACM **61**, 46 (2018).

[12] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, *Sok: Layer-two blockchain protocols,* in *Financial Cryptography*, Lecture Notes in Computer Science, Vol. 12059 (Springer, 2020) pp. 201–226.

[13] G. Wood, *Ethereum: A secure decentralised generalised transaction ledger,* Ethereum Project Yellow Paper **151** (2014).

[14] F. Tschorsch and B. Scheuermann, *Bitcoin and beyond: A technical survey on decentralized digital currencies,* IEEE Commun. Surv. Tutorials **18**, 2084 (2016).

[15] Intel, *Intel software guard extensions (intel sgx),* (2019), available at: https://software.intel.com/en-us/sgx.

[16] V. Buterin, *Thoughts on utxos,* (2016), available at: https://medium.com/@ConsenSys/thoughts-on-utxo-by-vitalik-buterin-2bb782c67e53/.

[17] F. Sun, *Utxo vs account/balance model,* (2018), available at: https://medium.com/@sunflora98/utxo-vs-account-balance-model-5e6470f4e0cf.

[18] Q. DuPont, *Experiments in algorithmic governance: A history and ethnography of "the dao," a failed decentralized autonomous organization,* Bitcoin and beyond , 157 (2017).

[19] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, and J. Herrera-Joancomartí, *Analysis of the bitcoin UTXO set,* in *Financial Cryptography Workshops*, Lecture Notes in Computer Science, Vol. 10958 (Springer, 2018) pp. 78–91.

[20] K. Wüst and A. Gervais, *Do you need a blockchain?* in *CVCBT* (IEEE, 2018) pp. 45–54.

[21] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick, *Hyperledger fabric: a distributed operating system for permissioned blockchains,* in *EuroSys* (ACM, 2018) pp. 30:1–30:15.

[22] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, *Corda: An introduction,* R3 CEV, August **1**, 15 (2016).

[23] Y. Sompolinsky and A. Zohar, *Accelerating bitcoin's transaction processing. fast money grows on trees, not chains.* IACR Cryptology ePrint Archive **2013** (2013).

[24] L. Lamport, R. E. Shostak, and M. C. Pease, *The byzantine generals problem,* ACM Trans. Program. Lang. Syst. **4**, 382 (1982).

[25] L. Lamport, *The part-time parliament,* ACM Trans. Comput. Syst. **16**, 133 (1998).

[26] M. Castro and B. Liskov, *Practical byzantine fault tolerance,* in *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*, edited by M. I. Seltzer and P. J. Leach (USENIX Association, 1999) pp. 173–186.

[27] J. R. Douceur, *The sybil attack,* in *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, Lecture Notes in Computer Science, Vol. 2429, edited by P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron (Springer, 2002) pp. 251–260.

[28] S. King and S. Nadal, *Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,* self-published paper, August **19** (2012).

[29] I. Bentov, A. Gabizon, and A. Mizrahi, *Cryptocurrencies without proof of work,* in [79], pp. 142–157.

[30] S. Micali, *ALGORAND: the efficient and democratic ledger,* CoRR **abs/1607.01341** (2016), arXiv:1607.01341 .

[31] I. Bentov, R. Pass, and E. Shi, *Snow white: Provably secure proofs of stake,* IACR Cryptology ePrint Archive **2016**, 919 (2016).

[32] A. Kiayias, A. Russell, B. David, and R. Oliynykov, *Ouroboros: A provably secure proof-of-stake blockchain protocol,* in *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, Lecture Notes in Computer Science, Vol. 10401, edited by J. Katz and H. Shacham (Springer, 2017) pp. 357–388.

[33] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas, *Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability,* in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, edited by D. Lie, M. Mannan, M. Backes, and X. Wang (ACM, 2018) pp. 913–930.

[34] B. David, P. Gazi, A. Kiayias, and A. Russell, *Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain,* in *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, Lecture Notes in Computer Science, Vol. 10821, edited by J. B. Nielsen and V. Rijmen (Springer, 2018) pp. 66–98.

[35] G. Ateniese, I. Bonacina, A. Faonio, and N. Galesi, *Proofs of space: When space is of the essence,* in *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, Lecture Notes in Computer Science, Vol. 8642, edited by M. Abdalla and R. D. Prisco (Springer, 2014) pp. 538–557.

[36] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, *Proofs of space,* in *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, Lecture Notes in Computer Science, Vol. 9216, edited by R. Gennaro and M. Robshaw (Springer, 2015) pp. 585–605.

[37] T. Hønsi, *Spacemint-a cryptocurrency based on proofs of space,* IACR Cryptology ePrint Archive (2017).

[38] CoinMarketCap, *Today's cryptocurrency prices by market cap,* (2020), available at: `https://coinmarketcap.com/`.

[39] B. Wiki, *Segregated witness,* (2019), available at: `https://en.bitcoin.it/wiki/Segregated_Witness`.

[40] J. Poon and T. Dryja, *The bitcoin lightning network: scalable off-chain instant payments,* (2016), available at: `https://lightning.network/lightning-network-paper.pdf`.

[41] B. T. AG, *Raiden network,* (2019), available at: `https://raiden.network/`.

[42] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, *Tumblebit: An untrusted bitcoin-compatible anonymous payment hub,* (2017).

[43] R. Khalil, A. Gervais, and G. Felley, *Nocust–a securely scalable commit-chain,* (2018), available at: https://eprint.iacr.org/2018/642.pdf.

[44] J. Poon and V. Buterin, *Plasma: Scalable autonomous smart contracts,* (2017), available at: https://plasma.io/plasma.pdf.

[45] *Real-time lightning network statistics,* (2019), available at: https://1ml.com/statistics.

[46] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi, *Generalized bitcoin-compatible channels,* IACR Cryptol. ePrint Arch. **2020**, 476 (2020).

[47] A. Poelstra, *Lightning in scriptless scripts,* Mailing list post (), https://lists.launchpad.net/mimblewimble/msg00086.html.

[48] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, *Anonymous multi-hop locks for blockchain scalability and interoperability,* in *NDSS* (The Internet Society, 2019).

[49] J. A. Garay, A. Kiayias, and N. Leonardos, *The bitcoin backbone protocol: Analysis and applications,* in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, Lecture Notes in Computer Science, Vol. 9057, edited by E. Oswald and M. Fischlin (Springer, 2015) pp. 281–310.

[50] R. Pass, L. Seeman, and A. Shelat, *Analysis of the blockchain protocol in asynchronous networks,* in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, Lecture Notes in Computer Science, Vol. 10211, edited by J. Coron and J. B. Nielsen (2017) pp. 643–673.

[51] I. Eyal and E. G. Sirer, *Majority is not enough: Bitcoin mining is vulnerable,* in *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, Lecture Notes in Computer Science, Vol. 8437, edited by N. Christin and R. Safavi-Naini (Springer, 2014) pp. 436–454.

[52] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, *Optimal selfish mining strategies in bitcoin,* in *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, Lecture Notes in Computer Science, Vol. 9603, edited by J. Grossklags and B. Preneel (Springer, 2016) pp. 515–532.

[53] K. Nayak, S. Kumar, A. Miller, and E. Shi, *Stubborn mining: Generalizing selfish mining and combining with an eclipse attack,* in *IEEE European Symposium on*

Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016 (IEEE, 2016) pp. 305–320.

[54] E. Heilman, *One weird trick to stop selfish miners: Fresh bitcoins, A solution for the honest miner (poster abstract),* in *Financial Cryptography Workshops*, Lecture Notes in Computer Science, Vol. 8438 (Springer, 2014) pp. 161–162.

[55] S. Solat and M. Potop-Butucaru, *Zeroblock: Preventing selfish mining in bitcoin,* CoRR **abs/1605.02435** (2016).

[56] M. Saad, L. Njilla, C. A. Kamhoua, and A. Mohaisen, *Countering selfish mining in blockchains,* in *ICNC* (IEEE, 2019) pp. 360–364.

[57] N. I. of Standards and Technology, *FIPS PUB 180-2: Secure Hash Standard*, Tech. Rep. (2002).

[58] N. I. of Standards and Technology, *FIPS PUB 202 – SHA-3 standard: Permutation-based hash and extendable-output functions*, Tech. Rep. (2015).

[59] D. Johnson, A. Menezes, and S. Vanstone, *The elliptic curve digital signature algorithm (ecdsa),* International journal of information security **1**, 36 (2001).

[60] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,* SIAM J. Comput. **26**, 1484 (1997).

[61] R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems,* Communications of the ACM **21**, 120 (1978).

[62] L. K. Grover, *A fast quantum mechanical algorithm for database search,* in *STOC* (ACM, 1996) pp. 212–219.

[63] T. M. Fernández-Caramés and P. Fraga-Lamas, *Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks,* IEEE Access **8**, 21091 (2020).

[64] W. Yin, Q. Wen, W. Li, H. Zhang, and Z. Jin, *An anti-quantum transaction authentication approach in blockchain,* IEEE Access **6**, 5393 (2018).

[65] Y.-L. Gao, X.-B. Chen, Y.-L. Chen, Y. Sun, X.-X. Niu, and Y.-X. Yang, *A secure cryptocurrency scheme based on post-quantum blockchain,* IEEE Access **6**, 27205 (2018).

[66] E. Tairi, P. Moreno-Sanchez, and M. Maffei, *A^2l: Anonymous atomic locks for scalability and interoperability in payment channel hubs,* IACR Cryptology ePrint Archive **2019**, 589 (2019).

[67] A. Poelstra, *Adaptor signatures and atomic swaps from scriptless scripts,* (), https://github.com/ElementsProject/scriptless-scripts/blob/master/md/atomic-swap.md.

[68] T. Roughgarden, *Algorithmic game theory,* Commun. ACM **53**, 78 (2010).

[69] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, *On bitcoin and red balloons,* in *ACM Conference on Electronic Commerce, EC '12, Valencia, Spain, June 4-8, 2012*, edited by B. Faltings, K. Leyton-Brown, and P. Ipeirotis (ACM, 2012) pp. 56–73.

[70] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, *Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus,* CoRR **abs/1612.02916** (2016), arXiv:1612.02916 .

[71] VISA, *Visa inc. at a glance,* (2015), available at: `https://usa.visa.com/dam/VCOM/download/corporate/media/visa-fact-sheet-Jun2015.pdf`.

[72] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, *On scaling decentralized blockchains - (A position paper),* in [79].

[73] Y. Sompolinsky and A. Zohar, *Secure high-rate transaction processing in bitcoin,* in *Financial Cryptography*, Lecture Notes in Computer Science, Vol. 8975 (Springer, 2015) pp. 507–527.

[74] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, *Bitcoin-NG: A scalable blockchain protocol,* in *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, edited by K. J. Argyraki and R. Isaacs (USENIX Association, 2016) pp. 45–59.

[75] R. Pass and E. Shi, *Hybrid consensus: Efficient consensus in the permissionless model,* in *31 International Symposium on Distributed Computing* (2017) p. 6.

[76] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, *A secure sharding protocol for open blockchains,* in *Conference on Computer and Communications Security* (ACM, 2016) pp. 17–30.

[77] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, *Enabling blockchain innovations with pegged sidechains,* URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains (2014).

[78] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, *Perun: Virtual payment hubs over cryptocurrencies,* in *IEEE Symposium on Security and Privacy* (IEEE, 2019) pp. 106–123.

[79] J. Clark, S. Meiklejohn, P. Y. A. Ryan, D. S. Wallach, M. Brenner, and K. Rohloff, eds., *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, Lecture Notes in Computer Science, Vol. 9604 (Springer, 2016).

2

Transaction Propagation on Permissionless Blockchains: Incentive and Routing Mechanisms

Existing permissionless blockchain solutions rely on peer-to-peer propagation mechanisms, where nodes in a network transfer transaction they received to their neighbors. Unfortunately, there is no explicit incentive for such transaction propagation. Therefore, existing propagation mechanisms will not be sustainable in a fully decentralized blockchain with rational nodes. In this work, we formally define the problem of incentivizing nodes for transaction propagation. We propose an incentive mechanism where each node involved in the propagation of a transaction receives a share of the transaction fee. We also show that our proposal is Sybil-proof. Furthermore, we combine the incentive mechanism with smart routing to reduce the communication and storage costs at the same time. The proposed routing mechanism reduces the redundant transaction propagation from the size of the network to a factor of average shortest path length. The routing mechanism is built upon a specific type of consensus protocol where the round leader who creates the transaction block is known in advance. Note that our routing mechanism is a generic one and can be adopted independently from the incentive mechanism.

This chapter is based on the paper "*Transaction Propagation on Permissionless Blockchains: Incentive and Routing Mechanisms*" by Ersoy, O., Ren, Z., Erkin, Z., and Lagendijk, R. L. in CVCBT 2018, IEEE, pp. 20–30.

2.1. INTRODUCTION

In this work, we investigate transaction propagation on permissionless blockchains with respect to incentive compatibility and bandwidth efficiency. The former, incentive compatibility, is an essential component of permissionless blockchain to maintain its functionality with rational participants [1, 2]. The latter, bandwidth efficiency, is an important factor for efficient use of limited resources available in the network.

Although a number of works have studied incentive compatibility problem of blockchains, they are limited to mining mechanism, e.g. investigating *selfish mining attacks* [3–6], and *block withholding attacks* [7–10]. The existing blockchain solutions such as Bitcoin [11] and Ethereum [12] do not pay attention to incentives for transaction propagation in the network. This is due to the fact that the mining networks in those solutions are centralized in practice [13–15] and thus, they do not exhibit a fully decentralized structure. There are only two works that address incentive compatibility of transaction propagation in blockchain by Babaioff et al. [16] and Abraham et al. [17]. Unfortunately, both works suggest a specific solution for the incentive compatibility but do not provide a formal definition of the problem. Furthermore, the proposed solutions are also designed for certain network topologies.

In terms of bandwidth inefficiency, existing solutions suffer from multiple broadcasting of the same transaction over the network. For example, in Bitcoin, each transaction is received by the nodes (miners) in the network twice: once during the advertisement, i.e. broadcasting of the transaction at the beginning, and once after the validation, i.e. broadcasting of the block including the transaction. While validation is essential since each node in the network stores every validated transaction, the advertisement does not need to be received by all nodes. However, redundancy for advertisement is inevitable in such cases where the round leader who creates the validated block is unknown in advance since the transaction needs to be broadcast to all potential round leaders. In recent blockchain proposals where the round leader is known in advance, what we call *first-leader-then-block* (FLTB) type of consensus protocols [18–21], it is possible to improve bandwidth efficiency by reducing the communication cost by directly routing the transaction to the round leader. To the best of our knowledge, there is no prior work on optimizing bandwidth efficiency for fully decentralized blockchain.

In this work, our contribution is three-fold: 1) Sybil-proof incentive compatible propagation mechanism, 2) bandwidth-efficient routing mechanism, and 3) bandwidth and storage efficient transaction propagation mechanism which combines the first two mechanisms.

We formally define incentive compatibility of propagation mechanisms in fully decentralized blockchain networks. We show that there is no Sybil-proof and incentive compatible propagation mechanism for poorly connected networks (specifically for 1-connected networks). For other network topologies, we find the following incentive compatible and Sybil-proof formula, which distributes the transaction fee among propagating nodes:

$$f_{[i]}^{k} = \begin{cases} F \cdot C(1-C)^{i-1} & \text{for } i < k, \\ F \cdot (1-C)^{k-1} & \text{for } i = k, \end{cases}$$

where F is the fee, k is the length of the propagation path, $f_{[i]}^{k}$ is the share of the i^{th} node in that path, and C is a parameter related to the network topology. The incentive mechanism is independent of the choice of consensus protocol and works with any consensus protocol.

We propose a routing mechanism compatible with FLTB-type consensus protocols. Our proposal reduces the communication cost of the transaction propagation from the size of the network to the scale of average shortest path length. In a random network topology of more than 500 nodes, we achieve over 97% communication cost reduction compared to de facto propagation mechanism for the advertisement. Furthermore, we also present a propagation mechanism which combines our incentive and routing mechanisms in a storage and bandwidth efficient way. For incentive mechanism, our combined protocol requires storing only a single signature to provide the integrity of the path, unlike the existing works, which use a signature chain including signatures of each node in the path.

The rest of the paper is organized as follows: Section 2.2 presents the related work. Our blockchain model and notations are defined in Section 2.3. Section 2.4 formulates requirements of the incentive problem and computes the generic solution. Smart routing mechanism is presented in Section 2.5 and combined with incentive mechanism in Section 2.6.

2.2. Related Work

The lack of incentive for information propagation in a peer-to-peer network has been known and studied in different settings [22–25]. Kleinberg and Raghavan [24] proposed an incentive scheme for finding the answer for a given query in a tree-structured network topology. Li et al. [25] focused on node discovery in a homogeneous network where each node has the same probability of having an answer for the query. In [22, 23], the authors analyzed the incentive problem for multi-level marketing which rewards referrals if the advertisement produces a purchase. In these marketing models, the reward is shared among all nodes in the tree including the propagation path.

The proposed solutions for peer-to-peer networks [22–25] are not applicable for the permissionless blockchains. In peer-to-peer solutions, nodes are asked to provide a specific datum like the position of a peer or the answer to a query. In blockchains, however, transaction propagation is requested to advertise the transactions and eventually place them into a valid block. Alternatively, finding an answer to a query is equivalent to validation of a transaction by round leader in the blockchain. Query propagation in a peer-to-peer network has two main differences compared to a blockchain transaction propagation: nodes do not compete against the ones who forwarded the message to them and nodes cannot generate a response to a query that they do not have the answer, i.e. either they have the right answer or not. Whereas in a blockchain, a block is generated by the round leader and every node is a potential round leader. Essentially, nodes in a blockchain are competitors that have an incentive not to propagate whereas other peer-to-peer nodes do not have the incentive since they cannot generate the answer to the query by themselves.

Recently, blockchain oriented propagation mechanisms have been proposed [16, 17]. In [16], Babaioff et al. uncovered the incentive problem in the Bitcoin system where a rational node (miner) has no incentive to propagate a transaction. They focused on a specific type of network, namely regular d-ary directed tree with a height H, and assumed that each node has the same processing power. In that setting, the authors proposed a hybrid incentive (rewarding) scheme and proved that it is also Sybil-proof. In [17], Abraham et al. proposed a consensus mechanism, Solidus, offering an incentive to propagate transactions and validated blocks (puzzles). In their incentive mechanism, the amount of processing fee passed to the next node is determined by the sender. Both works rely on a signature chain to prevent any manipulation over the path and thereby, to secure the shares of propagating nodes.

[16] and [17] provided analyses of their proposals based on game theory. For the analysis, [16] assumes a tree-structured network which eliminates the case of competition against common neighbors and it is not realistic for blockchain network topology. Whereas, the analysis in [17] is limited to the case of competition between nodes that have common neighbors.

For bandwidth efficiency, to the best of our knowledge, there is no prior work for fully decentralized blockchain without dedicated miners (round leaders). Nevertheless, Li et al. [25] presented a distributed routing scheme for peer-to-peer networks. The authors focused on one-to-one routing which is dedicated to a single target node. Whereas in blockchain it needs to be one-to-all routing, which connects the complete network to the round leader. In addition, [25] does not take into account the possibility of a failing routing caused by a failing or malicious node in the routing path.

2.3. OUR BLOCKCHAIN MODEL AND NOTATIONS

In this section, we describe our blockchain model and the notation used in the paper.
Network. It is a dynamic peer-to-peer network means that there are nodes joining and leaving constantly. Unlike to the existing works [16, 17], we do not have a restriction on the network topology.
Participants. Each participant is denoted by a node in the network. We assume a permissionless blockchain where anyone can participate and contribute to the ledger directly. Moreover, there is no discrimination between nodes (participants), i.e., they can all be the owner of a transaction and propose a block as a miner (round leader). For identification, each node has a public and private key pair and can be validated by his public key.
Consensus and leader election. Incentive mechanism defined in Section 2.4 works regardless of the consensus structure. Whereas, the routing mechanism requires special treatment, which we call *first-leader-then-block* (*FLTB*) type consensus protocols.

FLTB protocols can be defined as the consensus model where the round leader is validated before he proposes the block. Any leader election mechanism which is independent of the prospective block of that leader can be converted into FLTB type. Examples of the FLTB consensus protocols are Proof-of-Work (PoW) based Bitcoin-NG [18] and several Proof-of-Stake (PoS) based ones [19–21].

The rest of the definitions and notations are listed below:

- *Neighbor nodes*: Directly connected nodes in the network, adjacency in the graph.
- *Client*: The source or the sender of a transaction. Client of a transaction T, denoted by c_T.
- *Round Leader*: The legitimate node (participant) who constructs the current block.

- *Intermediary Node*: A node on the transmission path between the round leader and a client.
- $\mathscr{L}^r$: The credential of round leader which validates the round leader of round r and can be verified by all nodes in the network. For example, it could be a special hash value in a PoW protocol or the proof of possessing the chosen coin in a PoS protocol. In general, regardless of the consensus mechanism, credentials are linked to the public key of the leader and can be verified by a corresponding signature.
- $\pi(n_i)$: The probability of node n_i being the round leader, also referred as the capacity of node n_i. It corresponds to the mining power in PoW or the stake size in PoS protocols and is assumed to be greater than zero for every node in the network. $\pi(S)$ corresponds to the total capacity of the all nodes in set S.
- $\mathscr{N}_K^T$: The set of nodes who know (received) the transaction T. $\mathscr{N}_K^{n,T}$ presents the set from the point of view of node n (including n itself).
- $\mathscr{N}_{NK}^T$: The set of nodes who do not know (received) transaction T yet. $\mathscr{N}_{NK}^{n,T}$ denotes the set from the point of view of node n and includes only the neighbors of n.

2.4. INCENTIVE MECHANISM

We now describe our incentive mechanism. For the sustainable functioning of a fully decentralized blockchain where the nodes (participants) are able to create new identities and behave according to their incentives, propagation mechanism needs to be Sybil-proof and incentive compatible [1].

Conventional incentive instrument, namely transaction fee, almost always refers to the reward of the round leader. Here, we refer transaction fee as it consists of the reward to propagate and to validate transactions. Thereby, rational nodes are encouraged not only to validate transactions but also to propagate them. How to determine the fee is out of the scope of this paper but we assume that each transaction fee is predefined by either the client or a known function. We focus on how to automatically allocate the fee among all the contributors of the process.

Fee sharing function (rewarding mechanism). The fee sharing function distributes the transaction fee among the propagating nodes and the round leader. Note that it is highly probable that the same transaction is received more than once by the round leader (and intermediary nodes) because of the propagation mechanism. A rational round leader would choose the one which maximizes his profit. Like existing works [16, 17], the fee sharing function described here deals with the path which is included in the block. For a transaction (added to the block) with fee F and propagation path P, the function $\mathscr{F}$ determines the shares of each node involved:

$$\mathscr{F}:\{F,P\} \longrightarrow \{f_{[i]}^{|P|}\}_{i=1}^{|P|} \text{ where } \sum_{i=1}^{|P|} f_{[i]}^{|P|} = F.$$

$|P|$ denotes the number of nodes involved in the processing of a transaction with fee F, where $|P|-1$ of the nodes are in the propagation path between the client and the round leader. Let $|P| = k$, i.e., length of the propagation path of the transaction is k. Then, $f_{[i]}^{|P|}$

denotes the share of i^{th} node in the propagation path, $f^k_{[k]}$ is the share of the round leader and $\sum_{i=1}^{k} f^k_{[i]} = F$.

In the rest of the section, we formulate the necessities of the fee sharing function to incentivize propagation of an arbitrary transaction T with fee F. An ideal incentive compatible propagation mechanism should satisfy the following properties:

1. *Sybil-proofness*: An intermediary node, as well as the round leader, should not benefit from introducing Sybil nodes to the network.

2. *Game theoretically soundness*: A transaction should not be kept among a subset of the network. There should be adequate incentive for rational nodes willing to propagate, thence it will eventually reach to the whole network.

By formulating these conditions, we achieve the following theorem (where C is a constant which can be chosen according to the network connectivity):

Theorem 2.1. In a 2- or more connected blockchain network, each rational node $n \in \mathcal{N}^T_K$ with $\pi(n) < C \cdot \pi(\mathcal{N}^{n,T}_K)$ propagates transaction T without introducing Sybil nodes, if the transaction fee F is shared by the following method:

$$f^k_{[i]} = \begin{cases} F \cdot C(1-C)^{i-1} & \text{for } 1 \le i < k, \\ F \cdot (1-C)^{k-1} & \text{for } i = k. \end{cases}$$

Proof of the theorem is divided into the following sections. The requirements are formulated in Sections 2.4.1 and 2.4.2, and the fee sharing function satisfying them is computed in Section 2.4.3.

2.4.1. SYBIL-PROOFNESS

Here, we use the same definition of Sybil nodes in [16]: fake identities sharing the same neighbors with the original node that do not increase the connectivity of the network. Because of the Sybil-proof consensus algorithm, Sybil nodes do not increase the capacity of their owner, i.e., the probability of being the round leader.

We investigate the problem in two different settings: 1-connected networks and the rest. k-connected network means that removal of any $k-1$ nodes does not disconnect the network. In 1-connected networks, there exists a bridge which is the only connection between two distinct subnetworks. Though 1-connected network model seems to be unrealistic topology for permissionless blockchains, it is important to see the intuition behind the non-competition effect.

1-connected networks. In 1-connected networks, there are critical nodes which have special positions in the propagation paths between some node pairs. A critical node for a node pair appears in all possible paths between these two nodes. The following lemma shows that non-competing advantage of critical nodes makes it impossible to have a Sybil-proof incentive mechanism for 1-connected networks.

Lemma 2.2 (Impossibility Lemma)**.** For 1-connected networks, there is no Sybil-proof and incentive compatible propagation mechanism which rewards every node in the propagation path.

Proof. Assume that, because of 1-connectedness of the network, a node n_i have a critical position for a transaction T, meaning that it is certain he will be included in the propagation path of that transaction. If n_i is one side of the bridge combining two distinct subnetworks, n_i can be sure that each transaction coming from its subnetwork and validated in the other one has to pass through n_i. In Figure 2.1, we illustrate the two possible paths of a transaction passing through n_i. Since the round leader and also intermediary nodes after n_i will receive one of the paths, they do not have any choice but accept the path sent by n_i.

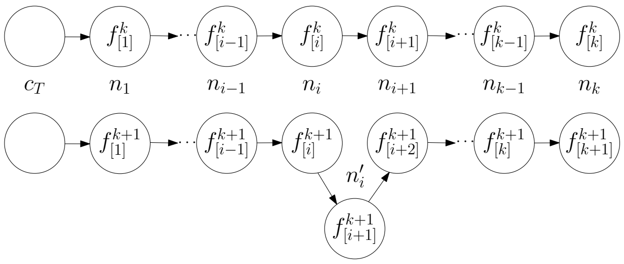


Figure 2.1: The fee sharing before and after a Sybil node $n_{i'}$ added by n_i

Now, we investigate the share of a node n_i with and without a Sybil node. As given in Figure 2.1, n_i is the i^{th} node in the original propagation path and his corresponding fee shares are $f^k_{[i]}$ and $f^{k+1}_{[i]} + f^{k+1}_{[i+1]}$. In order to demotivate n_i, $f^k_{[i]}$ should be greater than or equal to $f^{k+1}_{[i]} + f^{k+1}_{[i+1]}$. Since the position of the node would change for different transactions and rounds, the condition should hold for all positions:

$$\begin{aligned}
\forall\, i \in \{1,\ldots,k\}, \qquad & f^k_{[i]} \geq f^{k+1}_{[i]} + f^{k+1}_{[i+1]} \\
\text{(summing for all } i\text{'s)} \implies \qquad & \sum_{i=1}^{k} f^k_{[i]} \geq \sum_{i=1}^{k} f^{k+1}_{[i]} + \sum_{i=1}^{k} f^{k+1}_{[i+1]} \\
\text{(Definition of } \mathscr{F}\text{)} \implies \qquad & F \geq F - f^{k+1}_{[k+1]} + F - f^{k+1}_{[1]} \\
\implies \qquad & f^{k+1}_{[k+1]} + f^{k+1}_{[1]} \geq F \\
\text{(Definition of } \mathscr{F}\text{)} \implies \qquad & f^{k+1}_{[k+1]} + f^{k+1}_{[1]} = F.
\end{aligned}$$

Therefore, other than the first propagating node and the round leader, there is no reward for the rest of the propagating nodes which contradicts with rational behavior. □

Eclipse and partitioning. Note that this monopolized behavior is similar to the eclipse and partitioning attacks where the adversary separates the network into two distinct group and controls all the connections between them [26, 27]. Indeed, Lemma 2.2 can be generalized to the case where the adversary is able to control all the outgoing connections of a client. In that case, there is no way to deviate the adversary from creating Sybil nodes for that specific transaction. We assume that client nodes are able to defend against the eclipse attacks using the countermeasures defined in [26].

In a 2- or more connected network, there are multiple paths between any two nodes. Therefore, we can immediately focus on the multiple paths case where there are competing paths for the same transaction and the round leader includes one of them to the block.

A node can profit from a fee by either being an intermediary node who propagates it or being the round leader who creates the block. We investigate the Sybil-proof conditions of intermediary nodes and the round leader separately.

Intermediary nodes An intermediary node can be deviated by the actions of the nodes who receive the transaction afterwards. Since there are multiple paths, the round leader will receive the same transaction from at least two different paths. In other words, the round leader would decline all but one of the paths (for each transaction). An intermediary node will be demotivated if introducing a Sybil node would increase the chance of rejection of his path.

If the share of the round leader decreases as the propagation path length increases, then he will choose the shortest path for each transaction. In that case, introducing Sybil nodes will decrease his chance to be included in the block. Therefore, providing larger gain to the leader for choosing the shortest path is sufficient and can be formulated as $f^{k}_{[k]} > f^{k+1}_{[k+1]}$.

Round leader In some cases, round leader is determined before the block is created or even several rounds earlier [18–20]. Since the round leader is guaranteed to be in the propagation path, it is needed to be taken into account separately. In addition, an intermediary node can propagate righteously to his neighbors and then add Sybil nodes for his own mining process. Therefore, in any case (predefined leader or not), it is necessary to make an additional policy for the round leader.

In the case of s Sybil nodes, share of the round leader will change from $f^{k}_{[k]}$ to $\sum_{i=0}^{s} f^{k+s}_{[k+i]}$ for some k. In order to deviate the round leader, $f^{k}_{[k]} \geq \sum_{i=0}^{s} f^{k+s}_{[k+i]}$ is required.

Since the latter condition includes the former one (as $f^{k+1}_{[k]} > 0$), Sybil proofness condition can be formulated as:

$$\forall\, k \geq 1, \forall\, s \geq 1, \quad f^{k}_{[k]} \geq \sum_{i=0}^{s} f^{k+s}_{[k+i]}. \tag{2.1}$$

2.4.2. Incentive Compatibility

The decision of the propagation of a transaction can be analyzed as a simultaneous move game where each party takes action without knowing strategies of the others. All players (nodes in our case) are assumed to be rational and they decide their actions deducing that the others will also act rationally. Some nodes may cooperate with each other. We assume that colluding neighboring nodes already share every transaction with each other and take actions as one. In other words, they act as a single combined node in the network which can be seen as Sybil nodes.

Here, we investigate the propagation decision by comparing the change in the expected rewards for a transaction T. In the beginning, each transaction is shared with some nodes, at least with the neighbors of the client. We will find the required condition to propagate through the whole network. We first investigate the propagation decision by comparing the change in the expected rewards immediately after the action. Then, we extend our analysis with a permanence condition which guarantees that the ones who propagate will not suffer from any future actions.

We show that the sharing decision of a node is independent of the probability of his neighboring nodes being the round leader. Instead, it depends on his own probability against the rest who knows the transaction.

Lemma 2.3 (Equity Lemma)**.** Propagation decision of a node is independent from the neighbors' capacities. A rational node would propagate to either all of its neighbors or none of them.

Proof. Let a transaction T with fee F is known by a node n, and its distance to the c_T is k. The expected reward of node n can be defined as a function $R(\cdot)$ whose input corresponds to the capacities of the nodes who received T from n, then

$$R(X) = \frac{f^{k}_{[k]} \cdot \pi(n) + f^{k+1}_{[k]} \cdot X}{\pi(\mathcal{N}^{n,T}_{K}) + X}.$$

We show that $R(\cdot)$ is a monotone function. In order to show that a function is a monotone, it is enough to show that the sign of its derivative does not change in the domain range. For our case, it can be seen that the sign is independent of the input:

$$\begin{aligned} R'(X) &= \frac{f^{k+1}_{[k]}\left(\pi(\mathcal{N}^{n,T}_{K}) + X\right) - \left(f^{k}_{[k]}\pi(n) + f^{k+1}_{[k]}X\right)}{\left(\pi(\mathcal{N}^{n,T}_{K}) + X\right)^2} \\ &= \frac{f^{k+1}_{[k]}\pi(\mathcal{N}^{n,T}_{K}) - f^{k}_{[k]}\pi(n)}{\left(\pi(\mathcal{N}^{n,T}_{K}) + X\right)^2}. \end{aligned}$$

Since $R(\cdot)$ is a monotone function, then it achieves the maximum value at one of the boundary values. In our case, the boundary values are $X = 0$ where no neighbors received the transaction and $X = \pi\left(\mathcal{N}^{n,T}_{NK}\right)$ where all neighbors received it. Here, we omit the fact that $\pi(\cdot)$ is also a monotone function. Thus, we can say that a rational node maximizes his profit by propagating to either all of its neighbors or none of them. □

Lemma 2.3 simplifies to evaluate interfering multiple node decisions which is discussed in the following Lemma.

Lemma 2.4 (Propagation Lemma)**.** Let a node $n \in \mathcal{N}^{T}_{K}$, $\mathcal{N}^{n,T}_{NK} \neq \emptyset$ where the distance between n and c_T is k. All neighbors of n will be aware of T if

$$\frac{f^{k+1}_{[k]}}{f^{k}_{[k]}} > \frac{\pi(n)}{\pi(\mathcal{N}^{n,T}_{K})}.$$

Proof. Assume that some of the neighbors of n are not aware of T, i.e., $\mathcal{N}^{n,T}_{NK} \neq \emptyset$. From Lemma 2.3, we know that n did not propagate the transaction to any of his neighbors.

Table 2.1: The expected reward of n from T regarding possible decisions of n and the rest of $\mathcal{N}_K^{n,T}$.

		$\mathcal{N}_K^{n,T}$ (excluding n)	
	Decision	Not Propagate	(some) Propagate
n	Not Propagate	$\frac{f_{[k]}^{k}\cdot\pi(n)}{\pi(\mathcal{N}_K^{n,T})}$	$\frac{f_{[k]}^{k}\cdot\pi(n)}{\pi(\mathcal{N}_K^{n,T})+\pi(CN)+\pi(NCN_2)}$
	Propagate	$\frac{f_{[k]}^{k}\cdot\pi(n)+f_{[k]}^{k+1}\cdot\pi(\mathcal{N}_{NK}^{n,T})}{\pi(\mathcal{N}_K^{n,T})+\pi(\mathcal{N}_{NK}^{n,T})}$	$\frac{f_{[k]}^{k}\cdot\pi(n)+f_{[k]}^{k+1}\cdot\pi(NCN_1)+\alpha f_{[k]}^{k+1}\cdot\pi(CN)}{\pi(\mathcal{N}_K^{n,T})+\pi(\mathcal{N}_{NK}^{n,T})+\pi(NCN_2)}$

Therefore, at the moment, the only way that n profits from T is being the round leader with a reward $f_{[k]}^{k}$.

Table 2.1 presents expected reward of n with respect to each possible action of n and $\mathcal{N}_K^{n,T}$. The propagation decision of $\mathcal{N}_K^{n,T}$ may not include all its members, thereby all possible decisions are taken into account. Here, CN corresponds to the common neighbors of n and $\mathcal{N}_K^{n,T}$, NCN_1 distinct neighbors of n and NCN_2 distinct neighbors of $\mathcal{N}_K^{n,T}$ (who decide to propagate), i.e., $CN \bigcup NCN_1 = \mathcal{N}_{NK}^{n,T}$. Since CN is received the transaction from both n and the rest of the $\mathcal{N}_K^{n,T}$, α represents the percentage of the ones in CN decided to continue with the one including n.

If all nodes of $\mathcal{N}_K^{n,T}$ decide not to propagate with their neighbors, then n will benefit from propagating T in the case of

$$\frac{f_{[k]}^{k}\cdot\pi(n)+f_{[k]}^{k+1}\cdot\pi(\mathcal{N}_{NK}^{n,T})}{\pi(\mathcal{N}_K^{n,T})+\pi(\mathcal{N}_{NK}^{n,T})} > \frac{f_{[k]}^{k}\cdot\pi(n)}{\pi(\mathcal{N}_K^{n,T})} \iff \frac{f_{[k]}^{k+1}}{f_{[k]}^{k}} > \frac{\pi(n)}{\pi(\mathcal{N}_K^{n,T})}.$$

If (some) nodes in $\mathcal{N}_K^{n,T}$ decide to propagate T, then n will benefit from propagating T in the case of

$$\frac{f_{[k]}^{k}\cdot\pi(n)+f_{[k]}^{k+1}\cdot\pi(NCN_1)+\alpha f_{[k]}^{k+1}\cdot\pi(CN)}{\pi(\mathcal{N}_K^{n,T})+\pi(\mathcal{N}_{NK}^{n,T})+\pi(NCN_2)} > \frac{f_{[k]}^{k}\cdot\pi(n)}{\pi(\mathcal{N}_K^{n,T})+\pi(CN)+\pi(NCN_2)}$$
$$\Longleftarrow \frac{f_{[k]}^{k+1}}{f_{[k]}^{k}} > \frac{\pi(n)}{\pi(\mathcal{N}_K^{n,T})+\pi(CN)+\pi(NCN_2)} \text{ and } NCN_1 \neq \emptyset.$$

Note that $NCN_1 = \emptyset$ means that all the neighbors of n are also neighbors of $\mathcal{N}_K^{n,T}$ who decide to propagate. In addition, the sufficiency condition is independent of α. Therefore, in any case, if $\frac{f_{[k]}^{k+1}}{f_{[k]}^{k}} > \frac{\pi(n)}{\pi(\mathcal{N}_K^{n,T})}$ is satisfied, then all neighbors of n will be aware of the transaction. □

Corollary 2.5. Let $f_{[k]}^{k+1} \geq C\cdot f_{[k]}^{k}$ for some constant $C \in (0,1)$. $\mathcal{N}_K^{T}$ will continue to expand until there is no more node $n \in \mathcal{N}_K^{T}$ having neighbors in $\mathcal{N}_{NK}^{T}$ and satisfying $\pi(n) < C\cdot\pi(\mathcal{N}_K^{n,T})$.

Remark I. Here, it is possible to define different C_k values for each distance k, i.e., $f_{[k]}^{k+1} \geq C_k \cdot f_{[k]}^{k}$. One might argue that, as the distance increases, it could be possible to find nodes satisfying $\frac{\pi(n)}{\pi(\mathcal{N}_K^{n,T})} < C_k$ for smaller C_k values. However, as seen in Section 2.6, this is not always the case. In addition, the intermediate node may not know the exact distance, thus using the same C value would make the decision simpler.
Remark II. Note that the propagation decision is based on $\mathcal{N}_K^{n,T}$ instead of $\mathcal{N}_K^{T}$ since the latter one may not be available. This could lead to better consequences for propagation because nodes may predict $\mathcal{N}_K^{T}$ greater than its actual size and decide accordingly. Nonetheless, a carefully chosen C value will lead the nodes to share it with an overwhelming probability.
Remark III. Being the round leader should be more appealing than being an intermediary node, thus the round leader would try to fulfill the round block capacity to maximize his profit. The system may not work at full capacity if the nodes gain the same reward from propagating instead of validating (as the round leader) transactions. In Corollary 2.5, the propagation condition is given as $f_{[k]}^{k+1} \geq C \cdot f_{[k]}^{k}$. We fix the condition in favor of the round leader:

$$\forall\, k, \quad f_{[k]}^{k+1} = C \cdot f_{[k]}^{k}. \tag{2.2}$$

Permanence condition. In the simultaneous move analysis, we investigated one step at a time, i.e., what will happen immediately after the decision of propagation. However, all possible future actions should be taken into account. For example, the sender of a transaction should consider the possibility of the further propagation done by the receiver. From Lemma 2.3, capacities of the neighboring nodes do not have any influence on the sharing decision. Unless the processing fee share decreases, which is caused by some possible future actions like increased path length, the same lemma will be satisfied. If the share of a propagating node is non-decreasing with respect to the path length, then the ones who propagate will not suffer from any future actions. This can be formulated as

$$\forall\, i < k, \quad f_{[i]}^{k} \geq f_{[i]}^{k+1}. \tag{2.3}$$

2.4.3. FEE SHARING FUNCTION

With the equations obtained from the required conditions, we can uniquely determine the fee sharing function and conclude Theorem 2.1. First, using permanence condition (2.3), Sybil-proofness condition (2.1), can be reduced to $f_{[k]}^{k} \geq f_{[k+1]}^{k+1} + f_{[k]}^{k+1}$:

$$\begin{aligned}
\forall\, k \geq 1,\ f_{[k]}^{k} \quad & \geq f_{[k+1]}^{k+1} + f_{[k]}^{k+1} \geq f_{[k+2]}^{k+2} + f_{[k+1]}^{k+2} + f_{[k]}^{k+1} \\
& \geq f_{[k+3]}^{k+3} + f_{[k+2]}^{k+3} + f_{[k+1]}^{k+2} + f_{[k]}^{k+1} \geq \cdots \\
\forall\, s \geq 1, \quad & \geq f_{[k+s]}^{k+s} + \sum_{i=0}^{s-1} f_{[k+i]}^{k+i+1} \geq f_{[k+s]}^{k+s} + \sum_{i=0}^{s-1} f_{[k+i]}^{k+s}.
\end{aligned}$$

Therefore, we can update the Sybil-proofness condition as:

$$\forall\, k \geq 1, \quad f_{[k]}^{k} \geq f_{[k+1]}^{k+1} + f_{[k]}^{k+1}. \tag{2.4}$$

Then, we can obtain the following equations:

$$\begin{aligned}
\text{Using (2.4)} \quad & & \sum_{i=1}^{k} f_{[i]}^{i} \geq \sum_{i=1}^{k} f_{[i+1]}^{i+1} + \sum_{i=1}^{k} f_{[i]}^{i+1} \\
& \Longrightarrow & F = f_{[1]}^{1} \geq f_{[k+1]}^{k+1} + \sum_{i=1}^{k} f_{[i]}^{i+1} \\
\text{Using (2.3)} \quad & \Longrightarrow & F \geq f_{[k+1]}^{k+1} + \sum_{i=1}^{k} f_{[i]}^{k+1} = F \\
& \Longrightarrow & f_{[i]}^{k} = f_{[i]}^{k+1} \text{ and } f_{[k]}^{k} = f_{[k+1]}^{k+1} + f_{[k]}^{k+1}. \qquad (2.5)
\end{aligned}$$

After all, we can finalize the fee sharing function which corresponds to Theorem 2.1. Using (2.2) and (2.5), the share of the round leader can be computed:

$$f_{[k]}^{k} = f_{[k-1]}^{k-1}(1-C) = \cdots = F \cdot (1-C)^{k-1}. \qquad (2.6)$$

Using (2.5) and (2.6), the share of an intermediary node can be computed:

$$\forall\, i < k, \quad f_{[i]}^{k} = f_{[i]}^{i+1} = F \cdot C(1-C)^{i-1}.$$

2.4.4. Discussion

Integration. Implementation of the incentive mechanism should take into account the security and efficiency concerns. The propagation path should be immutable in a way that an adversary cannot add or subtract any node neither in the propagation process nor after the block generation. At the same time, storage efficiency is also essential since these path logs are needed to be stored in the ledger by every node. Both existing incentive-compatible blockchain solutions [16, 17] adopted a signature chaining mechanism where each propagated message includes the public key of the receiver and signature of the sender. This protocol prevents any manipulation over the path and thereby secures the shares of each contributor. It requires additional storage which is the signatures of the contributors. Although signature chaining solution requires the knowledge of the public key of the receiver and stores signatures of each sender, it is generic and can be applied to any blockchain. In Section 2.6, we present a novel and storage-efficient solution which is feasible for *FLTB* blockchains. It is embedded into routing mechanism and does not require the knowledge of the public keys of the neighboring nodes.

Determining C parameter. C value plays an important role to make sure that there will be incentive to propagate a transaction for some nodes until it reaches to the whole network. On the one hand, as the choice for the C value increases, it will be easier to satisfy the propagation condition since there will be more chance to find nodes satisfying $\pi(n) < C \cdot \pi(\mathcal{N}_K^T)$. On the other hand, the higher C value, the lower fee remains for the rest of the propagation path. It significantly reduces the fee of the round leader, thereby the incentive. For these reasons, it is required to choose a moderate C value, e.g., a reasonable choice would be $C = \frac{2}{N_{con}}$ where N_{con} denotes default number of connections of a node. For example, in Bitcoin network where $N_{con} = 8$, nodes will propagate unless they assume that their mining power is greater than 25% of the ones having the transaction. Even at the

very beginning, at least N_{con} nodes have the transaction, $C = \frac{2}{N_{con}}$ setting would provide overwhelming probability to have nodes willing to propagate according to Corollary 2.5.
Client ($0-$capacity) nodes. The main goal of the propagation incentive mechanism is to make sure that the transactions are received by the nodes who are capable of validating transactions as well as creating blocks. For that reason, we mainly focused on the nodes having a capacity greater than zero, i.e., $\pi(\cdot) > 0$. Nevertheless, a client node can be seen as a potential capacity node because of the possible propagation of the client. Regarding Lemma 2.3 and permanence condition (2.3), a rational node, who decided to propagate, would benefit from propagating to the client nodes as well. At the same time, a client node will always benefit from propagating any transaction since otherwise it will not have any chance to gain a fee.
Decentralization effect. In the conventional permissionless blockchains, all rewards including block reward and transaction fees are given to the block owner. In other words, nodes have only one incentive to participate in the network: being round leader. The less chance individual nodes have to be the round leader, the more they are motivated to join into centralized forms (e.g. mining pools) [13, 28]. Conversely, the transaction fee is shared with all propagators nodes. In addition, since many transactions are included in a single block, aiming processing fees of (some) transactions has significantly more chance than being the round leader. Thereby, it is reasonable to conclude that incentive mechanism would have a positive impact on the decentralization of the permissionless blockchains.

2.5. ROUTING MECHANISM

As a non-hierarchical peer-to-peer network, the blockchain ledger is validated by all nodes (miners) individually. This requires broadcasting every data and blocks over the network since every node needs to keep a record of the chain to validate new blocks. In existing permissionless blockchains, every transaction is broadcast throughout the network by the client, then the new block including (some of) these is constructed and broadcast by the round leader. Hence, each transaction is broadcast at least twice. Even more (`inv`) messages are sent to check the awareness of the neighbors on the transaction.

In Nakamoto-like consensus protocols, the round leader is validated simultaneously with his proposed block where the redundant propagation of the client is inevitable. In *FLTB* protocols, on the other hand, it is possible to validate the round leader before the block is proposed. It enables to determine a direct route between each client and the round leader. Our routing mechanism in Algorithm 1 finds the shortest paths between clients and the round leader for each round. Instead of sending each transaction to all nodes in the network, it is relayed over the shortest path between the client and the leader. The distance between (almost) any two nodes in a connected graph is dramatically smaller than the size of the network [29]. This is equivalent to cost reduction from $O(N)$ to $O(\ln N)$ in a random network of size N [30, 31].

The treat model of routing mechanism we present in this section considers a malicious adversary rather than a rational one. In the routing mechanism, a malicious adversary may try to block or censor some of the transaction propagations.

Our protocol can be divided into two parts: *Recognition Phase* where the routes are determined and *Transaction Phase* where the transactions are propagated (see Figure 2.2).

Algorithm 1 The Routing Algorithm

```
Recognition Phase
Leader provides his credential $\mathscr{L}^r$ to his neighbors.
for Node $n_1$ to $n_N$ do
    if First time receiving $\mathscr{L}^r$ then
        Store ID of the sender (gradient) node $n_j$, i.e., $gn_i \leftarrow n_j$
        Propagate $\mathscr{L}^r$ to neighbors.
    end if
end for

Transaction Phase
Client provides transaction $T$ to his neighbors.
for Each node $n_i$ receiving $T$ do
    if First time receiving $T$ then
        Send it to the $gn_i$
    end if
end for
```

First, in the recognition phase, the round leader is recognized throughout the network and his credential is propagated with a standard gossip protocol. Each node n_i learns his closest node towards the round leader, *gradient node* (gn_i), who is the first node forwarding the credential. In the transaction phase, each client forwards his transaction to (some of) his neighbors. Then, each node, receiving a transaction for the first time, directly transmits to his gradient node. Here, the reason for clients to broadcast to more than one neighbor is that one path could yield a single point of failure. It could be caused by the nodes who fail or maliciously censor some of the transactions. As presented in the experimental results, forwarding transaction to a few of the neighbors (precisely N_{con}) is sufficient. Note that, the routing mechanism works under asynchronous network assumptions since a client does not have to wait for all nodes but N_{con} of his neighbors. Similarly, for an intermediary node, waiting for the first credential message is enough to propagate received transactions.

Locational privacy. There have been several papers investigating anonymity in the permissionless blockchain networks, especially for the Bitcoin network [32–34]. It is found out that matching public keys and IP addresses can be done by eavesdropping. In this manner, *FLTB*-based blockchains may expose to DoS (denial-of-service) attacks against to the round leader. We want to stress that our routing mechanism does not leak any more locational information about the position of the leader other than the original *FLTB* protocols do. It just takes advantage of the announcement of the leader which is done exactly in the same manner with the *FLTB* protocols. Therefore, our routing mechanism does not cause any additional vulnerabilities for DoS-like attacks against the round leader. Yet, it is possible to improve the locational privacy via anonymity phase where the message is first forwarded in a line of nodes, then diffused from there [35]. The extra cost of anonymity would be a few nodes on the line which is still proportional to the logarithmic size of the network.

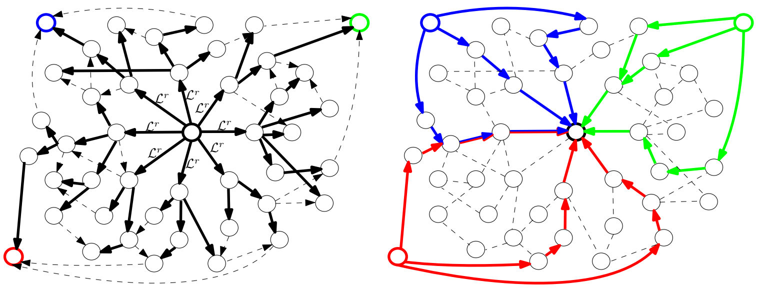


Figure 2.2: The Routing Mechanism. The left one illustrates the Recognition Phase and connections to the gradient nodes are shown with bold solid lines. On the right, three clients and their transaction paths are presented.

2.5.1. EXPERIMENTAL RESULTS

In this experiment, we use Barabási-Albert (BA) graph model [30] which simulates peer discovery in a peer-to-peer network. It starts with a well-connected small graph and each new node is connected to some of the previous nodes with a probability proportional to their degrees.

Barabási-Albert (BA) [30] and Erdős-Rényi (ER) [31] graph models have been used to simulate permissionless blockchains [36, 37]. In our setting, we combine both models where the network starts with a small ER graph and grows according to BA model. We start with 50 nodes in ER model [31] with edge probability of 1/2, meaning that on average each node has 25 connections. Then, each new node is added by connecting with N_{con} nodes in the network. For each (N, N_{con}) pair analyzed, we generated various graphs using Python graph library [38].

Bandwidth gain. In [39], the average shortest path length between any two nodes, i.e., the average path length, of a BA graph is shown to be in the order of $\frac{\ln N}{\ln\ln N}$. Hence, our routing protocol reduces the communication cost of a message transaction from $O(N)$ to $O(N_{con} \cdot \frac{\ln N}{\ln\ln N})$. The communication gain is up to 99% for scaled networks (see Figure 2.3), which can be verified by counting the average number of nodes visited per transaction. Here, we assume that the first arriving credential is coming from the node which is closest to the leader with respect to the number of nodes in between. In other words, the delay between any two nodes is computed by the node-distance.

In Figure 2.3, we count only one redundant communication for each transaction. Even more redundancy is caused by the flooding of each transaction because the same transaction is received from different neighboring nodes. In other words, the total redundancy is not N, but on average $N_{con} \cdot N$. In the existing blockchains, this additional redundancy is reduced by the sending the hash of the transaction to check whether the neighbor has it or not. If storage size of the transaction is relative to the size of the hash, then the total number of relays of a transaction would be significantly more than double of the network size. For example, Statoshi info [40], a block explorer of Bitcoin, shows that average incoming bandwidth usage for the transactions (`tx`) is, 2.87 KBps, less than for the checking messages (`inv`), 4.12 KBps (measurements taken between 02:00 AM and 14:00 PM in 13 of Feb. 2018). To conclude, since our mechanism does not suffer from the flooding effect, the actual communication gain would be much higher than the result in Figure 2.3.

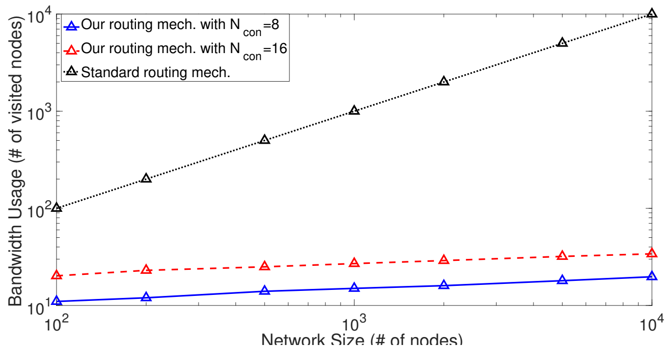


Figure 2.3: Communication cost for advertisement of a transaction.

Failing transmissions. Since each transaction is propagated among a small set of nodes, we need to take into account the possibility of propagation failure which can be caused by the nodes who fail or censor the transaction. The failure probability of a transaction can be approximated by $\left(1-(1-h)^{\frac{\ln N}{\ln\ln N}-1}\right)^{N_{con}}$ where h denotes the probability of a node in the network who fails or censors the transaction. These failing nodes are the ones who were present at the recognition phase and failed just afterwards. Long-term offline nodes can be ignored since they will not be chosen as gradient nodes. Thus, Figure 2.4 demonstrates that our routing is robust against instant network fluctuations. For a blockchain network with $N = 10000$ and $N_{con} = 8$, similar to Bitcoin network, if 30% of the active nodes fail after the recognition phase, only 9% of the transactions will be affected.

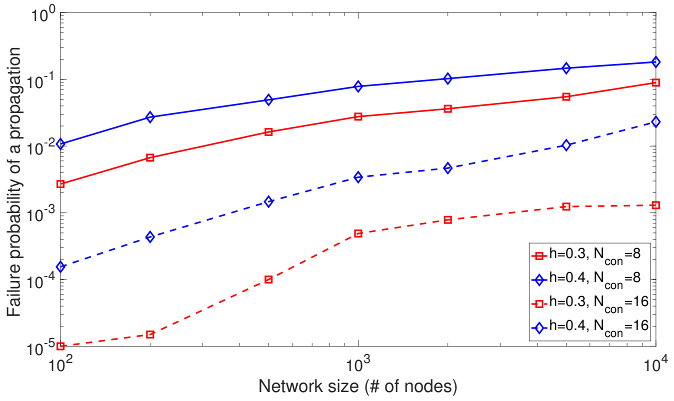


Figure 2.4: Probability of a transaction failing to be received by the round leader where h is the probability of an intermediary node being a failing or censoring node.

2.6. COMBINED PROPAGATION MECHANISM

In this section, we show how to deploy both of the incentive and routing mechanisms for any blockchain having a *FLTB* consensus protocol. At first glance, they seem to conflict with each other because the incentive mechanism is used to encourage propagation while the routing mechanism helps to reduce redundant propagation. We combine them in a way that rational nodes are encouraged to propagate only the transactions which are coming from the predefined paths of the routing mechanism. As demonstrated in Algorithm 2, we use the same infrastructure with the routing mechanism, and we include proofs of the intermediary nodes such that their contributions cannot be denied. Each transaction path is defined and secured by a path identifier which includes the public keys of the propagating nodes. Blocks consist of transactions as well as their path identifiers used to claim processing fee shares.

In the recognition phase, each intermediary node conveys the leader credential and the path identifier. Incoming and outgoing path identifiers of a node n are denoted by IN_n and OUT_n, which are used to validate and secure the propagation path. The round leader ℓ produces the initial identifier, $OUT_\ell = H(\mathscr{L}^r, PK_\ell)$, and propagates to his neighbors. Each node n updates the identifier coming from the gradient node by $OUT_n = H(IN_n, PK_n)$. This operation is done just for the gradient node (first one sending $\mathscr{L}^r$), then updated identifier and the credential are forwarded to the neighbors. Nodes

Algorithm 2 The Combined Propagation Algorithm

```
Recognition Phase
Leader $l$ propagates $\mathscr{L}^r$
for Each node $n_i$ do
    if First time receiving $\mathscr{L}^r$ and $IN_{n'}$ then
        if $\mathscr{L}^r$ is valid then
            Assign $IN_{n_i} \leftarrow IN_{n'}$ and gradient node as $n'$
            Compute $OUT_{n_i} = H(IN_{n_i}, PK_{n_i})$
            Propagate $\mathscr{L}^r$ and $OUT_{n_i}$ to neighbors.
        end if
    end if
end for

Transaction Phase
Client $c_T$ provides $Signed(T, IN_{c_T})$ (and $\mathscr{PK} = \emptyset$) to the first $N_{con}$ gradient nodes.
for Each node $n_i$ receiving $Signed(T, IN_{c_T})$ and $\mathscr{PK}$ do
    if First time receiving $T$ then
        if Signature path holds then
            Update $\mathscr{PK} \leftarrow \mathscr{PK} \bigcup \{PK_{n_i}\}$
            Send $Signed(T, IN_{c_T})$ and $\mathscr{PK}$ to the gradient node.
        end if
    end if
end for
```

may ignore the subsequent identifiers except a client who stores the first N_{con} ones for the transaction phase.

After the routing paths are determined, each client delivers the signed transaction and the incoming identifier to his N_{con} neighbors. The first receiving nodes, check the signature, then add their public keys to the transaction and forward it to their gradient nodes. From that point, each intermediary node in the path first checks the validity of the path via the public keys included and his own identifier, then forwards the transaction including his public key to the gradient node.

Once transactions are received by the round leader, he includes the valid ones into the block. The block consists of the credential, hash of the previous block and valid transactions with their paths. Then, the block is propagated throughout the network.

Incentive for block propagation. As a consequence of the incentive and routing mechanisms, intermediary nodes also have incentives to propagate the block since they share processing fees. Even more, the ones who are closer to the leader would have higher motivation since they probably gain from more transactions.

Storage efficiency. Any propagation incentive mechanism requires additional data storage than the data itself to keep track of the propagation path. Previous works having incentive [16, 17] utilize signature chains where each node signs the transaction and the public key of the receiver. Therefore, additional to the transaction, the signature package of each propagating node is included. On the other hand, our solution with the path identification benefits from the recognition phase of the routing protocol, and its additional storage requirement is only the public keys of propagating nodes and a signature of the client. Since the ability to claim propagation reward and the validation of the path need to be available, our propagation mechanism demands minimal storage components.

Privacy of the intermediary nodes. Signature chains and the proposed path identifier yield a direct connection between nodes network ID and their public keys. Unlike signature chains, our solution consists of two phases and the propagating nodes validate it by checking whether their input is preserved or not. This enables us to tackle the privacy issue by replacing plain public keys with commitments. Instead of directly including a public key, each node can obscure it in a simple commitment with a random number ($CT_i = H(PK_i, R_i)$). All verifications can be handled with the commitments while claiming propagation reward requires to reveal it. The commitment version uses the same network structure without compromising the identities of the nodes except clients and the round leader. The location of the round leader and clients will be known to their neighbors. They may need to update their key pairs or replace their connections for the next rounds.

2.7. Conclusion

In this work, we investigated two transaction propagation related problems of blockchains: incentive and bandwidth efficiency. We presented an incentive mechanism encouraging nodes to propagate messages, and a routing mechanism reducing the redundant communication cost.

We analyzed the necessary and sufficient conditions providing an incentive to propagate messages as well as to deviate participants (nodes) from introducing Sybil nodes. We studied different types of network topologies and we showed the impossibility result of the Sybil-proofness for the 1-connected model. We formulated the incentive-compatible propagation mechanism and proved that it obeys the rational behavior.

We presented a new aspect of the consensus algorithms, namely first-leader-then-block protocols. We proposed a smart routing mechanism for these protocols, which reduces the redundant transaction propagation from the size of the network to the scale of average shortest path length. Finally, we combined incentive and routing mechanisms in a compatible and memory-efficient way.

Future work and open questions. In Section 2.4.4, we mentioned the parameter choice and possible outcomes of the incentive mechanism. Detailed effect of incentive model and parameter choice are left as a future work. Another open question is the effect of the incentive mechanism on the topology of the network. Nodes would benefit from increasing their connection to contribute more transaction propagations, i.e., it would increase the connectivity of the network. Using that result, a rigorous analysis on the choice of the C parameter can be done. Finally, there are open problems regarding the economics of the transaction fee: analyzing the accuracy of the de facto formulas in the existing cryptocurrencies with respect to the cost of the propagation and validation and investigating the possible impacts of the sharing fee like decentralization effect.

REFERENCES

[1] G. W. Peters and E. Panayi, *Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money,* in *Banking Beyond Banks and Money* (Springer, 2016) pp. 239–278.

[2] Y. Sompolinsky and A. Zohar, *Bitcoin's underlying incentives,* Commun. ACM **61**, 46 (2018).

[3] I. Eyal and E. G. Sirer, *Majority is not enough: Bitcoin mining is vulnerable,* in [41], pp. 436–454.

[4] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, *Optimal selfish mining strategies in bitcoin,* in *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, Lecture Notes in Computer Science, Vol. 9603, edited by J. Grossklags and B. Preneel (Springer, 2016) pp. 515–532.

[5] K. Nayak, S. Kumar, A. Miller, and E. Shi, *Stubborn mining: Generalizing selfish mining and combining with an eclipse attack,* in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016* (IEEE, 2016) pp. 305–320.

[6] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, *On the security and performance of proof of work blockchains,* in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, edited by E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi (ACM, 2016) pp. 3–16.

[7] M. Rosenfeld, *Analysis of bitcoin pooled mining reward systems,* CoRR **abs/1112.4980** (2011), arXiv:1112.4980 .

[8] N. T. Courtois and L. Bahack, *On subversive miner strategies and block withholding attack in bitcoin digital currency,* CoRR **abs/1402.1718** (2014), arXiv:1402.1718 .

[9] I. Eyal, *The miner's dilemma,* in *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015* (IEEE Computer Society, 2015) pp. 89–103.

[10] S. Bag, S. Ruj, and K. Sakurai, *Bitcoin block withholding attack: Analysis and mitigation,* IEEE Trans. Information Forensics and Security **12**, 1967 (2017).

[11] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system,* (2008).

[12] G. Wood, *Ethereum: A secure decentralised generalised transaction ledger,* Ethereum Project Yellow Paper **151** (2014).

[13] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, *Is bitcoin a decentralized currency?* IEEE Security & Privacy **12**, 54 (2014).

[14] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, *Discovering bitcoin's public topology and influential nodes,* (May 2015).

[15] A. E. Gencer, S. Basu, I. Eyal, R. van Renesse, and E. G. Sirer, *Decentralization in bitcoin and ethereum networks,* CoRR **abs/1801.03998** (2018), arXiv:1801.03998 .

[16] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, *On bitcoin and red balloons,* in [42], pp. 56–73.

[17] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, *Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus,* CoRR **abs/1612.02916** (2016), arXiv:1612.02916 .

[18] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, *Bitcoin-NG: A scalable blockchain protocol,* in *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, edited by K. J. Argyraki and R. Isaacs (USENIX Association, 2016) pp. 45–59.

[19] I. Bentov, A. Gabizon, and A. Mizrahi, *Cryptocurrencies without proof of work,* in *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, Lecture Notes in Computer Science, Vol. 9604, edited by J. Clark, S. Meiklejohn, P. Y. A. Ryan, D. S. Wallach, M. Brenner, and K. Rohloff (Springer, 2016) pp. 142–157.

[20] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, *Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract],* SIGMETRICS Performance Evaluation Review **42**, 34 (2014).

[21] A. Kiayias, A. Russell, B. David, and R. Oliynykov, *Ouroboros: A provably secure proof-of-stake blockchain protocol,* in *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, Lecture Notes in Computer Science, Vol. 10401, edited by J. Katz and H. Shacham (Springer, 2017) pp. 357–388.

[22] F. Drucker and L. Fleischer, *Simpler sybil-proof mechanisms for multi-level marketing,* in [42], pp. 441–458.

[23] Y. Emek, R. Karidi, M. Tennenholtz, and A. Zohar, *Mechanisms for multi-level marketing,* in *Proceedings 12th ACM Conference on Electronic Commerce (EC-2011), San Jose, CA, USA, June 5-9, 2011*, edited by Y. Shoham, Y. Chen, and T. Roughgarden (ACM, 2011) pp. 209–218.

[24] J. M. Kleinberg and P. Raghavan, *Query incentive networks,* in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings* (IEEE Computer Society, 2005) pp. 132–141.

[25] C. Li, B. Yu, and K. P. Sycara, *An incentive mechanism for message relaying in unstructured peer-to-peer systems,* Electronic Commerce Research and Applications **8**, 315 (2009).

[26] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, *Eclipse attacks on bitcoin's peer-to-peer network,* in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, edited by J. Jung and T. Holz (USENIX Association, 2015) pp. 129–144.

[27] M. Apostolaki, A. Zohar, and L. Vanbever, *Hijacking bitcoin: Routing attacks on cryptocurrencies,* in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017* (IEEE Computer Society, 2017) pp. 375–392.

[28] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein, *Bitcoin mining pools: A cooperative game theoretic analysis,* in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, edited by G. Weiss, P. Yolum, R. H. Bordini, and E. Elkind (ACM, 2015) pp. 919–927.

[29] J. Travers and S. Milgram, *The small world problem,* Phychology Today **1**, 61 (1967).

[30] R. Albert and A.-L. Barabási, *Statistical mechanics of complex networks,* Reviews of Modern Physics **74**, 47 (2002).

[31] P. Erdös and A. Rényi, *On the evolution of random graphs,* Publ. Math. Inst. Hung. Acad. Sci **5**, 17 (1960).

[32] A. Biryukov, D. Khovratovich, and I. Pustogarov, *Deanonymisation of clients in bitcoin P2P network,* in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, edited by G. Ahn, M. Yung, and N. Li (ACM, 2014) pp. 15–29.

[33] G. C. Fanti and P. Viswanath, *Deanonymization in the bitcoin P2P network,* in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, edited by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett (2017) pp. 1364–1373.

[34] P. Koshy, D. Koshy, and P. D. McDaniel, *An analysis of anonymity in bitcoin using P2P network traffic,* in [41], pp. 469–485.

[35] S. B. Venkatakrishnan, G. C. Fanti, and P. Viswanath, *Dandelion: Redesigning the bitcoin network for anonymity,* POMACS **1**, 22:1 (2017).

[36] T. Neudecker, P. Andelfinger, and H. Hartenstein, *Timing analysis for inferring the topology of the bitcoin peer-to-peer network,* in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), Toulouse, France, July 18-21, 2016* (IEEE Computer Society, 2016) pp. 358–367.

[37] M. Berini Sarrias, *Bitcoin Network Simulator Data Explotation,* Master's thesis, Universitat Oberta de Catalunya (2015).

[38] T. Nepusz, *IGraph: High performance graph data structures and algorithms,* (2006–).

[39] A. Fronczak, P. Fronczak, and J. A. Hołyst, *Average path length in random networks,* Physical Review E **70**, 056110 (2004).

[40] Statoshi Team, *Statoshi info,* `http://statoshi.info` (retreived 13 Feb. 2018).

[41] N. Christin and R. Safavi-Naini, eds., *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, Lecture Notes in Computer Science, Vol. 8437 (Springer, 2014).

[42] B. Faltings, K. Leyton-Brown, and P. Ipeirotis, eds., *ACM Conference on Electronic Commerce, EC '12, Valencia, Spain, June 4-8, 2012* (ACM, 2012).

3

How to Profit from Payment Channels

Payment channel networks like Bitcoin's Lightning network are an auspicious approach for realizing high transaction throughput and almost-instant confirmations in blockchain networks. However, the ability to successfully conduct payments in such networks relies on the willingness of participants to lock collateral in the network. In Lightning, the key financial incentive to lock collateral are low fees for routing payments of other participants. While users can choose these fees, real-world data indicates that they mainly stick to default fees. By providing insights on beneficial choices for fees, we aim to incentivize users to lock more collateral and improve the effectiveness of the network.

In this paper, we consider a node **A** *that given the network topology and the channel details establishes channels and chooses fees to maximize its financial gain. Our contributions are i) formalization of the optimization problem, ii) proving that the problem is NP-hard, and iii) designing and evaluating a greedy algorithm to approximate the optimal solution. In each step, our greedy algorithm establishes a channel that maximizes the increase to* **A***'s total reward, which corresponds to maximizing the number of shortest paths passing through* **A**. *Our simulation study leveraged real-world data sets to quantify the impact of our gain optimization and indicates that our strategy is at least a factor two better than other strategies.*

This chapter is based on the paper "*How to Profit from Payments Channels*" by Ersoy, O., Roos, S., and Erkin, Z. in Financial Cryptography 2020, pp. 284–303.

3.1. Introduction

Payment channel networks [1] overcome the need to globally agree on every transaction in a blockchain. Instead, nodes can open and close *channels* that they can use to transfer coins directly. In the absence of disputes, transactions only require local communication between the parties involved in a transaction. Nodes without a direct payment channel can route payments via intermediaries to avoid the transaction fees and delays of channel opening. Thus, by moving transactions off-chain, payment channels have the potential to drastically increase the transaction throughput while reducing the confirmation times from tens of minutes to sub-seconds. The most notable examples of payment channel networks are Bitcoin's Lightning [2] and Ethereum's Raiden [3].

When opening a payment channel, nodes need to lock coins that they cannot use outside of the channel during the lifetime of the channel. This opportunity cost makes it unattractive to maintain payment channels. However, routing payments in a network requires that the network has well-funded channels [1]. The key incentives for locking collateral in a channel are i) frequent transaction with the other party [4] and ii) financial gain through routing fees [5], i.e., fees that nodes charge for routing payments as intermediaries. Our analysis of the Lightning network shows that the fees charged for routing are currently low and mainly equal to the default value [6]. We conjecture that the current payment channel networks primarily rely on the first incentive. However, research on the Lightning network suggests that this incentive entails networks of a low resilience with a few central hubs [7]. Analyzing the second incentives and show-casing that payment channels can entail financial profit is the most promising avenue of research to incentivize the participation in payment channel networks and fully leverage the potential of this promising blockchain scalability approach.

In this paper, we adapt a payment channel network (PCN) model based on Lightning. We assume a known topology and fees. Nodes select the cheapest path to conduct a payment. A node **A** aims to maximize its profit through routing fees by choosing both its payment channels and fees. The problem is challenging as higher fees indicate a higher profit if the node routes the payment but also a lower probability to be chosen for routing due to the transactions taking the cheapest path.

Despite the importance of fees in payment channel networks, the issue has been mainly ignored in past research. The majority of papers deal with cryptographic protocols for channel establishment and multi-hop payments (e.g., [4, 8–11]) as well as algorithms for routing payments (e.g., [12–14]). There is some work on comparing routing fees to the on-chain fees of blockchains and presenting an economical analysis of the relation between the two fee types [5, 15]. It is interesting to note that routing fees are related to the payment value whereas on-chain blockchain fees usually relate to the size of the transactions. In contrast, Di Stasi et al. [16] evaluated the impact of routing fees on keeping channels balanced, i.e., ensuring that a channel is not used exclusively in one direction. The authors suggest a novel linear fee policy for each channel to improve channel balances. Most similar to our work, Avarikioti et al. [17] studied the optimal fee assignment of channels from the point of view of a payment service provider (PSP). The authors analyzed optimal channel fees of the whole network that maximizes the total reward of the PSP instead of focusing on a node, which defines our problem. However, the authors can only solve for tree-structured networks, which does not make the approach useful in practice.

We are hence the first to cover the aspect of maximizing fees in payment channel networks. More precisely, we formalize the problem of maximizing fees in a Lightning-inspired system model. We present an algorithm for solving the defined optimization problem heuristically. Our greedy algorithm iteratively i) adds channels and ii) selects fees such that each added channel increases the profit maximally for the previously selected channels. For this purpose, we leverage the concept of (edge) betweenness centrality, i.e., the fraction of cheapest paths a vertex or edge is contained in. We evaluate our algorithm for real-world data sets of the Lightning network. Our evaluation strongly indicates that our approach does not only greatly improve the profit in comparison to default fees but also that leveraging betweenness centrality for selecting channels offers considerably better results than other network centrality measures. More preciously, our algorithm increases the profit by a factor 4 in comparison to default fee values and is at least a factor 2 better than other strategies. Our evaluation further demonstrates that nodes with already established channels can increase their profit by utilizing only our fee selection algorithm without establishing new channels.

3.2. BACKGROUND

This section summarizes key concepts from the field of payment channels. Furthermore, as our algorithm relies on graph centrality metrics, this section defines these metrics and gives some intuition on their role.

3.2.1. PAYMENT CHANNEL NETWORKS

Payment channel networks are one key approaches to scaling blockchains by moving transactions off-chain [1]. Two parties open a payment channel through an initial funding transaction on the blockchain that locks coins such that they can only be used for transactions between the two parties. After this initial funding transaction, the two parties can conduct payments without directly interacting with the blockchain. They commit to the latest balance of the channel, i.e., the distribution of the total number of locked coins over the two parties. For instance, let nodes u and v open a payment channel such that u locks x coins and v locks y coins. The initial *balance* of the channel is (x, y) and its total *capacity* is $x + y$. If u sends one coin to v, the balance changes to $(x-1, y+1)$.

In case of a dispute about the channel balance, the signed commitments documenting the state changes are published on the blockchain. The blockchain consensus then assigns the coins according to the latest valid channel state. Once the two parties decide to close their channel, they have to conduct a closing transaction on the blockchain. Afterward, they receive the coins locked in the channel with the number of coins per party corresponding to the channel balance at the time of the closure. In the absence of disputes, the intermediary transactions are almost instant and the number of transaction is merely bound locally by the bandwidth and latency of nodes.

Establishing a payment channel does not make sense if parties do not trade with each other regularly due to i) the on-chain fees for establishing the channel and ii) the opportunity cost caused by locking coins to the channel. Thus, most nodes will only establish a few channels with frequent trading partners. Routing payments via a path consisting of multiple channels nevertheless allows nodes to trade without having a direct channel. For instance, a node s can make a payment to a node r via two intermediary nodes u and v, meaning that the payment is routed via three payment channels: s to u, u to v, and v to r. The balances along all these channels change according to the transaction value.

The intermediary nodes charge fees for the use of their channels. For a channel Ch_i from u to v, these fees consist of a basic fee BF_{Ch_i} for using the channel and fee rate FR_{Ch_i} per transferred unit. The overall fee of a transaction tx for the channel is hence

$$\mathbf{f}(Ch_i, tx) = BF_{Ch_i} + FR_{Ch_i} \cdot |tx|, \tag{3.1}$$

where $|tx|$ denotes the transaction amount. The fees are determined by and paid to u. The sender s has to pay the fees. Note that the fee calculation formula given in Equation 3.1 is specific to the Lightning network [18]. Still, the other payment or state channel networks have a similar structure.

3.2.2. Graph Centrality Metrics

In this work, we model a PCN network as a directed graph. In this manner, each node in the payment channel represents a vertex in the graph and each channel is represented by two directional edges between the nodes (one for each direction). The channel fees correspond to the weights of the edges.

As a consequence, we can make use of graph metrics that characterize the importance of certain nodes in a weighted directed graph. Our key metrics are (vertex) betweenness centrality and edge betweenness centrality.

Definition 3.1 (Betweenness Centrality)**.** The betweenness centrality of a vertex [19] v is proportional to the total number of shortest paths that pass through that vertex, i.e.,

$$\mathbf{bc}(v) = \sum_{\substack{s\neq t\neq v \\ \sigma_{st}\neq 0}} \frac{\sigma_{stv}}{\sigma_{st}},$$

where σ_{st} denotes the number of shortest paths between s and t and σ_{stv} is the number of such shortest paths containing the vertex v.

Similarly, the edge betweenness centrality [20] of an edge relates to the total number of shortest paths that pass through that edge, i.e.,

$$\mathbf{e}([v_1 v_2]) = \sum_{\substack{s\neq t \\ \sigma_{st}\neq 0}} \frac{\sigma_{st[v_1 v_2]}}{\sigma_{st}},$$

where $\sigma_{st[v_1 v_2]}$ is the number of shortest paths passing through the edge $[v_1 v_2]$.

The analysis of this paper makes use of the following result about vertex betweenness centrality to assess the suitability of our greedy heuristic for selecting channel fees.

Theorem 3.2 ([21])**.** For each vertex v, betweenness centrality function $\mathbf{bc}(v)$ is a monotone function for the set of edges incident to v.

An important problem concerning the betweenness centrality is the *maximum betweenness improvement* (MBI) problem.

Definition 3.3 (MBI problem [21]). *Maximum Betweenness Improvement* problem: Given a directed graph G and a vertex v, find k edges incident to node v such that $\mathbf{bc}(v)$ is maximal.

With the help of the following theorem concerning the MBI problem, we prove that our problem of maximizing the reward (MRI) is NP-hard.

Theorem 3.4 ([21]). MBI problem cannot be approximated in polynomial time within a factor greater than $1 - \frac{1}{2\epsilon}$,unless $P = NP$.

3.3. OUR PCN MODEL

There are a number of PCNs with Lightning [2], Raiden [3], Perun [22] and Celer [23] being key examples. All of them use slightly different assumptions and properties. We base our system model on Bitcoin's Lightning network.

In the following, we first describe our PCN model **LN**. In this model, we then define the problem of an individual participant aiming to maximize their gain. We summarize the notation used in the paper in Table 3.1.

Table 3.1: Notation and Abbreviation Table

Symbol	Explanation
CSF	The channel selection function.
CFF	The channel fee function.
LN	The payment channel network.
$\mathbf{c}(X)$	The total amount of coins of X.
$\mathbf{f}(Ch, tx)$[1]	The charging fee of the channel Ch for a transaction of value tx.
$\mathbf{bc}(n, \mathbf{N})$[1]	The betweenness centrality of the node n in a network **N**.
$\mathbf{e}(Ch, \mathbf{N})$[1]	The edge betweenness centrality of the channel Ch in a network **N**.
$\mathbf{s}(Ch_i)$, $\mathbf{r}(Ch_i)$	The source and destination nodes of the channel Ch_i.
$ChCost$	The channel opening and closing on-chain cost.

3.3.1. NETWORK TOPOLOGY, FEES, AND ROUTING

Nodes open and close payment channels through blockchain transactions. For simplicity, we assume that the cost $ChCost$ of opening and closing remains constant over time.

In Lightning, the complete topology of the network is known to every node. Nodes publicly announce on the blockchain that they establish or close a channel. Furthermore,

[1]For brevity in the notation, tx and **N** can be omitted unless they alter with time.

nodes willing to route payments announce their channels and fees to the complete network. Thus, we assume in our model that both the topology and the fees of all nodes are publicly known. For simplicity, we assume that the topology and routing fees of the nodes that do not strategically change them remain fixed over time. Otherwise, our fee selection strategy would require a model to anticipate the expected changes. Current research on payment channel networks does not provide such a model. Our analysis of the Lightning network data from $1ml.com$ indicates that fees are indeed usually the default value. As topology changes require on-chain transactions, which are costly in both time and on-chain fees, the topology also should not change considerably. Moreover, we assume that nodes apply source routing to find one cheapest path from source to destination, as is the case in the current implementation of Lightning.

3.3.2. Problem Definition

We represent a network **LN** as a graph $G = (V, E)$ of vertices V and edges E. A node **A** aims to maximize its revenue in running a node in a payment channel network. For this purpose, **A** opens channels with other nodes in the network, each channel having a total cost of $ChCost$ for opening and closing. We assume that **A** can strategically select the nodes it establishes channels with from all nodes in the network. After all, these nodes do not need to invest anything into the channel as **A** completely funds them and they will likely receive additional monetary gains through routing fees. Furthermore, **A** has a budget of $\mathbf{c}(\mathbf{A})$ coins to use as collateral for the channels in total.

Formally, let C be the set of channels established by **A**. For each channel $Ch_i \in C$, we have the coins allocated to the channel $\mathbf{c}(Ch_i)$ and the channel fee $\mathbf{f}(Ch_i, tx)$ for a transaction value tx. Wlog, transaction values are integers between 1 and $\mathbf{T}_{max}$ following a distribution T. Let $X_i(tx, S, R)$ be the event that a transaction of value tx going from a node S to a node R passes through the channel Ch_i. Then the expected fee from that transaction is $\mathbf{f}(Ch_i, tx)Pr[X_i(tx, S, R)]$. Last, we require the distribution M that returns a sender-receiver pair. **A**'s objective is to find C, f, and $\mathbf{c}()$ such that the overall expected gain of one transaction

$$\sum_{\substack{\forall S,R \in V \\ S \neq R \neq \mathbf{A}}} Pr(M = (S,R)) \sum_{j=1}^{\mathbf{T}_{max}} Pr(T = j) \sum_{Ch_i \in C} \mathbf{f}(Ch_i, j) \cdot Pr[X_i(j, S, R)] \tag{3.2}$$

is maximized while adhering to the constraint that $\sum_{Ch_i \in C} \mathbf{c}(Ch_i) \leq \mathbf{c}(\mathbf{A})$. Equation 3.2 computes the expected gain over the involved variables T and M. If the capacity of the channel $\mathbf{c}(Ch_i)$ is less than the transaction amount tx, $Pr[X_i(tx, S, R)] = 0$. Similarly, if there does not exist a shortest path from S to R that passes through Ch_i, $Pr[X_i(tx, S, R)] = 0$. Otherwise, $Pr[X_i(tx, S, R)]$ is equal to the number of shortest paths from S to R passing through Ch_i divided by the total number of shortest paths from S to R.

Note that Equation 3.2 ignores the cost of opening C channels, $|C| \cdot ChCost$. The impact of this cost depends on the number of transactions K that occur during the lifetime of a channel. Let max be the maximal value for Equation 3.2. The overall gain of the node is then the difference: $K \cdot max - |C| \cdot ChCost$. By increasing the lifetime of the channel arbitrarily, the impact of $|C| \cdot ChCost$ diminishes, which is why we disregard it for Equation 3.2. Our model furthermore disregards the opportunity cost caused by locking coins due to the absence of suitable models for such a cost.

3.4. OUR FEE STRATEGY

We start by showing that maximizing the objective function given in Equation 3.2 is NP-hard. Afterwards, we present our greedy algorithm for approximating a solution. As our algorithm contains an equation for choosing channel fees without a closed-form solution, the last part of the section demonstrates a method for solving the equation numerically.

Our proof and algorithm act on a version of Equation 3.2 for specific distributions T and M. In the absence of real-world data for these distributions, we utilize two straightforward distributions. Concretely, our work considers a fixed transaction value, i.e., the random variable T only takes one value tx. For the distribution M, which characterizes the likelihood of two nodes to trade, assuming that all nodes are equally likely to trade with each other is the most natural choice in the absence of a concrete alternative model. Thus, M is a uniform distribution over all pairs of nodes in the following.

For the design of our algorithm, we furthermore bound the maximal channel fee by f_{max}. Assuming a maximal channel fee does not reduce the generality of our approach. As nodes send payments along the path with the lowest fee, any channel fee that entails the channel is not contained in any such path can be disregarded.

3.4.1. NP-HARDNESS OF THE PROBLEM

Before presenting the actual proof, we rephrase Equation 3.2 to relate it to the concept of (edge) betweenness centrality.

Choosing M to be a uniform distribution implies that $Pr(M = (S,R)) = \frac{1}{(|V|-1)(|V|-2)}$[2] is a constant, which can disregarded for the optimization. Furthermore, choosing a constant transaction value tx removes the second sum in Equation 3.2. Hence our modified objective function is

$$\sum_{Ch_i \in C} \mathbf{f}(Ch_i, tx) \cdot Pr[X_i(tx, S, R)]. \tag{3.3}$$

The next step relates $Pr[X_i(\text{tx}, S, R)]$ in Equation 3.3 to the betweenness centrality. There are two important quantities to consider: the number of shortest paths including the channel and total fee reward gained from these paths. Maximizing the number of shortest paths passing through a channel or node corresponds to the edge or vertex betweenness centrality (BC), respectively, as defined in Section 3.2. However, maximizing the BC does not necessarily imply maximal revenue. As fees represent edge weights, the shortest path here is a path whose edges have the minimal sum of weights. Choosing low fees hence increases the probability to be contained in the shortest path but low fees also indicate a low gain from each transaction.

Rather, the expected reward of a channel Ch_i is equal to the probability of the transaction passing through that channel times the fee. Note that each channel needs to have a capacity of at least tx for the payment to choose this path. Thus, an optimal solution for Equation 3.3 will only create channels of sufficient capacity and we can exclude the capacity aspect from $Pr[X_i(tx, S, R)]$. With $\mathbf{e}(Ch_i)$ denoting the edge betweenness centrality of

[2] $(|V|-1)(|V|-2)$ is the number of pairs of nodes when not including **A**

a channel Ch_i with fees $\mathbf{f}(Ch_i)$[3], the formal expression for the expected reward of Ch_i is

$$\mathsf{ER}(Ch_i) = \mathbf{f}(Ch_i) \cdot \mathbf{e}(Ch_i). \tag{3.4}$$

As a consequence, the total expected reward of **A** from Equation 3.3 is

$$\mathsf{ER}(\mathbf{A}) = \sum_{Ch_i \in C} \mathsf{ER}(Ch_i). \tag{3.5}$$

Now, we can formally define the problem from Equation 3.2 as the *maximum reward improvement* (MRI) problem.

Definition 3.5 (MRI Problem)**.** *Maximum Reward Improvement* problem: For a payment channel network $\mathscr{L}\mathscr{N}$ and a node n, find k channels incident to node n such that $\mathsf{ER}(n)$ is maximized.

The following theorem states that it is not possible to design an algorithm `CSF` that finds the optimum solution within polynomial time, unless $P = NP$.

Theorem 3.6 (MRI Approximation Theorem)**.** MRI problem cannot be approximated in polynomial time within a factor greater than $1 - \frac{1}{2\epsilon}$, unless $P = NP$.

Proof. To prove this theorem, we reduce our MRI problem to the MBI problem presented in Definition 3.3. Using Equation 3.5, we can formulate the MRI problem as follows:

$$\mathsf{MRI}(\mathbf{LN}, n, k) \to \mathscr{C}\mathscr{H}_M = \underset{\substack{|\mathscr{C}\mathscr{H}| \leq k \\ \mathbf{s}(Ch_i)=n \\ \mathbf{f}(Ch_i)\in[1,f_{max}]}}{\operatorname{argmax}} \left(\mathsf{ER}(n) = \sum_{Ch_i \in \mathscr{C}\mathscr{H}} \mathsf{ER}(Ch_i) \right).$$

We introduce a subproblem, namely MRI_FF, where the upper limit of the fee f_{max} is equal to 1, which means that all the channel fee are equal to 1. Using the Equation 3.4, MRI_FF can be formulated as:

$$\mathsf{MRI\_FF}(\mathbf{LN}, n, N_c) \to \mathscr{C}\mathscr{H}_M = \underset{\substack{|\mathscr{C}\mathscr{H}| \leq k \\ \mathbf{s}(Ch_i)=n}}{\operatorname{argmax}} \left(\sum_{Ch_i \in \mathscr{C}\mathscr{H}} \mathbf{e}(Ch_i) \right) \tag{3.6}$$

$$\overset{(*)}{=} \underset{\substack{|\mathscr{C}\mathscr{H}| \leq k \\ \mathbf{s}(Ch_i)||\mathbf{r}(Ch_i)=n}}{\operatorname{argmax}} \quad (bc_n) \overset{(**)}{=} \mathsf{MBI}(\mathbf{LN}, n, k),$$

which reduces to the MBI problem. Here, the first equality $(*)$ holds because the summation of the all shortest paths passing from out-going edges is equal to the total number of shortest paths passing through that node. In other words, the summation of edge betweenness centrality of all out-going edges of a node is equal to betweenness centrality of that node. The second equality $(*)$ follows from the definition of the MBI problem given in Definition 3.3.

[3]For the rest of section, we drop the transaction amount tx from the channel fee formula $\mathbf{f}(Ch_i)$ as it is fixed.

Now, we can prove our theorem by contradiction. Let assume there exists an approximation to MRI problem within a factor greater than $1-\frac{1}{2\epsilon}$. Then, the same approximation would hold for the subproblem of MRI, MRI_FF with a certain maximal fee, namely 1. However, in Equation 3.6, we showed that MRI_FF problem is equivalent to the MBI problem. This contradicts Theorem 3.4. Therefore, MRI problem cannot be polynomially approximated within a factor greater than $1-\frac{1}{2\epsilon}$, unless $P = NP$. $\square$

3.4.2. CHANNEL SELECTION FUNCTION

We present a *greedy* algorithm CSF to approximate the MRI problem. CSF takes the PCN and the requested number of channels as input and outputs the set of nodes to whom channels are created. It internally calls CFF, the algorithm for deciding the fee of a channel. Formally, we have

$$\begin{aligned}
&\mathtt{CFF}(\mathcal{CH} \cup Ch) \rightarrow R_{Ch}: \\
&R_{Ch} = \mathsf{TotalER}(\mathcal{CH} \cup Ch, f) \text{ where } f = \underset{f_i \in [1, f_{max}]}{\operatorname{argmax}} \left(\mathsf{TotalER}(\mathcal{CH} \cup Ch, f_i)\right), \\
&\mathsf{TotalER}(\mathcal{CH} \cup Ch, f_i) = \mathsf{ER}(Ch)_{\mathbf{f}(Ch)=f_i} + \sum_{Ch_j \in \mathcal{CH}} \mathsf{ER}(Ch_j).
\end{aligned} \tag{3.7}$$

As detailed in Algorithm 3, our greedy algorithm for CSF consists of the following five key steps:

1. Start with an initial PCN of nodes and channels.
2. At each step, try all possible channels between our node and other nodes.
3. Compute the maximum reward of the channel by using CFF.
4. Connect to the node who gives the maximum reward and update the PCN.
5. Go to step (2) until the desired number of channels is established.

Next, we ascertain that channel additions cannot reduce the expected revenue, indicating that nodes should add all channels they can fund. Here, it is important to note that we do not take into account the channel opening cost $ChCost$. Thus, if the marginal reward improvement of a new channel is zero, there is no point in add the channel.

Theorem 3.7 (Monotonicity)**.** The objective function of Algorithm 3 is a monotone non-decreasing function.

Proof. A function $\mathcal{F}: \Omega \rightarrow \mathbb{R}$ is a monotone function if it satisfies the following condition:

$$\forall S \subseteq T \subseteq \Omega, \quad \mathcal{F}(S) \leq \mathcal{F}(T). \tag{3.8}$$

In our case, we have to show that $\mathtt{CFF}(\mathcal{CH} \cup [n, n_i]) \geq \mathtt{CFF}(\mathcal{CH})$ for any solution $\mathcal{CH}$ and node n_i such that $[n, n_i] \notin \mathcal{CH}$ where $\mathcal{CH}$ is the current channel list of node n.

Note that CFF checks for all possible fee values to maximize the total reward. In that sense, it would be enough to show that for the maximum fee value f_{max}, which can be formulated by using Equation 3.7 (with $\mathbf{LN}_0 = \mathbf{LN} \cup \mathcal{CH}$ and $\mathbf{LN}_i = \mathbf{LN} \cup \mathcal{CH} \cup [n, n_i]$):

Algorithm 3 Channel Selection Function

```
Input: $\mathbf{LN}$ and $N_c$
Output: $\mathcal{CH}$
 1: function CSF($\mathbf{LN}, N_c$)
 2:     $\mathcal{CH} \leftarrow \emptyset$
 3:     while $|\mathcal{CH}| < N_c$ do
 4:         $maxRew \leftarrow 0, selectednode = None$
 5:         for Each node $n_i \in \mathbf{LN}$ do
 6:             Create a channel between $(n, n_i)$: $\mathbf{LN}_i \leftarrow AddEdges(\mathbf{LN}, [n, n_i])$
 7:             Calculate the reward $R_{n_i} \leftarrow \mathsf{CFF}(\mathbf{LN}_i, \mathcal{CH} \cup [n, n_i])$
 8:             if $maxRew \leq R_{n_i}$ then
 9:                 $maxRew = R_{n_i}$
10:                 $selectednode = n_i$
11:             end if
12:         end for
13:         $\mathcal{CH} \leftarrow \mathcal{CH} \bigcup \{selectednode\}$
14:         $\mathbf{LN} \leftarrow AddEdges(\mathbf{LN}, [n, selectednode])$
15:     end while
16:     return $\mathcal{CH}$
17: end function
```

$$\mathsf{CFF}(\mathbf{LN}, \mathcal{CH} \cup [n, n_i]) \geq \mathsf{TotalER}(\mathbf{LN}, \mathcal{CH} \cup [n, n_i], f = f_{max}) \overset{?}{\geq} \mathsf{CFF}(\mathbf{LN}, \mathcal{CH})$$

$$\iff \mathsf{ER}(Ch, \mathbf{LN}_i)_{f=f_{max}} + \sum_{\forall Ch_j \in \mathcal{CH}} \mathsf{ER}(Ch_j, \mathbf{LN}_i) \overset{?}{\geq} \sum_{\forall Ch_j \in \mathcal{CH}} \mathsf{ER}(Ch_j, \mathbf{LN}_0)$$

$$\iff \mathsf{ER}(Ch, \mathbf{LN}_i)_{f=f_{max}} \overset{?}{\geq} \sum_{\forall Ch_j \in \mathcal{CH}} \mathsf{ER}(Ch_j, \mathbf{LN}_0) - \mathsf{ER}(Ch_j, \mathbf{LN}_i)$$

$$\iff \mathbf{e}([n, n_i], \mathbf{LN}_i) \cdot f_{max} \overset{?}{\geq} \sum_{\forall Ch_j \in \mathcal{CH}} \big(\mathbf{e}(Ch_j, \mathbf{LN}_0) - \mathbf{e}(Ch_j, \mathbf{LN}_i)\big) \cdot \mathbf{f}(Ch_j)$$

$$\overset{(*)}{\Longleftarrow} \mathbf{e}([n, n_i], \mathbf{LN}_i) \overset{?}{\geq} \sum_{\forall Ch_j \in \mathcal{CH}} \big(\mathbf{e}(Ch_j, \mathbf{LN}_0) - \mathbf{e}(Ch_j, \mathbf{LN}_i)\big)$$

$$\iff \mathbf{e}([n, n_i], \mathbf{LN}_i) + \sum_{\forall Ch_j \in \mathcal{CH}} \mathbf{e}(Ch_j, \mathbf{LN}_i) \overset{?}{\geq} \sum_{\forall Ch_j \in \mathcal{CH}} \mathbf{e}(Ch_j, \mathbf{LN}_0)$$

$$\overset{(**)}{\iff} \mathbf{bc}(n, \mathbf{LN}_i) \overset{?}{\geq} \mathbf{bc}(n, \mathbf{LN}_0).$$

Here, $(*)$ condition is true since for all channels $\mathbf{f}(Ch_i) \leq f_{max}$ by the definition. Also, each term $\mathbf{e}(Ch_j, \mathbf{LN}_0) - \mathbf{e}(Ch_j, \mathbf{LN}_i)$ is non-negative as new channels of node n cannot increase the number of shortest paths passing through existing channels of the same node. Thus, the multiplication with a positive number preserves the inequality. $(**)$ is satisfied since the summation of edge betweenness centrality of all out-going edges of a node is equal to betweenness centrality of that node. At the end, $\mathbf{bc}(n, \mathbf{LN}_i) \geq \mathbf{bc}(n, \mathbf{LN}_0)$

holds because betweenness centrality is a monotone function, see Theorem 3.2. □

3.4.3. EFFICIENT SEARCH ALGORITHM FOR THE CHANNEL FEE FUNCTION

No closed-form formula finds the best fee amount maximizing the expected reward due to the term $\mathbf{e}(Ch)$ for a channel Ch. Here, we analyze Equation 3.4 to minimize the computational cost by discarding some parts of the search space. First of all, since $\mathbf{e}(\mathbf{LN})$ is not affected by changes to the fees of channels, the denominator is irrelevant for optimizing the $\mathsf{ER}(Ch)$. Therefore, `CFF` can be seen as a function of the edge betweenness centrality of the channel $\mathbf{e}(Ch)$ and its fee $\mathbf{f}(Ch)$. Secondly, $\mathbf{e}(Ch)$ is negatively affected by $\mathbf{f}(Ch)$ because increasing the fee means an increase in the weight of the edge that results in a lower chance of being in the shortest paths (see Figure 3.1 for an illustrative example).

Two observations give rise to an efficient search algorithm for finding the most suitable fee. The first observation utilizes the fact that edge betweenness centrality is a monotone decreasing function concerning the channel fee. Let the expected reward of a channel for chosen fees $f_3 > f_1$ be $r_1 = e_1 \cdot f_1$ and $r_3 = e_3 \cdot f_3$, respectively. If $r_3 > r_1$, let

$$f_2 = f_1 \cdot \frac{r_3}{r_1} = f_3 \cdot \frac{e_3}{e_1}. \tag{3.9}$$

It can be seen that the expected reward r_α for any fee f_α where $f_1 < f_\alpha \leq f_2$ is at most r_3:

$$r_\alpha = e_\alpha \cdot f_\alpha \leq e_1 \cdot f_\alpha \leq e_1 \cdot f_2 = e_3 \cdot f_3 = r_3. \tag{3.10}$$

In other words, there is no need to compute the expected reward values for the fees in between f_1 and f_2 as they cannot be optimal values.

The second observation is that increasing the fee of an out-going channel Ch cannot decrease the edge betweenness of another out-going channel Ch' of the same node. Such an increase can only reduce the edge betweenness of channels that are on a path containing Ch by removing the path from the set of shortest paths. However, as shortest

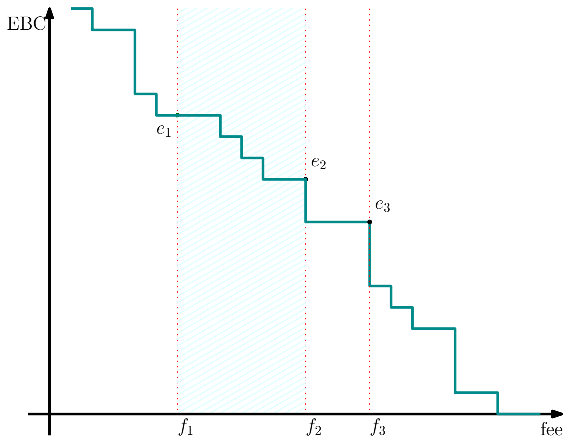


Figure 3.1: Illustrative example of the EBC vs. fee relationship of a channel.

paths cannot have loops, two out-going channels of the same node cannot be on the same shortest path. Now, let $\mathcal{CH}$ be the set of previously selected channels. Let r'_1 and r'_3 be the sum of the expected fees of all channels $Ch' \in \mathcal{CH}$ for fees f_1 and f_3 with $f_3 > f_1$. By the above observation, we have $r'_3 \geq r'_1$.

Our recursive algorithm divides the space of all possible fee values from 1 to f_{max} into d intervals. For each interval i, let $r_i = \mathsf{ER}(Ch, \mathbf{f}(Ch) = f_i)$ be the expected reward of Ch and $r'_i \leftarrow \sum_{Ch' \in \mathcal{CH}} \mathsf{ER}(Ch', \mathbf{f}(Ch'))$ be the total reward of the other channels. By the first observation, the maximal increase in r_i is $\frac{f_{i+1}}{f_i}$ and by the second observation $r'_{i+1} \geq r'_i$ as $f_{i+1} > f_i$. Thus, the maximum possible reward value for interval i is $\widetilde{R}_i = r_i \cdot \frac{f_{i+1}}{f_i} + r'_{i+1}$. If $\widetilde{R}_i$ is greater than the current maximum reward value, the algorithm recursively searches for a maximum in the interval, otherwise discards the interval. Algorithm 4 is the pseudocode of our recursive algorithm.

This completes the description of our algorithm, which we evaluate in the following in comparison to other approaches based on common centrality metrics.

3.5. Evaluation

In this section, we evaluate our proposed fee strategy for a real-world topology. Our evaluation quantifies the total reward gained by **A** when using our greedy algorithm.

To emphasize the high effectiveness of our solution, we compared it with other channel and fee selection algorithm. For the channel selection, we considered random nodes as well as connecting to nodes with a high centrality for three centrality metrics: i) degree, i.e., connecting to the nodes with the most connections, ii) betweenness centrality, and iii) pagerank [24]. For the fee strategy, we compute the results for both cases where the channel fees are the default values and they are determined by `CFF`.

3.5.1. Model

In Lightning Network, the upcoming transactions and current balances of channels are not known. Thus, we need to model the network and transactions.

Transactions. Like Section 3.4, our evaluation assumes that all source-destination pairs are equally likely. Furthermore, we categorize the transactions into three groups based on the amounts:

- *Micro payments* are the transactions involving a very small amount of coins. To represent this category, we use the transaction amount of 100 Satoshi, which is about one cent[4]. An example of a use case would be the streaming services where you pay small amounts per service.
- *Medium payments*: are the transactions spent for daily living expenses like buying a coffee, which is represented with 10000 Satoshi.
- *Macro payments*: are transactions of high amounts, which is represented with 1000000 Satoshi. The amount of these transactions are in the order of 100 Euros.

[4]https://awebanalysis.com/en/convert-satoshi-to-euro-eur/

Algorithm 4 Channel Fee Function

```
Input: LN, $\mathcal{CH}$ and $Ch$
Output: $R_{max}$ and $f_{max}$
function CFF(LN, $\mathcal{CH} \cup Ch, f_l, f_h$)
    % Initialization: $f_l \leftarrow 1, f_h \leftarrow ChCost, R_{max} \leftarrow 0, f_{max} \leftarrow 1$
    % d is the division parameter
    if $f_h - f_l \leq \mathsf{d}$ then % Anchor step:
        for $f \in \{f_l, \dots, f_h\}$ do
            $[r, r'] \leftarrow \mathsf{TotalER}(\mathcal{CH} \cup Ch, f)$
            Calculate the reward $R \leftarrow r + r'$
            if $R \geq R_{max}$ then
                $R_{max} \leftarrow R$
                $f_{max} \leftarrow f$
            end if
        end for
        return
    else % Recursion step:
        for $i \in \{1, \dots, \mathsf{d}\}$ do
            $f_i \leftarrow i \cdot \frac{f_h - f_l}{\mathsf{d}} + f_l$
        end for
        for $i \in \{1, \dots, \mathsf{d}\}$ do
            $[r_i, r'_i] \leftarrow \mathsf{TotalER}(\mathcal{CH} \cup Ch, f_i)$
            Calculate the reward $R_i = r_i + r'_i$
            if $R_i \geq R_{max}$ then
                $R_{max} \leftarrow R_i$
                $f_{max} \leftarrow f_i$
            end if
        end for
        for $i \in \{1, \dots, \mathsf{d}\}$ do
            Calculate the possible maximum reward $\widetilde{R}_i = r_i \cdot \frac{f_{i+1}}{f_i} + r'_{i+1}$
            if $\widetilde{R}_i > R_{max}$ then
                $f_l \leftarrow f_i, f_h \leftarrow f_{i+1}$
                return CFF(LN, $\mathcal{CH} \cup Ch, f_l, f_h$)
            else
                % Do nothing - Discard this interval
            end if
        end for
    end if
end function

function TotalER($\mathcal{CH} \cup Ch, f$)
    $r \leftarrow \mathsf{ER}(Ch, \mathbf{f}(Ch) = f)$
    $r' \leftarrow \sum_{\forall Ch_j \in \mathcal{CH}} \mathsf{ER}(Ch_j, \mathbf{f}(Ch_j))$
    return $[r, r']$
end function
```

From these categories, it is most likely that micro payments are usually restricted to nodes that have a direct channel. Otherwise, the base fee for the payment greatly exceeds the actual payment value. Therefore, our target transactions are medium and macro payments, which are analyzed separately.

Network. Following our system model in Section 3.3, networks are represented as weighted directed graphs. The weights of the edges in the graph model are calculated according to the fee rate and base fees of the channels. Since the fee rate depends on the transaction amount, the weights of the same edges for medium and macro payments will be different. The graph generated for the medium (macro) payments is called medium (macro) graph.

3.5.2. Setup

We obtained a snapshot of the Lightning Network (LN) data from $rompert.com$[5] on July 10 2019, which contains 4618 nodes and 68729 edges in total. When we delete the edges with insufficient capacity, the medium graph has 68697 edges and the macro graph has 32193 edges.

As a node requires at least two connections to be contained in any shortest paths, we first connected **A** to the two nodes with the highest degree (, which happen to have the highest pagerank as well). For these two connections, we use the default fee rate and base fee values in both directions of the edges. Based on this initial scenario, we now connect **A** to additional nodes.

The experiments use $ChCost = 8192$ Satoshi, which reflects the fluctuating Bitcoin transaction fee estimates[6]. When establishing a new channel, our simulation added edges in both directions. The base fee and the fee rate of the in-coming edge corresponded to the default value to model that i) most users currently stick to the default values and ii) **A** has no control over the in-coming channel fees as they are determined by the other party. For the outgoing edges, we utilize either `CFF` to determine the best fee value or use default values. When using `CFF`, we set $f_{max} = ChCost$. Otherwise, the total fee cost of the transaction in the payment network is higher than the cost in the Bitcoin network and the sender is hence unlikely to proceed with the payment.

3.5.3. Experimental Results

Figures 3.2 and 3.3 show the performance of our greedy algorithm in comparison to the other approaches in terms of the total reward improvement per new channel connections. The x-axis shows the number of connections added and the y-axis represents the total reward of node **A**. Since, for each case, we started with the same two connections, the total reward values have the same offset.

Figure 3.2 displays the result for the medium graph. When using default values, the reward was consistently lower than for our fee selection algorithm. More precisely, for centrality-based selection of channels, fee optimization increased the reward by a factor of roughly 2. Selecting channels strategically doubled the gain further in comparison

[5]The data source is corrected in the thesis.
[6]https://bitcoinfees.info/

to using Pagerank centrality, which was the most beneficial one of the centrality-based selection methods. Figure 3.3 shows the results of macro graph. The results were similar to the case of medium payments, though the overall gain was slightly higher.

In terms of fee computation efficiency, our experimental results show that the recursive algorithm described in Section 3.4.3 reduced the search space of fees in the magnitude of 10–100.

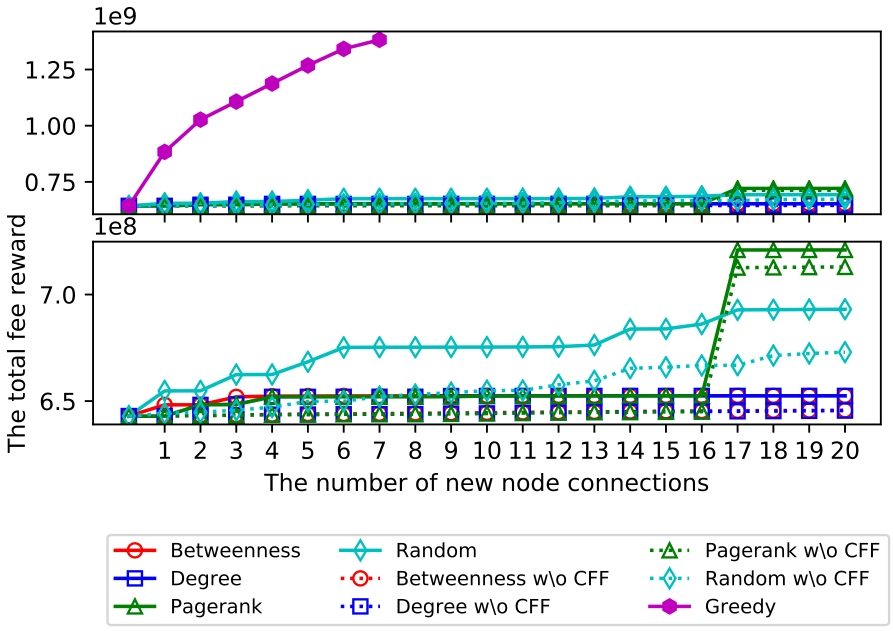


Figure 3.2: Total fee reward of our node in medium graph. The bottom figure excludes the greedy results to present a clear comparison of the rest.

3.5.4. DISCUSSION

From the experimental results, it can be seen that our greedy algorithm outperformed other centrality metrics. Furthermore, the beneficial effect of the fee selection function was evident when comparing the results with and without it.

Note that adding new connections to the nodes with the highest centrality metrics did not increase the total reward in comparison to random selection much, in particular for betweenness centrality. The reason here is that connecting to nodes with many shortest path passing them does not imply that the newly added channels offer shorter paths. Instead, directly focusing on the betweenness centrality of **A** results in larger improvements.

Figure 3.2 and 3.3 furthermore show few but notable differences between medium and macro payments. First, the overall gain was higher for macro payments as expected due to the higher transaction value and hence increased revenue for a similar fee rate.

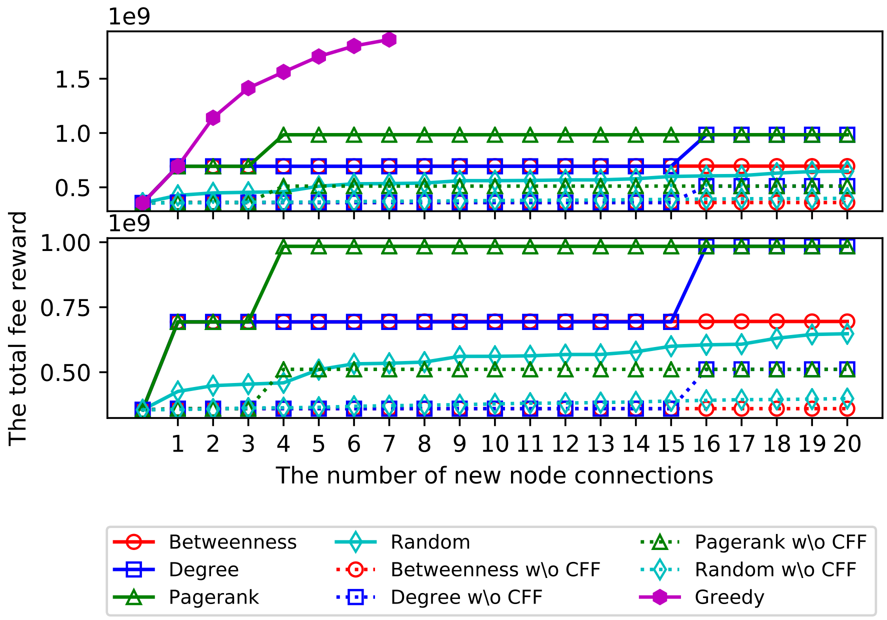


Figure 3.3: Total fee reward of our node in macro graph. The bottom figure excludes the greedy results to present a clear comparison of the rest.

However, the base rate, which is 1000 Satoshi by default[7] in comparison to a default rate of 0.001, dominates the fee value, so that the 100-fold increase in the transaction value does not translate to a similar increase in gain. Secondly, the differences between various centrality measures are more distinct for macro payments, see Figure 3.3.

Overall, our greedy algorithm promises higher fees for individual nodes. Even if nodes cannot or do not desire to select their channels, they can still gain an advantage by using our more sophisticated fee selection algorithm for already established channels.

One key limitation of our design is that it does not consider channel capacities as such. When all transactions have the same known value, **A** will only establish channels with sufficient collateral. However, in practice, **A** does not have such information and routing may fail due to a lack of capacity. Thus, integrating capacity information into both our model and our evaluation is clearly necessary in the future.

3.6. Conclusion

In this paper, we formalized an optimization problem for maximizing fees in payment channel networks, presented a heuristic algorithm for solving the problem, and evaluated our algorithm on real-world data sets. Our work demonstrates that routing fees can be a strong incentive for locking coins in payment channels. Fees as incentive hence have

[7]The default fee values may change regarding the imported implementation. Our analysis on dataset shows that 33177 out of 68733 edges use the defaults we adopted.

the potential to motivate rational users to fund payment channel and hence increase the ability of these networks to route payments.

In this work, we focused on one individual node aiming to optimize its profit. Future work should design a game-theoretical framework for networks containing only rational nodes aiming to maximize their profit. For the continued usage of payment channel networks, incentives should ensure that strategies for optimizing profit locally also optimize the overall network health in terms of the availability of cost-effective paths. It remains an open question if the current fee model is a suitable incentive to further collaboration and network health.

REFERENCES

[1] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, *Sok: Off the chain transactions.* IACR Cryptology ePrint Archive **2019**, 360 (2019).

[2] J. Poon and T. Dryja, *The bitcoin lightning network: scalable off-chain instant payments,* (2016), available at: https://lightning.network/lightning-network-paper.pdf.

[3] B. T. AG, *Raiden network,* (2019), available at: https://raiden.network/.

[4] C. Decker and R. Wattenhofer, *A fast and scalable payment network with bitcoin duplex micropayment channels,* in *Symposium on Self-Stabilizing Systems* (Springer, 2015).

[5] F. Engelmann, H. Kopp, F. Kargl, F. Glaser, and C. Weinhardt, *Towards an economic analysis of routing in payment channel networks,* in *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers, SERIAL@Middleware 2017* (ACM, 2017).

[6] E. Project, *Lightning-getroute,* (2019), available at: https://github.com/ElementsProject/lightning/blob/master/doc/lightning-getroute.7.

[7] E. Rohrer, J. Malliaris, and F. Tschorsch, *Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks,* in *Security & Privacy on the Blockchain* (2019).

[8] M. Hearn, *Micro-payment channels implementation now in bitcoinj,* (2013), available at: https://bitcointalk.org/index.php?topic=244656.0.

[9] C. Egger, P. Moreno-Sanchez, and M. Maffei, *Atomic multi-channel updates with constant collateral in bitcoin-compatible payment-channel networks,* in *CCS* (2019).

[10] C. Decker, R. Russell, and O. Osuntokun, *Eltoo: A simple layer2 protocol for bitcoin,* (2018), available at: https://blockstream.com/eltoo.pdf.

[11] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, *Sprites and state channels: Payment networks that go faster than lightning,* in *International Conference on Financial Cryptography and Data Security* (Springer, 2019).

[12] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, *Flare: An approach to routing in lightning network,* (2016), available at: https://bitfury.com/content/downloads/whitepaper_flare_an_approach_to_routing_in_lightning_network_7_7_2016.pdf.

[13] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, *Settling payments fast and private: Efficient decentralized routing for path-based transactions,* in *Network and Distributed Systems Security* (2018).

[14] P. Hoenisch and I. Weber, *Aodv–based routing for payment channel networks,* in *International Conference on Blockchain* (Springer, 2018).

[15] S. Brânzei, E. Segal-Halevi, and A. Zohar, *How to charge lightning,* CoRR **abs/1712.10222** (2017), arXiv:1712.10222 .

[16] G. D. Stasi, S. Avallone, R. Canonico, and G. Ventre, *Routing payments on the lightning network,* in *iThings/GreenCom/CPSCom/SmartData 2018* (2018).

[17] G. Avarikioti, G. Janssen, Y. Wang, and R. Wattenhofer, *Payment network design with fees,* in *ESORICS 2018 International Workshops, DPM 2018 and CBT 2018*, Lecture Notes in Computer Science, Vol. 11025, edited by J. García-Alfaro, J. Herrera-Joancomartí, G. Livraga, and R. Rios (Springer, 2018).

[18] *Basis of lightning technology,* Available at: `https://github.com/lightningnetwork/lightning-rfc/blob/master/00-introduction.md`.

[19] L. C. Freeman, *A set of measures of centrality based on betweenness,* Sociometry **40**, 35 (1977).

[20] M. Girvan and M. E. J. Newman, *Community structure in social and biological networks,* Proceedings of the National Academy of Sciences **99**, 7821 (2002).

[21] E. Bergamini, P. Crescenzi, G. D'Angelo, H. Meyerhenke, L. Severini, and Y. Velaj, *Improving the betweenness centrality of a node by adding links,* ACM Journal of Experimental Algorithmics **23** (2018), 10.1145/3166071.

[22] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, *Perun: Virtual payment channels over cryptographic currencies,* in *Symposium on Security and Privacy* (2019).

[23] M. Dong, Q. Liang, X. Li, and J. Liu, *Celer network: Bring internet scale to every blockchain,* arXiv preprint arXiv:1810.00037 (2018).

[24] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank citation ranking: Bringing order to the web.*, Tech. Rep. (Stanford InfoLab, 1999).

4

Bitcoin-Compatible Virtual Channels

Current permissionless cryptocurrencies such as Bitcoin suffer from a limited transaction rate and slow confirmation time, which hinders further adoption. Payment channels are one of the most promising solutions to address these problems, as they allow the parties of the channel to perform arbitrarily many payments in a peer-to-peer fashion while uploading only two transactions on the blockchain. This concept has been generalized into payment channel networks where a path of payment channels is used to settle the payment between two users that might not share a direct channel between them. However, this approach requires the active involvement of each user in the path, making the system less reliable (they might be offline), more expensive (they charge fees per payment), and slower (intermediaries need to be actively involved in the payment). To mitigate this issue, recent work has introduced the concept of virtual channels (IEEE S&P'19), which involve intermediaries only in the initial creation of a bridge between payer and payee, who can later on independently perform arbitrarily many off-chain transactions. Unfortunately, existing constructions are only available for Ethereum, as they rely on its account model and Turing-complete scripting language. The realization of virtual channels in other blockchain technologies with limited scripting capabilities, like Bitcoin, was so far considered an open challenge.

In this work, we present the first virtual channel protocols that are built on the UTXO-model and require a scripting language supporting only a digital signature scheme and a timelock functionality, being thus backward compatible with virtually every cryptocurrency, including Bitcoin. We formalize the security properties of virtual channels as an ideal functionality in the Universal Composability framework and prove that our protocol constitutes a secure realization thereof. We have prototyped and evaluated our protocol on the Bitcoin blockchain, demonstrating its efficiency: for n sequential payments, they

This chapter is based on the paper "*Bitcoin-Compatible Virtual Channels*" by Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P. and Riahi, S. in IEEE Symposium on Security and Privacy 2021 (in press).

require an off-chain exchange of $9+2\cdot n$ *transactions or a total of* $3524+695\cdot n$ *bytes, with no on-chain footprint in the optimistic case. This is a substantial improvement compared to routing payments in a payment channel network, which requires* $8\cdot n$ *transactions with a total of* $3026\cdot n$ *bytes to be exchanged.*

4.1. INTRODUCTION

Permissionless cryptocurrencies such as Bitcoin [1] have spurred increasing interest over the last years, putting forward a revolutionary, from both a technical and economical point of view, payment paradigm. Instead of relying on a central authority for transaction validation and accounting, Bitcoin relies on its core on a decentralized consensus protocol for these tasks. The consensus protocol establishes and maintains a distributed ledger that tracks every transaction, thereby enabling public verifiability. This approach, however, severely limits the transaction throughput and confirmation time, which in the case of Bitcoin is around ten transactions per second, and confirmation of an individual transaction can take up to 60 minutes. This is in stark contrast to central payment providers that offer instantaneous transaction confirmation and support orders of magnitude higher transaction throughput. These scalability issues hinder permissionless cryptocurrencies such as Bitcoin from serving a growing base of payments.

Within other research efforts [2–4], payment channels [5] have emerged as one of the most promising scalability solutions. The most prominent example that is currently deployed over Bitcoin is the so-called Lightning network [6], which at the time of writing hosts deposits worth more than 60M USD. A payment channel enables an arbitrary number of payments between users while committing only two transactions onto the blockchain. In a bit more detail, a payment channel between Alice and Bob is first created by a single on-chain transaction that deposits Bitcoins into a multi-signature address controlled by both users. The parties additionally ensure that they can get their Bitcoins back at a mutually agreed expiration time. They can then pay to each other (possibly many times) by exchanging authenticated off-chain messages that represent an update of their share of coins in the multi-signature address. The payment channel is finally closed when a user submits the last authenticated distribution of Bitcoins to the blockchain (or after the channel has expired).

Interestingly, it is possible to leverage a path of opened payment channels from the sender to the receiver with enough capacity to settle their payments off-chain, thereby creating a payment channel network (PCN) [6, 7]. Assume that Alice wants to pay Bob, and they do not have a payment channel between each other but rather are connected through an intermediary user Ingrid. Upon a successful off-chain update of the payment channel between Alice and Ingrid, the latter would update her payment channel with Bob to make the overall transaction effective. The key challenge is how to perform the sequence of updates atomically in order to prevent Ingrid from stealing the money from Alice without paying Bob. The standard technique for constructing PCNs requires the intermediary (e.g., Ingrid in the example from above) to be actively involved in each payment. This has multiple disadvantages, including (i) making the system less reliable (e.g., Ingrid might have to go offline), (ii) increasing the latency of each payment, (iii) augmenting its costs since each intermediary charges a fee per transaction, and (iv) revealing possibly sensitive payment information to the intermediaries [8–10].

An alternative approach for connecting multiple payment channels was introduced by Dziembowski et al. [11]. They propose the concept of *virtual channels* – an off-chain protocol that enables direct off-chain transactions without the involvement of the intermediary. Following our running example, a virtual channel can be created between Alice and Bob using their individual payment channels with Ingrid. Ingrid must collaborate with Alice and Bob only to create such virtual channel, which can then be used by Alice and Bob to perform arbitrarily many off-chain payments without involving Ingrid. Virtual channels offer strong security guarantees: each user does not lose money even if the others collude. A salient application of virtual payment channels is so-called payment hubs [11]. Since establishing a payment channel requires a deposit and active monitoring, the number of channels a user can establish is limited. With payment hubs [11], users have to establish just one payment channel with the hub and can then dynamically open and close virtual channels between each other on demand. Interestingly, since in a virtual channel the hub is not involved in the individual payments, even transactions worth fractions of cents can be carried out with low latency.

The design of secure virtual channels is very challenging since, as previously mentioned, it has to account for all possible compromise and collusion scenarios. For this purpose, existing virtual channel constructions [11] require smart contracts programmed over an expressive scripting language and the account model, as supported in Ethereum. This significantly simplifies the construction since the deposit of a channel, and its distribution between the end-points are stored in memory and can programmatically be updated. On the downside, however, these requirements currently limit the deployment of virtual channels to Ethereum.

It was an *open question* until now if virtual channels could be implemented at all in UTXO-based cryptocurrencies featuring only a limited scripting language, like Bitcoin and virtually all other permissionless cryptocurrencies. We believe that answering this question is important for several reasons. First, by limiting the trusted computing base (i.e., the scripting functionality supported by the underlying blockchain), we reduce the on-chain complexity of the virtual channel protocol. As bugs in smart contracts are manifold and notoriously hard to fix, our construction eliminates an additional attack vector by moving the complexity to the protocol level (rather than on-chain as in the construction from [11]). Second, investigating the minimal functionality that is required by the underlying ledger to support complex protocols is scientifically interesting. One may view this as a more general research direction of building a lambda calculus for off-chain protocols. Concretely, our construction shows that virtual channels can be built with stateless scripts, while earlier constructions required stateful on-chain computation. Finally, from a practical perspective, our construction can be integrated into the Lightning Network (the by far most prominent PCN), and thus our solution can offer the benefits of virtual payment channels/hubs to a broad user base.

4.1.1. Our Contributions

In this work, we develop the *first* protocols for building virtual channel hubs over cryptocurrencies that support limited scripting functionality. Our construction requires only digital signatures and timelocks, which are ubiquitously available in cryptocurrencies and well characterized. We also provide a comprehensive formal analysis of our constructions and benchmarks of a prototype implementation. Concretely, our contributions are summarized below.

- We present the first protocols for virtual channel hubs that are built for the UTXO-model and require a scripting language supporting only digital signature verification and timelock functionality, being thus compatible with virtually every cryptocurrency, including Bitcoin. Since in the Lightning network currently only 10 supernodes are involved in more than 25% of all channels, our technique can be used to reduce the load on these nodes, and thereby help to reduce latency.

- We offer two constructions that differ on whether (i) the virtual channel is guaranteed to stay off-chain for an encoded validity period, or (ii) the intermediary Ingrid can decide to offload the virtual channel (i.e., convert it into a direct channel between Alice and Bob), thereby removing its involvement in it. These two variants support different business and functionality models, analogous to non-preemptible and preemptible virtual machines in the cloud setting, with Ingrid playing the role of the service provider.
- We formalize the security properties of virtual channels as an ideal functionality in the UC framework [12], and prove that our protocols constitute a secure realization thereof. Since our virtual channels are built in the UTXO-model, our ideal functionality and formalization significantly differs from earlier work [11].
- We evaluate our protocol over two different PCN constructions, the Lightning Network (LN) [6] and Generalized channels (GC) [13], which extend LN channels to support functionality other than one-to-one payments. We show that for virtual channels on top of GC, n sequential payment operations require an off-chain exchange of $9+2\cdot n$ transactions or a total of $3524+695\cdot n$ bytes, as compared to $8\cdot n$ transactions or $3026\cdot n$ bytes when Ingrid routes the payment actively through the PCN. This means a virtual channel is already cheaper if two or more sequential payments are performed. For virtual channels over LN, n transactions require an off-chain exchange of $6292+2824\cdot n$ bytes, compared to $4776\cdot n$ bytes when routed through an intermediary. We have interacted with the Bitcoin blockchain to store the required transactions, demonstrating the compatibility of our protocol.

To summarize, for the first time in Bitcoin, we enable off-chain payments between users connected by payment channels via a hub without requiring the continuous presence of any intermediary. Hence, our solution increases the reliability and, at the same time, reduces the latency and costs of Bitcoin PCNs.

4.2. BACKGROUND

In this section, we first introduce notation and preliminaries on UTXO-based blockchains. We then overview the basics of payment and virtual channels, referring the reader to [7, 11, 14, 15] for further details. We finally discuss the main technical challenges one needs to overcome when constructing Bitcoin-compatible virtual channels.

4.2.1. UTXO-BASED BLOCKCHAINS

We adopt the notation for UTXO-based blockchains from [13], which we shortly review below.

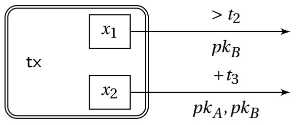
Attribute tuples Let T be a tuple of values, which we call in the following *attributes*. Each attribute in T is identified by a unique keyword, e.g., attr and referred to as T.attr.

Outputs and transactions We focus on blockchains based on the Unspent Transaction Output (UTXO) model, such as Bitcoin. In the UTXO model, coins are held in *outputs* of transactions. Formally, an output θ is an attribute tuple $(\theta.\mathsf{cash}, \theta.\varphi)$, where $\theta.\mathsf{cash}$ denotes the amount of coins associated with the output and $\theta.\varphi$ denotes the conditions that need to be satisfied in order to spend the output. The condition $\theta.\varphi$ can contain any set of operations (also called scripts) supported by the considered blockchain. We say that a user P controls or owns an output θ if $\theta.\varphi$ contains only a signature verification w.r.t. the public key of P.

In a nutshell, a *transaction* in the UTXO model, maps one or more existing outputs to a list of new outputs. The existing outputs are called *transaction inputs*. Formally, a transaction tx is an attribute tuple and consists of the following attributes $(\mathsf{tx.txid}, \mathsf{tx.Input}, \mathsf{tx.Output}, \mathsf{tx.TimeLock}, \mathsf{tx.Witness})$. The attribute $\mathsf{tx.txid} \in \{0,1\}^*$ is called the identifier of the transaction. The identifier is calculated as $\mathsf{tx.txid} := \mathcal{H}([\mathsf{tx}])$, where $\mathcal{H}$ is a hash function which is modeled as a random oracle and $[\mathsf{tx}]$ is the *body of the transaction* defined as $[\mathsf{tx}] := (\mathsf{tx.Input}, \mathsf{tx.Output}, \mathsf{tx.TimeLock})$. The attribute tx.Input is a vector of strings which identify the inputs of tx. Similarly, the outputs of the transaction tx.Output is the vector of new outputs of the transaction tx. The attribute $\mathsf{tx.TimeLock} \in \mathbb{N} \cup \{0\}$ denotes the absolute time-lock of the transaction, which intuitively means that transaction tx will not be accepted by the blockchain before the round defined by tx.TimeLock. The time-lock is by default set to 0, meaning that no time-lock is in place. Lastly, $\mathsf{tx.Witness} \in \{0,1\}^*$ called the transaction's witness, contains the witness of the transaction that is required to spend the transaction inputs.

We use charts in order to visualize the transaction flow in the rest of this work. We first explain the notation used in the charts and how they should be read. Transactions are shown using rectangles with rounded corners. Double edge rectangles are used to represent transactions that are already published on the blockchain. Single edge rectangles are transactions that could be published on the blockchain, but they are not yet. Each transaction contains one or more boxes (i.e., with squared corners) that represent the outputs of that transaction. The amount of coins allocated to each output is written inside the output box. In addition, the output condition is written on the arrow coming from the output.

In order to be concise, we use the following abbreviations for the frequently used conditions. Most outputs can only be spent by a transaction that is signed by a set of parties. In order to depict this condition, we write the public keys of all these parties *below* the arrow. We use the command `One–Sig` and `Multi–Sig` in the pseudocode. Other additional spending conditions are written *above* the arrow. The output script can have a relative time lock, i.e., a condition that is satisfied if and only if at least t rounds are passed since the transaction was published on the blockchain. We denote this output condition writing the string "$+t$" *above* the arrow (and `CheckRelative` in the pseudocode). In addition to relative time locks, an output can also have an absolute time lock, i.e., a condition that is satisfied only if t rounds elapsed since the blockchain was created and the first transaction was posted on it. We write the string "$> t$" *above* the arrow for this condition. Lastly, an output's spending condition might be a disjunction of multiple conditions. In other words it can be written as $\varphi = \varphi_1 \vee \cdots \vee \varphi_n$ for some $n \in \mathbb{N}$ where φ is the output script. In this case, we add a diamond shape to the corresponding transaction output. Each of the subconditions φ_i is then written above a separate arrow. An example is given in Fig. 4.1.



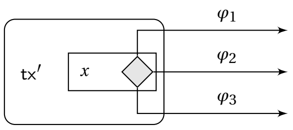


Figure 4.1: (Left) Transaction tx is published on the blockchain. The output of value x_1 can be spent by a transaction signed w.r.t. pk_B after round t_2, and the output of value x_2 can be spent by a transaction signed w.r.t. pk_A and pk_B but only if at least t_3 rounds passed since tx was accepted by the blockchain. (Right) Transaction tx′ is not published on the ledger. Its only output, which is of value x, can be spent by a transaction whose witness satisfies the output condition $\varphi_1 \vee \varphi_2 \vee \varphi_3$.

4.2.2. PAYMENT CHANNELS

A payment channel enables arbitrarily many transactions between users while requiring only two on-chain transactions. The first step when creating a payment channel is to deposit coins into an output controlled by two users. Once the money is deposited, the users can authorize new balance updates in a peer-to-peer fashion while having the guarantee that all coins are refunded at a mutually agreed time. In a bit more detail, a payment channel has three operations: *open*, *update* and *close*. We necessarily keep the description short and refer to [2, 13] for further reading.

Open Assume that Alice and Bob want to create a payment channel with an initial deposit of x_A and x_B coins, respectively. For that, Alice and Bob agree on a *funding transaction* (that we denote by $\mathtt{TX_f}$) that sets as inputs two outputs controlled by Alice and Bob holding x_A and x_B coins respectively, and transfers them to an output controlled by both Alice and Bob. When $\mathtt{TX_f}$ is added to the blockchain, the payment channel is effectively open.

Update Assume now that Alice wants to pay $\alpha \le x_A$ coins to Bob. For that, they create a new *commit transaction* $\mathtt{TX_c}$ representing the commitment from both users to the new balance of the channel. The commit transaction spends the output of $\mathtt{TX_f}$ into two new outputs: (i) one holding $x_A - \alpha$ coins controlled by Alice; and (ii) the other holding $x_B + \alpha$ coins controlled by Bob. Finally, parties exchange signatures on the commit transaction, which serve as valid witnesses for $\mathtt{TX_f}$. At this point, Alice (resp. Bob) could add $\mathtt{TX_c}$ to the blockchain. Instead, they keep it locally in their memory and overwrite it when they agree on another commitment transaction $\overline{\mathtt{TX}}_\mathtt{c}$ representing a newer balance of the channel. This, however, leads to the problem that there exist several commitment transactions that can possibly be added to the blockchain. Since all of them are spending the same output, only one can be accepted by the blockchain. Since it is impossible to prevent a malicious user from publishing an outdated commit transaction, payment channels require a mechanism that punishes such malicious behavior. This mechanism is typically called *revocation* and enables that an honest user can take all the coins locked in the channel if the dishonest user publishes an outdated commitment transaction.

Close Assume finally that Alice and Bob no longer wish to use the channel. Then, they can collaboratively close the channel by submitting the last commitment transaction

$\overline{\texttt{TX}}_\texttt{c}$ that they have agreed on to the blockchain. After it is accepted, the coins initially locked at the channel creation via $\texttt{TX}_\texttt{f}$ are redistributed to both users according to the last agreed balance. As aforementioned, if one of the users submits an outdated commitment transaction instead, the counterparty can punish the former through the revocation mechanism.

The Lightning Network [6] defines the state-of-the-art payment channel construction for Bitcoin.

4.2.3. Generalized Channels

The recent work of Aumayr et al. [13] proposes the concept of *generalized channels*. Generalized channels improve and extend payment channels (see Fig. 4.2 for details) in two ways. First, they extend the functionality of payment channels by offering off-chain execution of any script that is supported by the underlying ledger. Hence, one may view generalized channels as state channels for blockchains with restricted scripting functionality. Second, and more important for our work, generalized channels significantly improve the on-chain and off-chain communication complexity. More concretely, this efficiency improvement is achieved by introducing a so-called *split transaction* (that we denote as $\texttt{TX}_\texttt{s}$) along with a *punish-then-split* paradigm. In contrast to regular payment channels that require one revocation process per output in the commit transaction, the punish-then-split approach decouples the revocation process from the number of outputs in the commit transaction. This allows moving from revocation for each output to a single revocation for the entire channel. As shown in Figure 4.2, the commit transaction ($\texttt{TX}_\texttt{c}$) is only responsible for the punishment, while the split transaction ($\texttt{TX}_\texttt{s}$) holds the actual outputs of the channel.

The efficiency of generalized channels is further improved since they only require a single commit transaction per channel. This is in contrast to the payment channels used by Lightning, which require two distinct commit transactions for each channel user. We will discuss in Section 4.3.5 why the punish-then-split paradigm (and requiring only one commit transaction) is useful in order to improve the efficiency of our virtual channels for Bitcoin.

To simplify terminology, we will use the term *ledger channel* for all channels that are funded directly over the blockchain.

4.2.4. Channel Networks

The aforementioned payment and generalized channels allow two parties to issue transactions between each other while having to communicate with the blockchain only during the creation and closure of the channel. This on-chain communication can further be reduced by using *channel networks*.

Payment Channel Networks (PCNs) A PCN is a protocol that allows parties to connect multiple ledger channels to form a payment channel network. In this network, a sender can route a payment to a receiver as long as both parties are connected by a path in the network. Suppose that Alice and Bob are not directly connected via a ledger channel, but instead both maintain a channel with an intermediary party (Ingrid). In a nutshell, Alice can pay Bob by sending her coins to Ingrid who then forwards them in her ledger channel

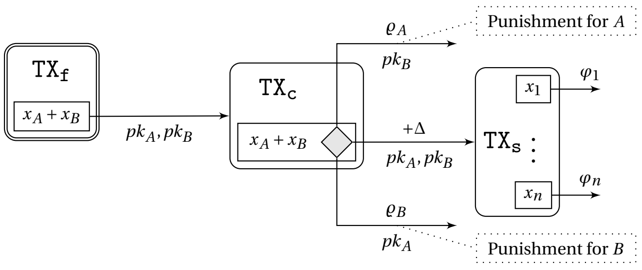


Figure 4.2: A generalized channel in the state $((x_1, \varphi_1), \ldots, (x_n, \varphi_n))$. The value of Δ upper bounds the time needed to publish a transaction on a blockchain. The condition ϱ_A represents the verification of A' revocation secret and ϱ_B represents the verification of B' revocation secret.

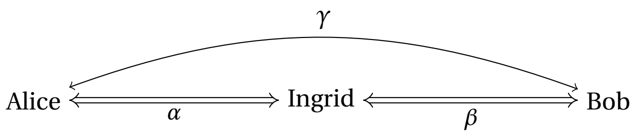


Figure 4.3: A virtual channel γ built over ledger channels α, β.

to Bob. Importantly, the protocol must achieve atomicity, i.e., either both transfers from Alice to Ingrid and from Ingrid to Bob happen, or neither of them goes through. Current PCNs such as the Lightning network use the HTLC-technique (hash-time-lock transaction), which comes with several drawbacks as mentioned in the introduction: (i) *low reliability* because the success of payments relies on Ingrid being online; (ii) *high latency* as each payment must be routed through Ingrid; (iii) *high-cost* as Ingrid may charge a fee for each payment between Alice and Bob; and (iv) *low privacy* as Ingrid can observe each payment that happens between Alice and Bob. To mitigate these issues, virtual channels have been proposed.

Virtual Channels An alternative solution to connect two payment channels with each other is offered by the concept of *virtual channels* [11]. Virtual channels allow Alice and Bob to send payments between each other without the involvement of the intermediary Ingrid. In some sense, they thus mimic the functionality offered by ledger channels, with the difference that they are not created directly over the blockchain but instead over two ledger channels. More concretely, as shown in Figure 4.3, a virtual channel γ between Alice and Bob with intermediary Ingrid is constructed on top of two ledger channels α and β. Ingrid is required to participate in the initial creation and final closing of the virtual channel. But importantly, Ingrid is not involved in any balance updates that occur in the virtual channel. This overcomes the four drawbacks mentioned above. While these advantages over PCNs make virtual channels an attractive off-chain solution, their design is far from trivial. Previous work showed how to construct virtual channels over a ledger that supports Turing complete smart contracts [11, 16, 17]. The smart contract acts in the protocol as a trust anchor that parties can fall back to in case of malicious behavior.

Through a rather complex protocol and careful smart contract design, existing virtual channel constructions guarantee that honest parties in the virtual channel will always get the coins they rightfully own. Unfortunately, most cryptocurrencies (including Bitcoin) do not offer Turing complete smart contracts, and hence the constructions from prior work cannot be used. In this work, we present a novel construction of virtual channels that makes only minimal assumptions on the underlying scripting functionality offered by the ledger.

4.3. Virtual Channels

In this section, we first give some notation before presenting the necessary properties for virtual channels and discussing design challenges. Finally, we present our protocol.

4.3.1. Definitions

We briefly recall some notation and definition for generalized channels [13] and extend the definition to generalized virtual channels. In order to make the distinction between the two types of channels clearer, we call the former generalized *ledger* channel (or ledger channels for short).

A *generalized ledger channel* as defined in [13] is a tuple $\gamma := (\gamma.\mathsf{id}, \gamma.\mathsf{Alice}, \gamma.\mathsf{Bob}, \gamma.\mathsf{cash}, \gamma.\mathsf{st})$, where $\gamma.\mathsf{id} \in \{0,1\}^*$ is the identifier of the channel, $\gamma.\mathsf{Alice}, \gamma.\mathsf{Bob} \in \mathcal{P}$ are the identities of the parties using the channel, $\gamma.\mathsf{cash} \in \mathbb{R}^{\geq 0}$ is a finite precision real number that represents the total amount of coins locked in this channel and $\gamma.\mathsf{st} = (\theta_1, \ldots, \theta_n)$ is the state of the channel. This state is composed of a list of *outputs*. Recall that each output θ_i has two attributes: the output value $\theta_i.\mathsf{cash} \in \mathbb{R}^{\geq 0}$ and the output condition $\theta_i.\varphi\colon \{0,1\}^* \times \mathbb{N} \times \mathbb{N} \to \{0,1\}$. For convenience, we define a set $\gamma.\mathsf{endUsers} := \{\gamma.\mathsf{Alice}, \gamma.\mathsf{Bob}\}$ and a function $\gamma.\mathsf{otherParty}\colon \gamma.\mathsf{endUsers} \to \gamma.\mathsf{endUsers}$, which on input $\gamma.\mathsf{Alice}$ outputs $\gamma.\mathsf{Bob}$ and on input $\gamma.\mathsf{Bob}$ returns $\gamma.\mathsf{Alice}$.

A generalized *virtual channel* (or for short virtual channel) is defined as a tuple $\gamma := (\gamma.\mathsf{id}, \gamma.\mathsf{Alice}, \gamma.\mathsf{Bob}, \gamma.\mathsf{cash}, \gamma.\mathsf{st}, \gamma.\mathsf{Ingrid}, \gamma.\mathsf{subchan}, \gamma.\mathsf{fee}, \gamma.\mathsf{val})$. The attributes $\gamma.\mathsf{id}$, $\gamma.\mathsf{Alice}, \gamma.\mathsf{Bob}, \gamma.\mathsf{cash}, \gamma.\mathsf{st}$ are defined as in the case of ledger channels. The additional attribute $\gamma.\mathsf{Ingrid} \in \mathcal{P}$ denotes the identity of the *intermediary* of the virtual channel γ. The set $\gamma.\mathsf{endUsers}$ and the function $\gamma.\mathsf{otherParty}$ are defined as before. Additionally, we also define the set $\gamma.\mathsf{users} := \{\gamma.\mathsf{Alice}, \gamma.\mathsf{Bob}, \gamma.\mathsf{Ingrid}\}$. The attribute $\gamma.\mathsf{subchan}$ is a function mapping $\gamma.\mathsf{endUsers}$ to a channel identifier; namely, the value $\gamma.\mathsf{subchan}(\gamma.\mathsf{Alice})$ refers to the identifier of the channel between $\gamma.\mathsf{Alice}$ and $\gamma.\mathsf{Ingrid}$ (i.e., the id of α from the description above); similarly, the value $\gamma.\mathsf{subchan}(\gamma.\mathsf{Bob})$ refers to the identifier of the channel between $\gamma.\mathsf{Bob}$ and $\gamma.\mathsf{Ingrid}$ (i.e., β from the description above). The value $\gamma.\mathsf{fee} \in \mathbb{R}^{\geq 0}$ represents the fee charged by $\gamma.\mathsf{Ingrid}$ for her service of being an intermediary of γ. Finally, we introduce the attribute $\gamma.\mathsf{val} \in \mathbb{N} \cup \{\bot\}$. If $\gamma.\mathsf{val} \neq \bot$, then we call γ a *virtual channel with validity* and the value of $\gamma.\mathsf{val}$ represents the round number until which γ remains open. Channels with $\gamma.\mathsf{val} = \bot$ are called *virtual channels without validity*.

4.3.2. Security and Efficiency Goals

We briefly recall the properties of generalized channels as defined in [13] and state the additional properties that we require from virtual channels.

Security goals Generalized ledger channels must satisfy three security properties, namely (S1) Consensus on creation, (S2) Consensus on update and (S3) Instant finality with punish. Intuitively, properties (S1) and (S2) guarantee that successful creation of a new channel as well as successful update of an existing channel happens if and only if both parties agree on the respective action. Property (S3) states that if a channel γ is successfully updated to the state $\gamma.\mathsf{st}$ and $\gamma.\mathsf{st}$ is the last state that the channel is updated to, then an honest party $P \in \gamma.\mathsf{endUsers}$ can either enforce this state on the ledger or P can enforce a state where she gets all the coins locked in the channel. We say that a state st is *enforced* when a transaction with this state appears on the ledger.

Since virtual channels are generalized channels whose funding transaction is not posted on the ledger yet, the above stated properties should hold for virtual channels as well with two subtle but important differences: (i) the creation of a virtual channel involves three parties (Alice, Ingrid and Bob) and hence consensus on creation for virtual channels can only be fulfilled if all three parties agree on the creation; (ii) the finality (i.e., offloading) of the virtual channel depends on whether Alice is expected to offload the virtual channel within a predetermined validity period (virtual channel with validity VC-V) or the offload task is delegated to the intermediary Ingrid without having a predefined validity period (virtual channel without validity VC-NV). In order to account for these two differences, virtual channels should also satisfy the following properties:

(V1) Balance security: If γ is a virtual channel and $\gamma.\mathsf{Ingrid}$ is honest, she never loses coins, even if $\gamma.\mathsf{Alice}$ and $\gamma.\mathsf{Bob}$ collude.

(V2) Offload with punish: If γ is a virtual channel *without* validity (VC-NV), then $\gamma.\mathsf{Ingrid}$ can transform γ to a ledger channel. Party $P \in \gamma.\mathsf{endUsers}$ can initiate the transformation which either completes or P can get financially compensated.

(V3) Validity with punish: If γ is a virtual channel *with* validity (VC-V), then $\gamma.\mathsf{Alice}$ can transform γ to a ledger channel. If γ is not transformed into a ledger channel or closed before time $\gamma.\mathsf{val}$, $\gamma.\mathsf{Ingrid}$ and $\gamma.\mathsf{Bob}$ can get financially compensated.

We first note that the instant finality with punish property (S3) does not provide any guarantees for $\mathsf{Ingrid} \not\in \gamma.\mathsf{endUsers}$, which is why we need to define (V1) for virtual channels. Properties (V2) and (V3) point out the main difference between VC-NV and VC-V. In a VC-NV γ, Ingrid is able to free her collateral from γ at any time by transforming the channel between Alice and Bob from a virtual channel to a ledger channel. Furthermore,

	L-Security	V-Security			Efficiency		
	S1 – S3	V1	V2	V3	E1	E2	E3
L	✓	-	-	-	✗	✓	✗
VC-V	✓	✓	✗	✓	✓	✓	✓
VC-NV	✓	✓	✓	✗	✓	✓	✓

Table 4.1: Comparison of security and efficiency goals for ledger channels (L), virtual channels with validity (VC-V) and virtual channels without validity (VC-NV).

in case Alice and Bob transform the virtual channel to a ledger channel or even misbehave, honest Ingrid is guaranteed that she will receive the collateral back. In a VC-V γ, Ingrid cannot transform a virtual channel into a ledger channel at any time she wants. Instead, there is a pre-agreed point in time, defined by γ.val, until when γ.endUsers have to close the virtual channel or transform it into a ledger channel (Ingrid's collateral is freed in both cases). If γ.endUsers fail to do so, Ingrid can get her collateral back through a punishment mechanism. Hence, γ.endUsers have a guarantee that their VC-V will remain a virtual channel until a certain round, after which they must ensure its closure or transformation to avoid punishments.

Efficiency goals Lastly, we define the following efficiency goals, which describe the number of rounds certain protocol steps require:

(E1) Constant round creation: Successful creation of a virtual channel takes a constant number of rounds.

(E2) Optimistic update: For a channel γ, this property guarantees that in the optimistic case when both parties in γ.endUsers are honest, a channel update takes a constant number of rounds.

(E3) Optimistic closure: In the optimistic case when all parties in γ.users are honest, the closure of a virtual channel takes a constant number of rounds.

Let us stress that property (E2) is common for all off-chain channels (i.e., both ledger and virtual channels). The properties (E1) and (E3) capture the additional property of virtual channels that in the optimistic case when all parties behave honestly, the entire life-cycle of the channel is performed completely off-chain.

We compare the security and efficiency goals for different types of channels in Table 4.1. Now, we formalize these properties as a UC ideal functionality.

4.3.3. IDEAL FUNCTIONALITY FOR VIRTUAL CHANNELS

Here we define the ideal functionality $\mathcal{F}_V$ that describes the ideal behavior of both ledger and virtual channels. The full description of the ideal functionalities can be found in the full version of this paper [18].

$\mathcal{F}_V$ can be viewed as an extension of the ledger channel functionality $\mathcal{F}_L$ defined in [13]. The functionality $\mathcal{F}_V$ is parameterized by a parameter T which upper bounds the maximum number of off-chain communication rounds between two parties required for any of the operations in $\mathcal{F}_L$. The ideal functionality $\mathcal{F}_V$ communicates with the parties $\mathcal{P}$, the simulator $\mathcal{S}$ and the ledger $\widehat{\mathcal{L}}$. It maintains a channel space Γ where it stores all currently opened ledger channels (together with their funding transaction tx) and virtual channels. Before we define $\mathcal{F}_V$ formally, we describe it at a high level.

Messages related to ledger channels For any message related to a ledger channel, $\mathcal{F}_V$ behaves as the functionality $\mathcal{F}_L$. That is, the corresponding code of $\mathcal{F}_L$ is executed when a message about a ledger channel γ is received. For the rest of this section, we discuss the behavior of $\mathcal{F}_V$ upon receiving a message about a virtual channel.

Create The creation of a virtual channel is equivalent to synchronously updating two ledger channels. Therefore, if all parties, namely γ.Alice, γ.Bob and γ.Ingrid, follow the protocol, i.e., update their ledger channels correctly, a virtual channel is successfully created. This is captured in the "All agreed" case of the functionality. Hence, if all parties send the `CREATE` message, the functionality returns `CREATED` to γ.users, keeps the underlying ledger channels locked and adds the virtual channel to its channel space Γ.

On the other hand, the creation of the virtual channel fails if after some time at least one of the parties does not send `CREATE` to the functionality. There are three possible situations: (i), the update is peacefully rejected and parties simply abort the virtual channel creation, (ii) both channels are forcefully closed, in order to prevent a situation where one of the channels is updated and the other one is not, (iii) if γ.Ingrid has not published the old state of one of her channels to the ledger after Δ rounds, it forcefully closes the ledger channels using the new state i.e., where γ.Ingrid behaves maliciously and can publish both the old and new states, while γ.Alice or γ.Bob can only publish the new state.

Update The update procedure for the virtual channel works in the same way as for ledger channels except in case of any disputes during the execution, the functionality calls `V–ForceClose` instead of `L–ForceClose`.

Offload We consider two types of offloading depending on whether the virtual channel is with or without validity. In the first case, offloading is initiated by one of the γ.endUsers before round γ.val, while for channels without validity, Ingrid can initiate the offloading at any time. Since offloading a virtual channel requires closure of the underlying subchannels, the functionality merely checks if either funding transaction of γ.subchan has been spent until round $T_1 + \Delta$. If not, the functionality outputs a message (`ERROR`). As in to [13], the `ERROR` message represents an impossible situation which should not happen as long as one of the parties is honest.

Close - channels without validity Upon receiving (`CLOSE`, *id*) from all parties in γ.users within $T_1 \leq 6T$ rounds (where the exact value of T_1 is specified by $\mathcal{S}$), all parties have peacefully agreed on closing the virtual channel, which is indicated by the "All Agreed" case. In this case the final balance of the parties is reflected on their underlying channels. When the update of Γ is completed, the ideal functionality sends `CLOSED` to all users. Due to the peaceful closure in this "All Agreed" case, the functionality defines property (E3).

If one of the (`CLOSE`, *id*) messages was not received within T_1 rounds ("Wait for others" case), the closing procedure fails. The following cases my happen: (i) the update procedure of an underlying ledger channel was aborted prematurely by γ.Alice or γ.Bob which would cause the virtual channel to be forcefully closed. (ii) γ.Ingrid refuses to revoke her state during the update of either one of the underlying ledger channels where the functionality waits Δ rounds and if γ.Ingrid has not published the old state to the ledger the functionality forcefully closes the ledger channels using the new state.

Close - channels with validity. This procedure starts in round $\gamma.\mathsf{val} - (4\Delta + 7T)$ to have enough time to forcefully close the channel if necessary. If within $T_1 \leq 6T$ rounds (where the exact value of T_1 is specified by $\mathscr{S}$) all γ.users agreed on closing the channel or if the simulator instructs the functionality to close the channel, the same steps as in the all agreed case for channels without validity are executed. Otherwise, after T_1 rounds, the functionality executes the forceful closure of the virtual channel.

Punish The punishment procedure is executed at the end of each round. It checks for every virtual channel γ if any of γ.subchan has just been closed and distinguishes if the consequence of closure was offloading or punishment. If after T_1 rounds (T_1 is set by $\mathscr{S}$) two transactions tx_1 and tx_2 are published on the ledger, where tx_1 refunds the collateral γ.cash + γ.fee to γ.Ingrid and tx_2 funds γ on-chain, then the virtual channel has been offloaded and the message (`OFFLOADED`) is sent to γ.users. If after T_1 rounds, only one transaction tx is on the ledger, which assigns γ.cash coins to a single honest party P and spends the funding transaction of only one of γ.subchan, the functionality sends (`PUNISHED`) to P. Otherwise, the functionality outputs (`ERROR`) to γ.users.

Notation In the functionality description, we use the notion of *rooted transactions* that we now explain (see Figure 4.4 for a concrete example). UTXO based blockchains can be viewed as a directed acyclic graph, where each node represents a transaction. Nodes corresponding to transactions tx_i and tx_j are connected with an edge if at least one of the outputs of tx_i is an input of tx_j, i.e, tx_i is (partially) funding tx_j. We denote the transitive reachability relation between nodes, which constitutes a partial order, as $\leq$. We say that a transaction tx is *rooted* in the set of transactions R if

1. $\forall \mathsf{tx}_i \leq \mathsf{tx}. \exists \mathsf{tx}_j \in R. \mathsf{tx}_j \leq \mathsf{tx}_i \vee \mathsf{tx}_i \leq \mathsf{tx}_j$,
2. $\forall \mathsf{tx}_i, \mathsf{tx}_j \in R. \mathsf{tx}_i \neq \mathsf{tx}_j, \mathsf{tx}_i \nleq \mathsf{tx}_j$ and
3. $\mathsf{tx} \notin R$.

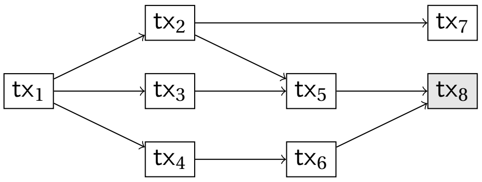

$$\begin{tikzcd}
 & \mathsf{tx}_2 \arrow[rr] \arrow[dr] & & \mathsf{tx}_7 \\
\mathsf{tx}_1 \arrow[ur] \arrow[r] \arrow[dr] & \mathsf{tx}_3 \arrow[r] & \mathsf{tx}_5 \arrow[r] & \mathsf{tx}_8 \\
 & \mathsf{tx}_4 \arrow[r] & \mathsf{tx}_6 \arrow[ur] &
\end{tikzcd}$$

Figure 4.4: The root sets of transaction tx_8 are $\{\mathsf{tx}_1\}$, $\{\mathsf{tx}_2, \mathsf{tx}_3, \mathsf{tx}_4\}$, $\{\mathsf{tx}_5, \mathsf{tx}_6\}$, $\{\mathsf{tx}_4, \mathsf{tx}_5\}$ and $\{\mathsf{tx}_2, \mathsf{tx}_3, \mathsf{tx}_6\}$.

Moreover, in order to simplify the notation in the functionality description, we write $m \overset{t}{\hookrightarrow} P$ as a short hand form for "send the message m to party P in round t." and $m \overset{t}{\hookleftarrow} P$ for "receive a message m from party P in round t".

Ideal Functionality $\mathscr{F}_V(T)$

Below we abbreviate $A := \gamma$.Alice, $B := \gamma$.Bob, $I = \gamma$.Ingrid. For $P \in \gamma$.endUsers, we denote $Q := \gamma$.otherParty(P).

For messages about ledger channels, behave as $\mathcal{F}_L(T, 1)$.

$\boxed{\text{Create}}$

Upon $(\texttt{CREATE}, \gamma) \overset{\tau}{\hookleftarrow} P$, let $\mathcal{S}$ define $T_1 \leq 8T$. If $P \in \gamma$.endUsers, then define a set S, where $S := \{id_P\} := \gamma.\mathsf{subchan}(P)$, otherwise define S as $S := \{id_P, id_Q\} := \gamma.\mathsf{subchan}$. Lock all channels in S and distinguish:

All agreed: If you already received both $(\texttt{CREATE}, \gamma) \overset{\tau_1}{\hookleftarrow} Q_1$ and $(\texttt{CREATE}, \gamma) \overset{\tau_2}{\hookleftarrow} Q_2$, where $Q_1, Q_2 \in \gamma.\mathsf{users} \setminus \{P\}$ and $\tau - T_1 \leq \tau_1 \leq \tau_2$, then in round $\tau_3 := \tau_1 + T_1$ proceed as:

1. Let $\mathcal{S}$ define $\vec{\theta}_A$ and $\vec{\theta}_B$ and set $(id_A, id_B) := \gamma.\mathsf{subchan}$.
2. Execute $\texttt{UpdateState}(id_A, \vec{\theta}_A)$, $\texttt{UpdateState}(id_B, \vec{\theta}_B)$, set $\Gamma(\gamma.\mathsf{id}) := \gamma$, send $(\texttt{CREATED}, \gamma) \overset{\tau_3}{\hookrightarrow} \gamma$.endUsers, stop.

Wait for others: Else wait for at most T_1 rounds to receive $(\texttt{CREATE}, \gamma) \xleftarrow{\tau_1 \leq \tau + T_1} Q_1$ and $(\texttt{CREATE}, \gamma) \xleftarrow{\tau_2 \leq \tau + T_1} Q_2$ where $Q_1, Q_2 \in \gamma.\mathsf{users} \setminus \{P\}$ (in that case option "All agreed" is executed). If at least one of those messages does not arrive before round $\tau + T_1$, do the following. For all $id_i \in S$, let $(\gamma_i, \mathsf{tx}_i) := \Gamma(id_i)$ and distinguish the following cases:

- If $\mathcal{S}$ sends (peaceful–reject, id_i), unlock id_i and stop.
- If γ.Ingrid is honest or if instructed by $\mathcal{S}$, execute $\texttt{L–ForceClose}(id_i)$ and stop.
- Otherwise wait for Δ rounds. If tx_i still unspent, then set $\vec{\theta}_{old} := \gamma_i.\mathsf{st}$, $\gamma_i.\mathsf{st} := \{\vec{\theta}_{old}, \vec{\theta}\}$ and $\Gamma(id_i) := (\gamma_i, \mathsf{tx}_i)$. Execute $\texttt{L–ForceClose}(id_i)$ and stop.

$\boxed{\text{Update}}$

Upon $(\texttt{UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \overset{\tau_0}{\longleftarrow} P$, where $P \in \gamma$.endUsers, behave as $\mathcal{F}_L(T, 1)$ yet replace the calls to $\texttt{L–ForceClose}$ in $\mathcal{F}_L(T, 1)$ with calls to $\texttt{V–ForceClose}$.

$\boxed{\text{Offload}}$

Upon $(\texttt{OFFLOAD}, id) \overset{\tau_0}{\longleftarrow} P$, execute $\texttt{Offload}(id)$.

$\boxed{\text{Close}}$

<u>Channels without validity:</u>

Upon $(\texttt{CLOSE}, id) \overset{\tau}{\hookleftarrow} P$, where $\gamma(id).\mathsf{val} = \perp$, let $\mathcal{S}$ define $T_1 \leq 6T$. If $P \in \gamma_i$.endUsers, define a set S, where $S := \{id_P\} := \gamma_i.\mathsf{subchan}(P)$, else define S as $S := \{id_P, id_Q\} := \gamma_i.\mathsf{subchan}$ and distinguish:

All agreed: If you received both messages $(\texttt{CLOSE}, id) \overset{\tau_1}{\hookleftarrow} Q_1$ and $(\texttt{CLOSE}, id) \overset{\tau_2}{\hookleftarrow} Q_2$, where $Q_1, Q_2 \in \gamma.\mathsf{users} \setminus \{P\}$ and $\tau - T_1 \leq \tau_1 \leq \tau_2$, then in round $\tau_3 := \tau_1 + T_1$ proceed as follows:

1. Let $\gamma := \Gamma(id)$, $(id_A, id_B) := \gamma.\mathsf{subchan}$.

2. Parse $\gamma.\mathsf{st} = \{(c_A, \texttt{One–Sig}_A), (c_B, \texttt{One–Sig}_B)\}$ and set

$$\vec{\theta}_A := ((c_A, \texttt{One–Sig}_A), (c_B + \gamma.\mathsf{fee}/2, \texttt{One–Sig}_I)),$$
$$\vec{\theta}_B := ((c_A + \gamma.\mathsf{fee}/2, \texttt{One–Sig}_I), (c_B, \texttt{One–Sig}_B)),$$

3. Unlock both subchannels and execute $\texttt{UpdateState}(id_A, \vec{\theta}_A)$ and $\texttt{UpdateState}(id_B, \vec{\theta}_B)$. Set $\Gamma(id) := \bot$ and send $(\texttt{CLOSED}, \gamma) \xhookrightarrow{\tau_3} \gamma.\mathsf{endUsers}$.

Wait for others: Else wait for at most T_1 rounds to receive $(\texttt{CLOSE}, \gamma) \xleftarrow{\tau_1 \le \tau + T_1} Q_1$ and $(\texttt{CLOSE}, \gamma) \xleftarrow{\tau_2 \le \tau + T_1} Q_2$ where $Q_1, Q_2 \in \gamma.\mathsf{users} \setminus \{P\}$ (in that case option "All agreed" is executed). For all $id_i \in S$ let $(\gamma_i, \mathsf{tx}_i) := \Gamma(id_i)$, if such messages are not received until round $\tau + T_1$, set $\vec{\theta}_{old} := \gamma'.\mathsf{st}$ and distinguish:

- If $\gamma.\mathsf{Ingrid}$ is honest or if instructed by $\mathcal{S}$, execute $\texttt{V–ForceClose}(id_i)$ and stop.
- Else wait for Δ rounds. If tx_i still unspent, set $\gamma_i.\mathsf{st} := \{\vec{\theta}_{old}, \vec{\theta}\}$ and $\Gamma(id_i) := (\gamma_i, \mathsf{tx}_i)$. Execute $\texttt{L–ForceClose}(id_i)$ and stop.

Channels with validity:
For every $\gamma \in \Gamma$ s.t. $\gamma.\mathsf{val} \neq \bot$, in round $\tau_0 := \gamma.\mathsf{val} - (4\Delta + 7T)$ proceed as follows: let $\mathcal{S}$ set $T_1 \le 6T$ and distinguish:

Peaceful close: If all parties in $\gamma.\mathsf{users}$ are honest or if instructed by $\mathcal{S}$, execute steps (1)–(3) of the "All agreed" case for channels without validity with $\tau_3 := \tau_0 + T_1$.

Force close: Else in round τ_3 execute $\texttt{V–ForceClose}(\gamma.\mathsf{id})$.

Punishment (executed at the end of every round)

For every id, where $\gamma := \Gamma(id)$ is a virtual channel, set $(id_A, id_B) := \gamma.\mathsf{subchan}$. If this is the first round when $\Gamma(id_A) = (\bot, \mathsf{tx}_A)$ or $\Gamma(id_B) = (\bot, \mathsf{tx}_B)$, i.e., one of the subchannels was just closed, then let $\mathcal{S}$ set $t_1 \le T'$, where $T' := \tau_0 + T + 5\Delta$ if $\gamma.\mathsf{val} = \bot$ and $T' := \gamma.\mathsf{val} + 3\Delta$ if $\gamma.\mathsf{val} \neq \bot$, and distinguish the following cases:

Offloaded: Latest in round t_1 the ledger $\widehat{\mathcal{L}}$ contains both

- a transaction tx_1 rooted at $\{\mathsf{tx}_A, \mathsf{tx}_B\}$ with an output $(\gamma.\mathsf{cash} + \gamma.\mathsf{fee}, \texttt{One–Sig}_I)$. In this case $(\texttt{OFFLOADED}, id) \xhookrightarrow{\tau_1} I$, where τ_1 is the round tx_1 appeared on $\widehat{\mathcal{L}}$.
- a transaction tx_2 with an output of value $\gamma.\mathsf{cash}$ and rooted at $\{\mathsf{tx}_A, \mathsf{tx}_B\}$, if $\gamma.\mathsf{val} = \bot$, and rooted at $\{\mathsf{tx}_A\}$, if $\gamma.\mathsf{val} \neq \bot$. Let τ_2 be the round when tx_2 appeared on $\widehat{\mathcal{L}}$. Then output $(\texttt{OFFLOADED}, id) \xhookrightarrow{\tau_2} \gamma.\mathsf{endUsers}$, set $\gamma' = \gamma$, $\gamma'.\mathsf{Ingrid} = \bot$, $\gamma'.\mathsf{subchan} = \bot$, $\gamma.\mathsf{val} = \bot$ and define $\Gamma(id) := (\gamma', \mathsf{tx}_2)$.

Punished: Else for every honest party $P \in \gamma.\mathsf{users}$, check the following: the ledger $\widehat{\mathcal{L}}$ contains in round $\tau_1 \le t_1$ a transaction tx rooted at either tx_A or tx_B with $(\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \texttt{One–Sig}_P)$ as output. In that case, output $(\texttt{PUNISHED}, id) \xhookrightarrow{\tau_1} P$. Set $\Gamma(id) = \bot$ in the first round when $\texttt{PUNISHED}$ was sent to all honest parties.

Error: If the above case is not true, then $(\texttt{ERROR}) \xhookrightarrow{t_1} \gamma.\mathsf{users}$.

V–ForceClose(id): Let τ_0 be the current round and $\gamma := \Gamma(id)$. Execute subprocedure Offload(id). Let $T' := \tau_0 + 2T + 8\Delta$ if $\gamma.\mathsf{val} = \perp$ and $T' := \gamma.\mathsf{val} + 3\Delta$ if $\gamma.\mathsf{val} \neq \perp$. If in round $\tau_1 \leq T'$ it holds that $\Gamma(id) = (\gamma, \mathsf{tx})$, execute subprocedure L–ForceClose(id).
Subprocedure Offload(id): Let τ_0 be the current round, $\gamma := \Gamma(id)$, $(id_\alpha, id_\beta) := \gamma.\mathsf{subchan}$, $(\alpha, \mathsf{tx}_A) := \Gamma(id_\alpha)$ and $(\beta, \mathsf{tx}_B) := \Gamma(id_\beta)$. If within Δ rounds, neither tx_A nor tx_B is spent, then output (ERROR) $\xhookrightarrow{\tau_0+\Delta}$ $\gamma.\mathsf{users}$.
Subprocedure UpdateState($id, \vec{\theta}$): Let $(\alpha, \mathsf{tx}) := \Gamma(id)$. Set $\alpha.\mathsf{st} := \vec{\theta}$ and update $\Gamma(id) := (\alpha, \mathsf{tx})$.

4.3.4. DESIGN CHALLENGES FOR CONSTRUCTING VIRTUAL CHANNELS

The main challenges that arise when constructing Bitcoin-compatible virtual channels stem from the need to ensure the security properties (V1) - (V3) as presented in the previous section. Namely, to guarantee balance security to the intermediary, we need to ensure that the virtual channel creation and closure is reflected symmetrically and synchronously on both underlying ledger channels. We identify this as a challenge (C1). As we discuss in more detail below, this can be solved by giving the intermediary the right of a “last say” in the virtual channel creation and closure procedures. However, a malicious intermediary could abuse such power and block virtual channel closure indefinitely. Therefore, the second challenge (C2) is to design a punishment mechanism that allows virtual channel users to either enforce closure or claim financial compensation. We provide some further details below.

Synchronous create and close (C1) The creation and closure of a virtual channel are done by updating the underlying ledger channels. In order to guarantee balance security for the intermediary, we must ensure that updates on both ledger channels are symmetric and either both of them succeed or both of them fail. That is, if the intermediary Ingrid loses coins in one ledger channel as a result of the virtual channel construction, then she has the guarantee of gaining the same amount of coins from the other ledger channel. Such an atomicity property can be achieved by allowing Ingrid to be the reacting party in both ledger channel update procedures. Namely, Ingrid has to receive symmetric update requests from both Alice and Bob before she confirms either of them.

As a result, Ingrid has the power to block a virtual channel creation and closure. For a virtual channel creation, this is not a problem. It simply represents the fact that Ingrid does not want to be an intermediary, and hence Alice and Bob have to find a different party. However, for virtual channel closing, this power of the intermediary results in a violation of the instant finality property for Alice and Bob, and requires a more involved mechanism.

Enforcing virtual channel state (C2) In contrast to standard ledger channels that rely on funding transactions that are published on the ledger, the funding transactions of a virtual channel are, in the optimistic case (i.e., when parties are honest), kept off-chain. In case of misbehavior (e.g., when malicious Ingrid refuses to close the virtual channel), however, honest parties must be able to publish the virtual channel funding transaction to the blockchain in order to enforce the latest state of the virtual channel. Unfortunately, the funding transactions can only be published if *both* of the underlying channels are closed in a state which funds the virtual channel. The fact that the virtual channel participants, Alice and Bob, respectively have control over just one of the underlying ledger channels further complicates this situation. For instance, one of the underlying ledger channels may be updated or closed maliciously at any time which would prevent the publishing of the funding transaction on the ledger.

4.3.5. Virtual Channel Protocol

We now show how to build virtual channels on top of generalized channels. We later discuss how our construction can be built over other channels such as Lightning and why generalized channels offer better efficiency.

As mentioned in the previous section, virtual channels are created and closed through an update of the underlying ledger channels. Hence, let us recall the update process of ledger channels, depicted as UpdateChan in Figs. 4.6 and 4.7, before explaining our construction in more detail. The update procedure consists of 4 steps, namely (1) the *Initialization* step, during which parties agree on the new state of the channel, (2) the *Preparation* step, where parties generate the transactions with the given state, (3) the *Setup* during which parties exchange their application-dependent data (e.g., for building virtual channels), and finally (4) the *Completion* step where parties commit to the new state and revoke the old one. We refer the reader to [13] for more details.

High level protocol description

We are now prepared to present a high-level description of our modular virtual channel protocol and explain how to solve the main technical challenges when designing virtual channels. In a nutshell, this modular protocol gives a generic framework on how to design virtual channels. Afterwards, we show how to instantiate this modular protocol with our virtual channel constructions without validity and with validity. We present the formal pseudocode for the modular protocol in Figure 4.5.

Create Let γ be a virtual channel that $A := \gamma.\mathsf{Alice}$ and $B := \gamma.\mathsf{Bob}$ want to create, using their generalized ledger channels with $I := \gamma.\mathsf{Ingrid}$. At a high level, the creation procedure of a virtual channel is a synchronous update of the underlying ledger channels. Given the ledger channels, we proceed as follows (see Fig. 4.6).

As a first step, each party $P \in \{A, B\}$ initiates an update of the respective ledger channel with I (step ①) who, upon receiving both update requests, checks if the requested states (i.e., θ_A and θ_B) are consistent. The parties use the identifiers tid_A and tid_B of their subchannels in order to build the virtual channel (step ②). Next, all three parties engage in a setup phase, in which the structure of the virtual channel is built (step ③). More concretely, all three parties agree on a funding transaction of the virtual channel which when published on the blockchain transforms the virtual channel to a ledger channel. When the setup phase is completed, i.e., the virtual channel structure has been built, the parties complete the ledger channel update procedures (step ④). It is crucial for the intermediary I to have the role of a reacting party during both channel updates. This gives her the power to wait until she is sure that both updates will complete successfully

Create virtual channel for $P \in \{A, B\}$

```
// Initiate creation of $\gamma$ with funding source $tid_A, tid_B$
Let $\gamma_P$ be the channel with id $\gamma$.subchan($P$).
Compute $\theta_P :=$ GenVChannelOutput($\gamma_P, P$)
Assign Setup $\leftarrow$ SetupVChannel$^P$($\gamma, tid_A, tid_B$)
if UpdateChan$^P$($\gamma_P, \theta_P$, Setup) returns
  UPDATE–OK Creation successful.
```

Close virtual channel for $P \in \{A, B\}$

```
// Initiate closure of $\gamma$
Let $\gamma_P$ be the channel with id $\gamma$.subchan($P$).
Parse $\gamma$.st $= \Big((c_P$, One–Sig$_{pk_P}$),
($c_Q$, One–Sig$_{pk_Q}$)$\Big)$
Compute $\vec{\theta}_P := \{(c_P$, One–Sig$_{pk_P}$),
($c_Q + \frac{\gamma.\text{fee}}{2}$, One–Sig$_{pk_I}$)$\}$
if UpdateChan$^P$($\gamma_P, \vec{\theta}_P, \perp$) returns UPDATE–OK
  Close successful.
else Execute Offload$^P$($\gamma$) and stop.
```

Update virtual channel for $P \in \{A, B\}$

```
// Initiate update of $\gamma$ with state $\vec{\theta}$
if UpdateChan$^P$($\gamma, \vec{\theta}, \perp$) returns UPDATE–OK
  Update successful.
else Execute Offload$^P$($\gamma$) and stop.
```

SetupVChannel(γ, tid_A, tid_B)

```
// Return the setup procedure Setup required for the setup
// of the virtual channel $\gamma$. The funding transaction and
// initial versions of split and commit transactions of $\gamma$
// are created. Moreover, the punishment and refund
// transactions are generated to be used in malicious cases.
```

GenVChannelOutput(γ_P, P)

```
// Return output $\theta$ of $\gamma_P$ that will fund the virtual channel
```

Create virtual channel for I

```
// React to creation of $\gamma$ with funding source $tid_A, tid_B$
For $P \in \{A, B\}$,
Let $\gamma_P$ be the channel with id $\gamma$.subchan($P$).
Compute $\theta_P :=$ GenVChannelOutput($\gamma_P, P$)
Assign Setup $\leftarrow$ SetupVChannel$^I$($\gamma, tid_A, tid_B$)
if UpdateChanSync$^I$($\gamma_A, \vec{\theta}_A, \gamma_B, \vec{\theta}_B$, Setup)
  returns UPDATE–OK Creation successful.
```

Close virtual channel for I

```
// React to closure of $\gamma$ for some $c_P, c_Q$ s.t. $c_P + c_Q = \gamma$.cash
For $P \in \{A, B\}$,
Let $\gamma_P$ be the sub-channel $\gamma$.subchan($P$).
Compute $\vec{\theta}_P = \{(c_P$, One–Sig$_{pk_P}$),
($c_Q + \frac{\gamma.\text{fee}}{2}$, One–Sig$_{pk_I}$)$\}$for $P \in \{A, B\}$.
if UpdateChanSync$^I$($\gamma_A, \vec{\theta}_A, \gamma_B, \vec{\theta}_B, \perp$) returns
  UPDATE–OK Close successful.
else Execute Offload$^I$($\gamma$).
```

Punish for all parties

```
// In every round check the ledger and punish misbehavior
for every open channel $\gamma$ execute Punish($\gamma$)
```

UpdateChan^P($\gamma, \vec{\theta}$, Setup) from [13]

```
// Initiate update of $\gamma$ with state $\vec{\theta}$ with Setup.
```

UpdateChanSync^I($\gamma_1, \vec{\theta}_1, \gamma_2, \vec{\theta}_2$, Setup)

```
// Initiate update of $\gamma_i$ with state $\vec{\theta}_i$ with Setup for $i = 1,2$
// simultaneously using the same steps of UpdateChan.
// At each step, wait for both channels before continuing.
// If one of them fails at any step, act as both failed.
```

PreCreateChan(TX$_f^\gamma$) from [13]

```
// Creates a channel $\gamma$ with initial versions of split and
// commit transactions. It follows the channel creation
// procedure given in [13], expect that
// the funding transaction is not published in the end.
```

Figure 4.5: Protocol for virtual channels. The protocol utilizes the generalized channel protocols from [13]. Specifically, the channel update protocol UpdateChan is used in a black-box fashion while also defining a synchronized version called UpdateChanSync. Moreover, the channel creation protocol PreCreateChan is used with the difference of not publishing the channel funding transaction of the virtual channel. The gray parts of the protocol differ between our tow constructions with and without validity and are specified in the protocol pseudocode and description.

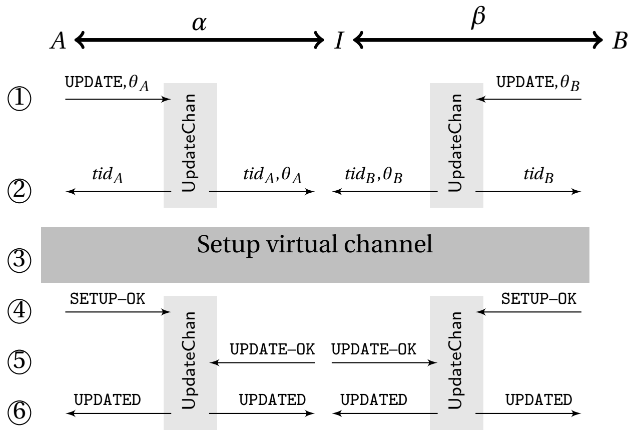


Figure 4.6: Modular creation procedure of a virtual channel on top of two ledger channels α and β.

and only then give her the final update agreement (step ⑤). Upon a successful execution, parties consider the channels as updated (step ⑥), which implies that the virtual channel γ was successfully created.

Update Updating the virtual channel essentially works in the same way as the update procedure of a ledger channel. As long as the update is successful or peacefully rejected (meaning that the reacting party rejects the update), the parties act as instructed in the ledger channel protocol. The situation is more delicate when the update fails because one of the parties misbehaved and aborted the procedure.

We note that aborts during a channel update might cause a problematic asymmetry between the parties. For instance, when one party already signed the new state of the channel while the other one did not; or when one party already revoked the old state of the channel but the other one did not. In a standard ledger channel, these disputes are resolved by a force close procedure, meaning that the honest party publishes the latest valid state on the blockchain, thereby forcefully closing the channel. Hence, within a finite number of rounds, the dispute is resolved and the instant finality property is preserved. We apply a similar technique for virtual channels. The main difference is that a virtual channel is not funded on-chain. Hence, we first need to offload the virtual channel to the ledger. In other words, we first need to transform a virtual channel into a ledger channel by publishing its funding transaction on-chain. This process is discussed later in this section. Once the funding transaction is published, the dispute is handled in the same way as for ledger channels.

Close The closure of a virtual channel is done by updating the underlying ledger channels α and β according to the latest state of the virtual channel γ.st. To this end, each party $P \in \{A, B\}$ computes the new state for the ledger channel $\vec{\theta}_P := \{(c_P, \texttt{One–Sig}_{pk_P}), (\gamma.\mathsf{cash} - c_P, \texttt{One–Sig}_{pk_I})\}$ where c_P is the latest balance of P in γ. All parties update their

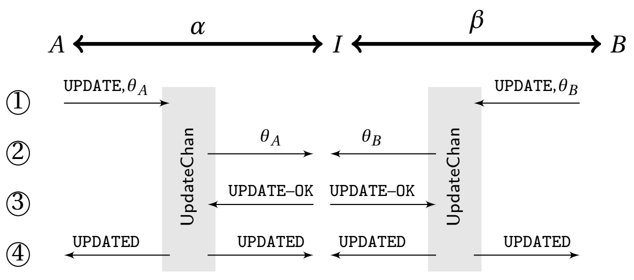


Figure 4.7: Modular close procedure of a virtual channel on top of two ledger channels α and β. For $P \in \{A, B\}$, $\vec{\theta}_P := \{(c_P, \texttt{One–Sig}_{pk_P}), (c_Q + \frac{\gamma.\text{fee}}{2}, \texttt{One–Sig}_{pk_I})\}$ where $\gamma.\text{st} = \left((c_P, \texttt{One–Sig}_{pk_P}), (c_Q, \texttt{One–Sig}_{pk_Q})\right)$.

ledger channels according to this state.

In a bit more detail, the closing procedure of a virtual channel proceeds as follows (see Figure 4.7). Each party P initiates an update of the underlying ledger channel with state $\vec{\theta}_P$ (step ①). Since both ledger channels must be updated synchronously, I waits for both parties to initiate the update procedure. Upon receiving the states from both parties (step ②), I checks that the states are consistent and if so, she agrees to the update of both ledger channels (step ③). Finally, after all parties have successfully revoked the previous ledger channel state, the virtual channel is considered to be closed.

In the pessimistic case (if the states $\vec{\theta}_A$ and $\vec{\theta}_B$ are inconsistent, revocation fails or I remains idle), parties must forcefully close their virtual channel by publishing the funding transaction (offloading) and closing the resulting ledger channel. This, together with the fact that I plays the role of the reacting party in its interactions with A and B, addresses the challenge (C1) as mentioned in Section 4.3.4.

Offload During the offload procedure, parties try to publish the funding transaction of the virtual channel γ which effectively transforms the virtual channel into a ledger channel. In a nutshell, during this procedure, parties try to publish the commit and split transactions of both underlying ledger channels and afterward the funding transaction of the virtual channel. In case offloading is prevented by some form of malicious behavior, parties can engage in the punishment procedure to ensure that they do not lose any funds.

Punish The concept of punishment in virtual channels is similar to that in ledger channels; namely in case that the latest state of a channel cannot be posted on the ledger, honest A or B are compensated by receiving all coins of the virtual channel while honest I will not lose coins. If the funding transaction of the virtual channel is posted on the ledger, the virtual channel is transformed into a ledger channel and parties can execute the regular punishment protocol for ledger channels. In addition to the ledger channel's punishment procedure, parties can punish if the funding transaction of γ cannot be published. Since this punishment, however, differs for each concrete instantiation, we will explain it in more detail for our protocols without validity and with validity in the following sections.

The offloading and punishment procedure together tackles challenge (C2) from Section 4.3.4.

CONCRETE INSTANTIATION WITHOUT VALIDITY

We now describe how the modular protocol explained above can be concretely instantiated with our construction for virtual channels without validity.

Create In our construction without validity, A and B must "prepare" the virtual channel during the setup procedure (step ③ in create of the modular protocol). This is done by executing the creation procedure of a regular ledger channel, i.e., they create a funding transaction with inputs tid_A and tid_B, as well as a commit and split transactions that spend the funding transaction. Once all three transactions are created, A and B sign them and exchange their signatures. Note that this corresponds to a normal channel opening, with the mere difference that the funding transaction is not published to the blockchain. In order to complete the virtual channel setup, A and B send the signed funding transaction to I who, upon receiving both signatures, sends her own signature on the transaction back to A and B. At this stage, the virtual channel is prepared, however, the creation is not completed yet. In order to finish the creation procedure, A, I, and B have to finish the update of their respective ledger channels. Once this is done, the virtual channel has been successfully created.

We illustrate the transaction structure prepared during the creation process in Fig. 4.8. The funding transaction of the virtual channel $\texttt{TX}_\texttt{f}$, which is generated during the create procedure, takes as input coins from both, the ledger channel α (represented by $\texttt{TX}_\texttt{s}^\texttt{A}$) and the ledger channel β (represented by $\texttt{TX}_\texttt{s}^\texttt{B}$). Both ledger channels jointly contribute a total of $2c + f$ coins so that c coins are later used to setup the virtual channel and the remaining $c + f$ coins are I's collateral and the fees paid to I for providing the service for A and B.[1] I's collateral and fees in the funding transaction $\texttt{TX}_\texttt{f}$ are the reason why I has to proactively monitor the virtual channel as she has an incentive to publish $\texttt{TX}_\texttt{f}$ in case any party misbehaves.

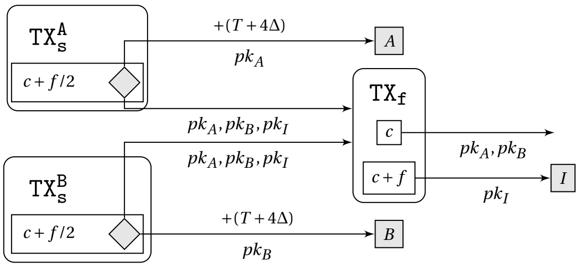


Figure 4.8: Funding of a virtual channel γ without validity. T upper bounds the number of off-chain communication rounds between two parties for any operation in the ledger channel.

[1] For simplicity we assume each of the parties contributes $f/2$ coins to I's total fees in addition to $c/2$ coins for funding the virtual channel.

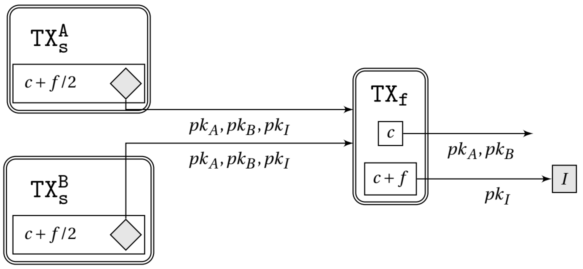


Figure 4.9: Transactions published after a successful offload.

Offload I is always able to offload the virtual channel by herself (i.e., without having to cooperate with another party) which guarantees that I can redeem her collateral at any time. We note that $P \in \{A, B\}$ can also initiate the offloading by publishing the commit and split transaction of their respective ledger channels. This forces I to publish the commit and split transactions of the respective other ledger channel, since I loses her collateral to P otherwise.

More precisely, if I wishes to offload the virtual channel γ and retrieve her collateral and fees, she can close both of her ledger channels with A and B (i.e., α and β) and publish the funding transaction of the virtual channel i.e., $\mathtt{TX}_\mathtt{f}$. This is possible as I is part of both ledger channels. A or B, on the other hand, are respectively part of only one ledger channel and hence they cannot offload the virtual channel individually. However, they can force I to offload by publishing the commit and split transactions of their respective channel with I (we will elaborate on this in the description of the punishment mechanism). Figure 4.9 illustrates the transactions that are posted on the blockchain in case of a successful offload. The figure shows that the split transactions of both underlying ledger channels have to be published such that eventually the funding transaction of the virtual channel can be published which completes the offloading procedure.

Punish Party $P \in \{A, B\}$ can punish I by taking all the coins on their respective ledger channels if the funding transaction of the virtual channel γ is not published on the ledger. In other words, it is I's responsibility to ensure that the state of her ledger channels with A and B are not updated while γ is open. Furthermore, upon one of the subchannels being closed, I must close the other subchannel in order to guarantee that both parties can post $\mathtt{TX}_\mathtt{f}$.

Let us now get into more details. Assume that A's ledger channel with I is closed, but the funding transaction $\mathtt{TX}_\mathtt{f}$ cannot be published on the blockchain. This means that I's channel with B (i.e., β) is still open or has been closed in a different state such that $\mathtt{TX}_\mathtt{f}$ cannot be published. In other words, Ingrid acted maliciously by wrongfully closing β in a different state or by not closing β at all. In this case, A must be able to get all the coins from her channel with Ingrid. This punishment works as follows: After A publishing the split transaction of α, I is given a certain time period to close her channel with B and publish the virtual channel's funding transaction $\mathtt{TX}_\mathtt{f}$. If I fails to do so in the prescribed time period, A receives all coins in her channel with I.

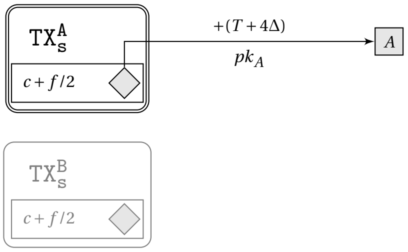

$\mathtt{TX}_{\mathtt{s}}^{\mathtt{A}}$
$c+f/2$
$+(T+4\Delta)$
pk_A
A
$\mathtt{TX}_{\mathtt{s}}^{\mathtt{B}}$
$c+f/2$

Figure 4.10: Transactions published after A successfully executed the punishment procedure. The grayed transaction $\mathtt{TX}_{\mathtt{s}}^{\mathtt{B}}$ indicates that this transaction has not been published.

We note that in this scenario, B (instead of I) might have been the malicious party by closing β in an outdated state, thereby leaving I no option to publish $\mathtt{TX}_{\mathtt{f}}$. However, in this case, I can punish B via the punishment mechanism of the underlying ledger channel and earn all the coins in β. Therefore, I will remain financially neutral as she gets punished by A but simultaneously compensated by B. Figure 4.10 illustrates the transactions that are posted on the blockchain in the case of A successfully executing the punishment mechanism. The case where B executes the punishment mechanism is analogous.

Note that so far we described our protocol without validity where the virtual channel can be offloaded by the intermediary whenever she wants. The drawback of this construction is that Ingrid needs to be proactive during the lifetime of the virtual channel, i.e., she has to constantly monitor the channel for potential misbehavior of Alice or Bob. This might be undesirable in scenarios where Ingrid plays the role of the intermediary in not just one but many different virtual channels at the same time (e.g. if Ingrid is a channel hub). For this reason, we developed an alternative solution which we call virtual channels with validity. In this solution each virtual channel has a predetermined time (which we call validity) which indicates until when the channel has to be closed again. If the channel is still open after this time, Ingrid has to become proactive in order to receive her collateral back. The obvious advantage of this approach is that Ingrid can remain inactive until the validity of a channel expires.

Concrete Instantiation With Validity

We now briefly present our virtual channel protocol with validity. We focus mainly on the creation of the virtual channel as this illustrates the main structural differences to our construction without validity.

Create Unlike the without validity case, the structure of the construction with validity is not symmetric (see Fig. 4.11). The output of the ledger channel between A and I is used as the input for the funding transaction of the virtual channel $\mathtt{TX}_{\mathtt{f}}$, whereas the output of the channel between B and I is used for the so-called refund transaction $\mathtt{TX}_{\mathtt{refund}}$.

A can create $\mathtt{TX}_{\mathtt{f}}$ on her own from the last state of her ledger channel with I. As a second step, A and B can already create the transactions required for the virtual channel γ. Additionally, I and B create the refund transaction which returns I's collateral if the

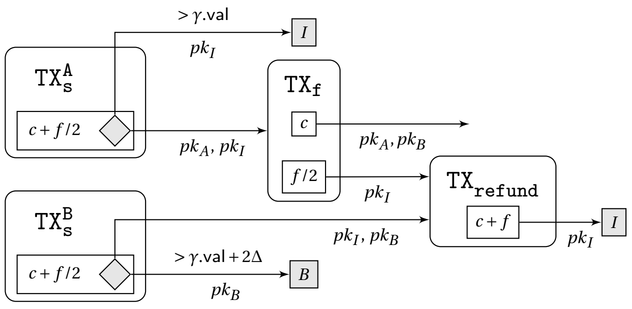


Figure 4.11: Funding of a virtual channel γ with validity $\gamma.\mathsf{val}$.

virtual channel is offloaded. Finally, the created transactions are signed in reverse order. In particular, B signs $\mathtt{TX_{refund}}$ so that I is ensured that she can publish it and receive her collateral and fees. Then, I signs $\mathtt{TX_f}$ and provides the signature to A, effectively authorizing her to publish $\mathtt{TX_f}$, thereby allowing A to offload the virtual channel.

Offload In our virtual channel with validity, only A can offload the virtual channel γ by publishing the commit and split transaction of her ledger channel with I. Although I and B are not able to offload the virtual channel, they have the guarantee that after round $\gamma.\mathsf{val}$ either the channel is offloaded or closed or they can punish A and get reimbursed.

Punish Recall that after a successful offload, the punishment mechanisms of generalized channels apply. We now discuss other malicious behaviors specific to this construction. In this protocol, only A can post the funding transaction of the virtual channel. If the virtual channel is not closed or offloaded by $\gamma.\mathsf{val}$, A is punished. A loses her coins to I and I loses her coins to B. Therefore, though B cannot offload the channel, he will get reimbursed from his ledger channel with I and I will get reimbursed regardless of whether the virtual channel is offloaded or not. At the time val, if the virtual channel is not honestly closed or the funding is not published, I submits the punishment transaction to reimburse her collateral. Therefore, at time $\mathsf{val} + \Delta$, either the punishment or the funding transaction is posted. If the virtual channel is offloaded, I can publish the refund transaction within Δ to get her coins back.

FURTHER DISCUSSION REGARDING OUR CONSTRUCTIONS

In the following, we present further considerations regarding our protocol, including remarks on concurrency, a discussion on how the protocol can be built on top of Lightning channels.

Concurrency When creating a virtual channel, we need to lock the underlying ledger channels α and β (i.e., no further updates can be made on the ledger channels as long as the virtual channel is open). This, however, is undesirable, because in most cases the ledger channels will have more coins available than what is needed for funding the virtual channel. We emphasize that this issue can be easily addressed (and hence supporting full concurrency) by using the channel splitting technique discussed in [13]. This means that before constructing the virtual channel Alice-Bob, parties would first *split* each underlying ledger channel off-chain in two channels: (i) one would contain the exact amount of coins for the virtual channel and (ii) the other one would contain the remaining coins that can be used in the underlying ledger channel.

Virtual channels over Lightning We will now discuss how our virtual channel constructions can be built on top of any ledger channel infrastructure that uses a *revocation/punishment* mechanism such as the Lightning Network [6]. The main complication arises from the fact that ledger channel constructions other than generalized channels require two commit transactions per channel state (one for each party). As depicted in Figure 4.12 (and unlike generalized channels in Figure 4.2), Alice and Bob each have a commit transaction $\mathtt{TX}_\mathtt{c}^\mathtt{A}$ and $\mathtt{TX}_\mathtt{c}^\mathtt{B}$ which spends the funding transaction $\mathtt{TX}_\mathtt{f}$ and distributes the coins. Therefore, in such channel constructions, it is a priori unclear which of these commit transactions will be posted and accepted on the blockchain (note that only one of them can be successfully published) and hence building applications (e.g., virtual channels) on top of such ledger channels becomes complex.

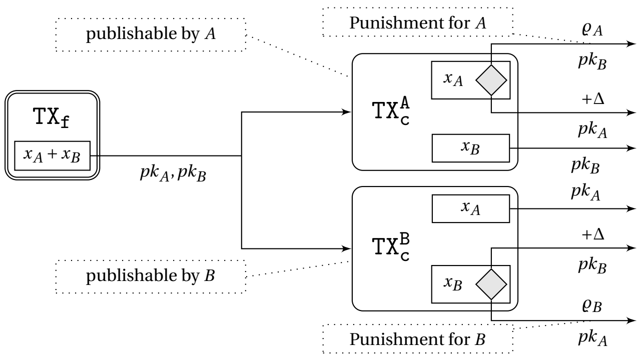


Figure 4.12: A Lightning style payment channel where A has x_A coins and B has x_B coins. Δ upper bounds the time needed to publish a transaction on a blockchain. condition ϱ_A represents the verification of A' revocation secret and h represents the verification of B' revocation secret.

In more detail, assume Alice and Bob want to build a virtual channel γ on top of their respective Lightning ledger channels with Ingrid, where both ledger channels consist of two commit transactions respectively (i.e., $(\mathtt{TX}_\mathtt{c}^\mathtt{A}, \mathtt{TX}_\mathtt{c}^\mathtt{IA})$ for the channel between Alice and Ingrid and $(\mathtt{TX}_\mathtt{c}^\mathtt{B}, \mathtt{TX}_\mathtt{c}^\mathtt{IB})$ for the channel between Bob and Ingrid). All three parties now have to make sure that the virtual channel can be funded (i.e., that the funding transaction of γ can be published to the blockchain) even in case of malicious behavior. To ensure this, parties have to prepare the funding transaction of γ with respect to all possible combinations of the commit transactions of the respective underlying ledger channels. Since there are four such combinations $((\mathtt{TX}_\mathtt{c}^\mathtt{A}, \mathtt{TX}_\mathtt{c}^\mathtt{B}), (\mathtt{TX}_\mathtt{c}^\mathtt{A}, \mathtt{TX}_\mathtt{c}^\mathtt{IB}), (\mathtt{TX}_\mathtt{c}^\mathtt{IA}, \mathtt{TX}_\mathtt{c}^\mathtt{B})$ and $(\mathtt{TX}_\mathtt{c}^\mathtt{IA}, \mathtt{TX}_\mathtt{c}^\mathtt{IB}))$, parties have to prepare four funding transactions for γ. Hence, updating such a virtual channel requires repeating the update procedure for all four funding transactions.

As generalized channels require only a single commit transaction per channel state building virtual channels on top of generalized channels offers a significant efficiency improvement in terms of off-chain communication complexity (see Section 4.5 for the detailed comparison).

4.4. SECURITY MODEL AND ANALYSIS

In order to model and prove the security of our virtual channel protocols, we use the global UC framework (GUC) [19] as in [13]. This framework allows for a global setup which we utilize to model a public blockchain. More precisely, our protocol uses a global ledger functionality $\widehat{\mathcal{L}}(\Delta, \Sigma)$, where Δ upper bounds the blockchain delay, i.e., the maximum number of rounds required to publish a transaction, and Σ is the signature scheme used by the blockchain. In this section, we only give a high-level idea behind our security analysis in the UC framework and refer the readers to the full version of the paper [18] for more details.

As a first step, we define the expected behavior of a virtual channel protocol in the form of an *ideal functionality* $\mathcal{F}_V$. The functionality defines the input/output behavior of a protocol, its impact on the global setup (e.g., ledger) and the possible ways an adversary can influence its execution (e.g., delaying messages). In order to prove that a concrete protocol is a secure virtual channel protocol, one must show that the protocol *emulates* the ideal functionality $\mathcal{F}_V$. This means that any attack that can be mounted on the protocol can also be mounted on the ideal functionality, hence the protocol is at least as secure as the ideal specification given by $\mathcal{F}_V$.

The proof of emulation consists of two steps. First, one must design a *simulator*, which simulates the actions of an adversary on the real-world protocol by interacting with the ideal functionality. Second, it must be shown that the execution of the real-world protocol being attacked by a real-world adversary is indistinguishable from the execution of the ideal functionality communicating with the constructed simulator. In UC, the ppt distinguisher who tries to distinguish these two executions is called the *environment*.

The main challenge when designing a simulator is to make sure that the environment sees transactions being posted on the ledger in the same round in both worlds. In addition, our simulator needs to ensure that the ideal functionality outputs the same set of messages in the same round as the protocol. We reduce the indistinguishability of the two executions to the security of the cryptographic primitives used in our protocol.

One of the advantages of using UC is its composability. In other words, one can use an ideal functionality in a black-box way in other protocols. This simplifies the process of designing new protocols as it allows to reuse existing results and enables modular protocol designs. We utilize this nice property of the UC framework and use the ideal functionality of the generalized channel from [13] when designing our virtual channel protocol.

Due to lack of space, we only mention the main security theorem and provide a high-level proof sketch here. We refer the reader to the full version of this paper [18] for the full proof.

Theorem 4.1. Let Σ be a signature scheme that is strongly unforgeable against chosen message attacks. Then for any ledger delay $\Delta \in \mathbb{N}$, the virtual channel protocol without validity as described in Section 4.3.5 working in $\mathscr{F}_{preL}(3,1)$-hybrid, UC-realizes the ideal functionality $\mathscr{F}_V(2)$.

We now give a proof sketch to show that the two properties (V1) Balance security and (V2) Offload with punish hold for honest parties. To this end, we analyze all possible cases in which the underlying ledger channels are maliciously closed, i.e., the cases when the virtual channel cannot be offloaded anymore. Note that if the virtual channel is offloaded, it is effectively transformed into a generalized ledger channel and satisfies the security properties of generalized channels.

If all parties behave honestly (V1) and (V2) hold trivially as I is always able to offload the virtual channel by publishing all transactions $\mathtt{TX}_\mathtt{s}^\mathtt{A}$, $\mathtt{TX}_\mathtt{s}^\mathtt{B}$ and $\mathtt{TX}_\mathtt{f}$. Furthermore, neither A nor B would ever lose their coins. Now consider the case where one of the underlying channels, e.g., the channel between B and I is closed in a different state such that $\mathtt{TX}_\mathtt{f}$ cannot be posted on the blockchain anymore (the case for the channel between A and I is analogous). As an honest A would not update her channel with I as long as the virtual channel is open, there are only two possible situations: (i) A is able to post $\mathtt{TX}_\mathtt{s}^\mathtt{A}$ which allows her to punish I (see Fig. 4.10), or (ii) I has maliciously closed her channel with A in an outdated and revoked state. In this case, A is able to punish I according to property (S3), i.e., instant finality with punish, of the underlying ledger channel (see Section 4.2 and Fig. 4.2 for more details on the punishment of the underlying channel). Therefore, (V2) is satisfied for A, since she can punish I and get financially compensated. Now let us analyze the maliciously closed channel between B and I, let us denote it β. If both parties are malicious, we do not need to prove anything as (V1) and (V2) should only hold for honest parties. In case B is honest, I must have closed β in an old state which would allow B to punish I. Hence (V2) holds and we do not need to prove (V1) as I is malicious. Analogously, if I is honest, malicious B must have closed β in an old state and hence I can punish B. Hence (V1) holds and we do not need to prove (V2) for malicious B). Hence, (V1) and (V2) hold for all honest parties.

4.5. PERFORMANCE EVALUATION

In this section, we first study the storage overhead on the blockchain as well as the communication overhead between users to use virtual channels. For each of these aspects, we evaluate both constructions (i.e., with and without validity) built on top of both generalized channels as well as Lightning channels and compare them. Finally, we evaluate the advantages of virtual channels over ledger channels in terms of routing communication overhead and fee costs. As testbed [20], the transactions are created in Python using the library `python-bitcoin-utils` and the Bitcoin *Script* language. To showcase compatibility and feasibility, we deployed these transactions successfully on the Bitcoin testnet.

4.5.1. COMMUNICATION OVERHEAD

We analyze the communication overhead imposed by the different operations, such as `CREATE`, `UPDATE`, `OFFLOAD` and `CLOSE`, by measuring the byte size of the transactions that need to be exchanged as well as the cost in USD necessary for posting the transactions that need to be published on-chain. The cost in USD is calculated by taking the price of 18803 USD per Bitcoin, and the average transaction fee of 104 satoshis per byte all of them at the time of writing. We detail in Table 4.2 the aforementioned costs measured for both virtual channel constructions building on top of generalized channels and on top of Lightning channels.

Operations	Generalized Channels										Lightning Channels									
	VC-NV					VC-V					VC-NV					VC-V				
	on-chain			off-chain		on-chain			off-chain		on-chain			off-chain		on-chain			off-chain	
	# txs	size	cost	# txs	size	# txs	size	cost	# txs	size	# txs	size	cost	# txs	size	# txs	size	cost	# txs	size
`CREATE`	0	0	0	7	2829	0	0	0	8	2803	0	0	0	16	7704	0	0	0	14	5722
`UPDATE`	0	0	0	2	695	0	0	0	2	695	0	0	0	8	2824	0	0	0	4	1412
`OFFLOAD`	5	2134	41.73	0	0	6	2108	41.22	0	0	3	1800	35.20	0	0	4	1778	34.77	0	0
`CLOSE` (opt)	0	0	0	4	1390	0	0	0	4	1390	0	0	0	4	1412	0	0	0	4	1412
`CLOSE` (pess)	7	2829	55.32	0	0	8	2803	54.81	0	0	4	2153	42.10	0	0	5	2131	41.67	0	0

Table 4.2: Evaluation of the virtual channels. For each operation we show: the number of on-chain and off-chain transactions (*# txs*) and their *size* in bytes. For on-chain transactions, *cost* is in USD and estimates cost of publish them on the ledger.

Perhaps the most relevant difference to ledger channels in practice is, in the `CREATE` and the optimistic `CLOSE` case, we do not have any on-chain transactions. This implies no on-chain fees for the opening and closing of virtual channels.

Virtual channels over generalized channels For the creation of a virtual channel (in Table 4.2, `CREATE` operation) on top of generalized channels, we need to update both ledger channels to a new state that can fund the virtual channel, requiring to exchange $2 \cdot 2$ transactions with 1494 (VC-NV) or 1422 (VC-V) bytes. Additionally, we need 640 bytes for $\mathtt{TX_f}$ (VC-NV) or 309 + 377 bytes for $\mathtt{TX_f}$ and $\mathtt{TX_{refund}}$ (VC-V). Finally, for both VC-NV and VC-V, we need the transactions representing the state of the the virtual channel itself which requires 431 bytes for $\mathtt{TX_c}$ and 264 bytes for $\mathtt{TX_s}$. This complete process results in 7 (VC-NV) or 8 (VC-V) transactions with a total of 2829 (VC-NV) or 2803 (VC-V) bytes. Forcefully closing (`CLOSE`(pess) operation) and offloading (`OFFLOAD` operation) requires the same set of transactions as with `CREATE`, minus the commitment and the split transaction (695 bytes) of the virtual channel in the latter case, both on-chain. Finally, we observe that the `UPDATE` and the optimistic `CLOSE`(opt) operation require 2 transactions (695 bytes) for both constructions, as they are designed as an update of a ledger channel.

Virtual channels over Lightning channels Building virtual channels on top of Lightning channels yields the following results. Instead of one commitment and one split transaction per ledger channel, we now need two commitment transactions per ledger channel, each of size 580 (VC-NV) or 546 (VC-V) bytes. Due to the fact that in both ledger channels, either commitment transaction can be published, we now need four $\mathtt{TX_f}$ of 640 bytes each (VC-NV) or two $\mathtt{TX_f}$ of 309 and four $\mathtt{TX_{refund}}$ of 377 bytes (VC-V). For every $\mathtt{TX_f}$, we need two commitment transactions of 353 bytes (in total, $8 \cdot 353$ in VC-NV or $4 \cdot 353$ in VC-V). For `OFFLOAD`, only one commitment transaction per ledger channel needs to be published, along with one $\mathtt{TX_f}$ (for VC-NV) and $\mathtt{TX_f}$ plus $\mathtt{TX_{refund}}$ (for VC-V). `CLOSE`(pess), needs to publish a commitment transaction in addition to `OFFLOAD`, resulting in 2153 (VC-NV) or 2131 (VC-V) bytes.

4.5.2. Comparison to Payment Channel Networks

In this section we compare virtual channels to multi-hop payments in a payment channel network (PCN). In a PCN, users route their payments via intermediaries. During the routing of a transaction tx, each intermediary party locks tx.cash coins as a "promise to pay" in their channels, a payment commitment that can technically be implemented as a Hash-Time Lock Contract (HTLC), e.g. as in the Lightning Network [6]. We now evaluate the difference in communication overhead and fee costs compared to virtual channels, summarize them in Table 4.3 and illustrate them in Fig. 4.13.

Routing communication overhead When performing a payment between Alice and Bob via an intermediary Ingrid in a multi-hop payment over generalized channels, the participants need to update both generalized channels with a "promise to pay", which require 2 transactions or 818 bytes per channel when implemented as HTLC. If they are successful, both generalized channels need to be updated again to "confirm the payment" (again, 2 transactions or 695 bytes per channel). This whole process results in 8 transactions or $2 \cdot 818 + 2 \cdot 695 = 3026$ off-chain bytes that need to be exchanged. Generically, if the parties want to perform n sequential payments, they need to exchange $8 \cdot n$ transaction with a total of $3026 \cdot n$ bytes.

Assume now that Alice and Bob were to perform the payment over a virtual channel without validity instead and that this virtual channel is not yet created. As shown in Table 4.2, they need to open the virtual channel for 2829 bytes, where they set the balance of the virtual channel already to the correct state after the payment, and then close it again for 1390 bytes, resulting in a total of 4219 off-chain bytes. However, if we again consider n sequential payments, the result would be $9 + 2 \cdot n$ transactions or $3524 + 695 \cdot n$ bytes, which supposes a reduction of $2331 \cdot n - 3524$ bytes with respect to relying on generalized channels only. This means that a virtual channel is already cheaper if only two (or more) sequential transactions are performed. We obtain similar results if we consider virtual channels with validity instead. For Lightning channels, the overhead is larger for both the multi-hop payment and the VC setting (Table 4.3).

Fee costs In a multi-hop payment tx in a PCN, the intermediary user Ingrid charges a base fee (BF) for being online and offering the routing service and relative fee (FR) for locking the amounts of coins (tx.cash) and changing the balance in the channel, so that

	Overhead in bytes			fees
	1 paym.	2 paym.	n payments	tx.cash in k payments
GC: PCN	3026	6052	$3026 \cdot n$	$\mathsf{BF} \cdot k + \mathsf{FR} \cdot \mathsf{tx.cash}$
GC: VC-NV	4219	4914	$3524 + 695 \cdot n$	$\mathsf{BF} + \mathsf{FR} \cdot \mathsf{tx.cash}$
GC: VC-V	4193	4888	$3498 + 695 \cdot n$	
LN: PCN	4776	9552	$4776 \cdot n$	$\mathsf{BF} \cdot k + \mathsf{FR} \cdot \mathsf{tx.cash}$
LN: VC-NV	9116	11940	$6292 + 2824 \cdot n$	$\mathsf{BF} + \mathsf{FR} \cdot \mathsf{tx.cash}$
LN: VC-V	5722	7134	$4310 + 1412 \cdot n$	

Table 4.3: Comparison of virtual channels (VC) to multi-hop payments (PCN) showing the overhead in bytes for a different number of payments and the difference in fees.

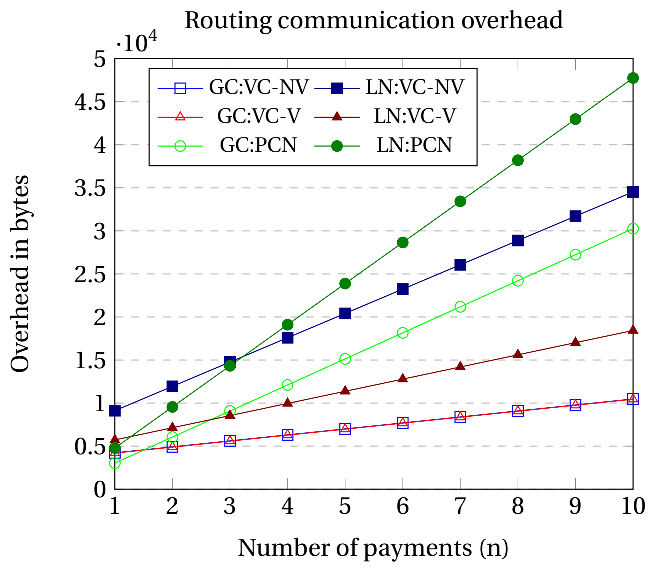


Figure 4.13: Pictorial illustration of Table 4.3.

$\mathsf{fee}(\mathsf{tx}) := \mathsf{BF} + \mathsf{FR} \cdot \mathsf{tx.cash}$. Note that at the time of writing, the fees are $\mathsf{BF} = 1$ satoshi and $\mathsf{FR} = 0.000001$.

In a virtual channel setting, $\gamma.\mathsf{Ingrid}$ can charge a base fee to collaborate to open and close the virtual channel, and also a relative fee to lock collateral coins in the virtual channel. However, no fees per payment are charged by Ingrid as she does not participate in them (and even does not know how many end-users performed)1. Let us now investigate the case of paying $\mathsf{tx.cash}$ in k micropayments of equal value. In PCN case, the total cost would be $\sum_{i=1}^{k} \mathsf{BF} + \mathsf{FR} \cdot \frac{\mathsf{tx.cash}}{k} = \mathsf{BF} \cdot k + \mathsf{FR} \cdot \mathsf{tx.cash}$. Whereas, in the virtual case, the parties first create a virtual channel γ with balance $\mathsf{tx.cash}$, and they will handle the micropayments in γ. Thereby, the cost would be only the opening cost of the virtual channel, for which we assumed $\mathsf{BF} + \mathsf{FR} \cdot \mathsf{tx.cash}$. Thus, if Alice and Bob would make more than one transaction, i.e., $k > 1$, it is beneficial to use virtual channels for reducing the fee costs by $\mathsf{BF} \cdot (k - 1)$.

Summary We find that the best construction in practice is the combination of virtual channels on top of generalized channels, as this yields the least overhead after only two or more sequential payments. However, building virtual channels over LN channels also yields less overhead than multi-hop PCN payments over LN.

4.6. Related Work

In this section, we position this work in the landscape of the literature for off-chain payments protocols.

Payment Channels Started from the Lightning Channels construction [6], the idea of 2-party payment channels has been largely used in academia and industry as a building block for more complex off-chain payment protocols. More recently, Aumayr et al. [13] have proposed a novel construction for 2-party payment channels that overcome some of the drawbacks of the original Lightning channels. While their benefit in terms of scalability is out of any doubt by now, payment channels are limited to payments between two users and consequently its overall utility.

A concurrent work [21] has also proposed a virtual channel construction over Bitcoin. However, their construction uses decreasing time-locks instead of a punishment mechanism in order to guarantee that only the latest state can be posted on the blockchain. As a consequence, their construction only allows a fixed number of transactions to be made during the lifetime of the virtual channel. This is quite restrictive as it requires users to close and open new virtual channels more frequently which goes against the purpose of virtual channels. Note that one cannot simply increase the time-lock as this would essentially lock the coins of the users for a longer period of time. Furthermore, our constructions are *generalized* virtual channels, i.e., they are not limited to just payments, but rather allow to run any Bitcoin script off-chain. In addition, we propose a modular approach compared to the monolithic construction in [21]. Finally, our work proposes two protocols, which each have their advantages in different use cases.

Payment Channel Networks (PCN) and Payment Channel Hub (PCH) A PCN allows a payment between two users that do not share a payment channel but are however connected through a path of payment channels. The notion of PCN started with the deployment of Lightning Network [6] for Bitcoin and Raiden Network [22] for Ethereum and has been widely studied in academia to research into different aspects such as privacy [7, 15], routing of payments [23], collateral management [24] and others. Similar to PCN, different constructions for PCH exist [25–27] that allow a payment between two users through a single intermediary, the payment hub. PCNs and PCHs, however, share the drawback that each payment between two users require the active involvement of the intermediary (or several intermediaries in the case of PCH), which reduces the reliability (e.g., the intermediary can go offline) and increases the cost of the payment (e.g., each intermediary charges a fee for the payment).

State Channels Several works [16, 17, 28, 29] have shown how to leverage the highly expressive scripting language available at Ethereum to construct (multi-party) state channels. A state channel allows the involved parties to carry out off-chain computations, possibly other than payments. Closer to our work, Dziembowski et al. [11] showed how to construct a virtual channel leveraging two payment channels defined in Ethereum. These approaches are, however, highly tight to the functionality provided by the Ethereum scripting language and their constructions cannot be reused in other cryptocurrencies. In this work, we instead show that virtual channels can be constructed from digital signatures and timelock mechanism only, which makes virtual channels accessible for virtually any cryptocurrency system available today.

4.7. CONCLUSION

Current PCNs route payments between two users through intermediate nodes, making the system less reliable (intermediaries might be offline), expensive (intermediaries charge a fee per payment), and privacy-invasive (intermediate nodes observe every payment they route). To mitigate this, recent work has introduced the concept of virtual channels, which involve intermediaries only in the creation of a bridge between payer and payee, who can later on independently perform arbitrarily many off-chain transactions. Unfortunately, existing constructions are only available for Ethereum, as they rely on its account model and Turing-complete scripting language.

In this work, we present the first virtual channel constructions that are built on the UTXO-model and require a scripting language supported by virtually every cryptocurrency, including Bitcoin. Our two protocols provide a tradeoff on who can offload the virtual channel, similar to the preemptible vs. non-preemptible virtual machines in the cloud setting. In other words, our virtual channel construction without validity is more suitable for intermediaries who can monitor the blockchain regularly, such as payment channel hubs, but can also close the virtual channel at anytime if desired. Our virtual channel protocol with validity however, is more suitable for light intermediaries who do not wish to be active during the lifetime of the virtual channel but cannot close the virtual channel before its validity has expired. We formalize the security properties of virtual channels in the UC framework, proving that our protocols constitute a secure realization thereof. We have prototyped our protocols and evaluated their efficiency: for n sequential payments in the optimistic case, they require $9+2\cdot n$ off-chain transactions for a total of $3524+695\cdot n$ bytes, with no on-chain footprint.

As mentioned in the introduction of this work, the task of designing secure virtual channels has been proven to be challenging even on a cryptocurrency like Ethereum [11] which supports smart contract execution. Unsurprisingly, this task becomes even more complex when building virtual channels for blockchains that support only a limited scripting language as it is not possible to take advantage of the full computation power of Turing complete smart contracts. Due to these significantly differing underlying assumptions (smart contracts vs. limited scripting languages), the virtual channel protocols based on Ethereum [11] and the protocols presented in this work are incomparable. We emphasize that we view our virtual channel constructions as complementary to the one presented in [11], as we do not aim to improve the construction of [11] but rather extend the concept of virtual channels to a broader class of blockchains.

We conjecture that it is possible to recursively build virtual channels on top of any two underlying channels (either ledger, virtual or a combination of them), requiring to adjust the timings for offloading channels: users of a virtual channel at layer k should have enough time to offload the (virtual/ledger) channels at layers 1 to $k-1$. Additionally, we envision that while virtual channels without validity might serve as a building block at any layer of recursion, virtual channels with validity period may be more suitable for the top layer as they have a predefined expiration time after which they would require to offload in any case all underlying layers. We plan to explore the recursive building of virtual channels in the near future. Additionally, we conjecture that virtual channels help with privacy, but we leave a formalization of this claim as interesting future work, as it involves a quantitative analysis that falls off the scope of this work.

Acknowledgments

This work was partly supported by the German Research Foundation (DFG) Emmy Noether Program *FA 1320/1-1*, by the *DFG CRC 1119 CROSSING* (project S7), by the German Federal Ministry of Education and Research (BMBF) *iBlockchain project* (grant nr. 16KIS0902), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the *National Research Center for Applied Cybersecurity ATHENE*, by the European Research Council (ERC) under the European Unions Horizon 2020 research (grant agreement No 771527-BROWSEC), by the Austrian Science Fund (FWF) through PROFET (grant agreement P31621) and the Meitner program (grant agreement M 2608-G27), by the Austrian Research Promotion Agency (FFG) through the Bridge-1 project PR4DLT (grant agreement 13808694) and the COMET K1 projects SBA and ABC, by the Vienna Business Agency through the project Vienna Cybersecurity and Privacy Research Center (VISP), by CoBloX Labs and by the ERC Project PREP-CRYPTO 724307.

REFERENCES

[1] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system,* (2009), http://bitcoin.org/bitcoin.pdf.

[2] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, *Sok: Layer-two blockchain protocols,* in *FC 2020* (2020) pp. 201–226.

[3] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, *Sok: Communication across distributed ledgers,* In press.

[4] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, *Sok: Consensus in the age of blockchains,* in *AFT 2019*, pp. 183–198.

[5] *Bitcoin wiki: Payment channels,* (2018), https://en.bitcoin.it/wiki/Payment_channels.

[6] J. Poon and T. Dryja, *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments,* (2016), draft version 0.5.9.2, available at https://lightning.network/lightning-network-paper.pdf.

[7] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, *Concurrency and privacy with payment-channel networks,* in *ACM CCS 17*, edited by B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu (ACM Press, 2017) pp. 455–471.

[8] U. Nisslmueller, K. Foerster, S. Schmid, and C. Decker, *Toward active and passive confidentiality attacks on cryptocurrency off-chain networks,* in *ICISSP*, edited by S. Furnell, P. Mori, E. R. Weippl, and O. Camp (2020) pp. 7–14.

[9] S. Tikhomirov, P. Moreno-Sanchez, and M. Maffei, *A quantitative analysis of security, anonymity and scalability for the lightning network,* in *IEEE EuroS&P* (2020) pp. 387–396.

[10] G. Kappos, H. Yousaf, A. M. Piotrowska, S. Kanjalkar, S. Delgado-Segura, A. Miller, and S. Meiklejohn, *An empirical analysis of privacy in the lightning network,* CoRR **abs/2003.12470** (2020), arXiv:2003.12470 .

[11] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, *Perun: Virtual payment hubs over cryptocurrencies,* in *IEEE SP* (2019) pp. 106–123.

[12] R. Canetti, *Universally composable security: A new paradigm for cryptographic protocols,* in *42nd FOCS* (IEEE Computer Society Press, 2001) pp. 136–145.

[13] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi, *Generalized bitcoin-compatible channels,* Cryptology ePrint Archive, Report 2020/476 (2020), https://eprint.iacr.org/2020/476.

[14] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Crypto-Currencies,* 1st ed. (O'Reilly Media, Inc., 2014).

[15] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, *Anonymous multi-hop locks for blockchain scalability and interoperability,* in *NDSS* (2019).

[16] S. Dziembowski, S. Faust, and K. Hostakova, *General state channel networks,* in *ACM CCS* (2018) pp. 949–966.

[17] S. Dziembowski, L. Eckey, S. Faust, J. Hesse, and K. Hostáková, *Multi-party virtual state channels,* in *EUROCRYPT 2019, Part I* (2019) pp. 625–656.

[18] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi, *Bitcoin-compatible virtual channels,* Cryptology ePrint Archive, Report 2020/554 (2020), https://eprint.iacr.org/2020/554.

[19] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, *Universally composable security with global setup,* in *TCC 2007,* LNCS, Vol. 4392, edited by S. P. Vadhan (Springer, Heidelberg, 2007) pp. 61–85.

[20] *Bitcoin-compatible virtual channels: Github repository,* (2020), https://github.com/utxo-virtual-channels/vc.

[21] M. Jourenko, M. Larangeira, and K. Tanaka, *Lightweight virtual payment channels,* in *CANS 2020* (2020) pp. 365–384.

[22] *Update from the raiden team on development progress, announcement of raidex,* (2017), https://tinyurl.com/z2snp9e.

[23] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, *Settling payments fast and private: Efficient decentralized routing for path-based transactions,* in *NDSS* (2018).

[24] C. Egger, P. Moreno-Sanchez, and M. Maffei, *Atomic multi-channel updates with constant collateral in bitcoin-compatible payment-channel networks,* in *ACM CCS* (2019) pp. 801–815.

[25] E. Tairi, P. Moreno-Sanchez, and M. Maffei, *A^2l: Anonymous atomic locks for scalability and interoperability in payment channel hubs,* In press.

[26] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, *TumbleBit: An untrusted bitcoin-compatible anonymous payment hub,* in *NDSS 2017* (The Internet Society, 2017).

[27] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, *Zerocash: Decentralized anonymous payments from bitcoin,* in *IEEE SP* (2014) pp. 459–474.

[28] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, *Sprites and state channels: Payment networks that go faster than lightning,* in *FC 2019* (2019) pp. 508–526.

[29] M. M. T. Chakravarty, S. Coretti, M. Fitzi, P. Gazi, P. Kant, A. Kiayias, and A. Russell, *Hydra: Fast isomorphic state channels,* IACR Cryptol. ePrint Arch. **2020**, 299 (2020).

5 POST-QUANTUM ADAPTOR SIGNATURES AND PAYMENT CHANNEL NETWORKS

Adaptor signatures, also known as scriptless scripts, *have recently become an important tool in addressing the scalability and interoperability issues of blockchain applications such as cryptocurrencies. An adaptor signature extends a digital signature in a way that a complete signature reveals a secret based on a cryptographic condition. It brings about various advantages such as (i) low on-chain cost, (ii) improved fungibility of transactions, and (iii) advanced functionality beyond the limitation of the blockchain's scripting language.*

In this work, we introduce the first post-quantum *adaptor signature, named* LAS*. Our construction relies on the standard lattice assumptions, namely Module-SIS and Module-LWE. There are certain challenges specific to the lattice setting, arising mainly from the so-called* knowledge gap *in lattice-based proof systems, that makes the realization of an adaptor signature and its applications difficult. We show how to overcome these technical difficulties without introducing additional on-chain costs.*

Our evaluation demonstrates that LAS *is essentially as efficient as an ordinary lattice-based signature in terms of both communication and computation. We further show how to achieve* post-quantum *atomic swaps and payment channel networks using* LAS.

This chapter is based on the paper "*Post-Quantum Adaptor Signatures and Payment Channel Networks*" by Esgin, M. F., Ersoy, O., and Erkin, Z. in ESORICS 2020, pp. 378–397.

5.1. Introduction

Blockchains are decentralized platforms run by miners, where each transaction on the blockchain can be seen as an application formed of some script(s). The scripting language of a blockchain defines potential functionalities that can be implemented on blockchain. Bitcoin, for example, consists of very few scripts, which restricts its use mainly into coin transactions. Ethereum, on the other hand, has a Turing-complete scripting language that enables users to run more advanced and complicated applications.

A user who wants to deploy and execute a transaction needs to pay a fee to the miners. The fee is determined by the storage and computational costs of running each script of the transaction. Thus, it is beneficial to handle some operations off-chain to reduce the on-chain fee paid to the miners. In this manner, Poelstra introduced the notion of *scriptless scripts* [1], which is later named as *adaptor signatures* [2, 3].

Adaptor signatures can be seen as an extension over a digital signature, where first a "pre-signature" is generated and its completion to a (full) signature reveals a secret based on a cryptographic condition. The conditions are defined over a hard relation such as the discrete log problem, and the complete signature reveals a witness matching with the statement embedded into the pre-signature. The verification of the signature is done in the same way as the original signature scheme. Thus, while the miners verify only the signature, parties involved in the signature generation can embed an additional condition.

The main advantages of adaptor signatures can be summarized as follows: (i) A significant reduction in on-chain costs, (ii) improved fungibility of transactions, and (iii) ability to incorporate complex conditions, which may otherwise be impossible to execute due to the limitation of the blockchain's scripting language. More specifically, if the condition is published on-chain separately, then it would incur additional storage and verification costs. At the same time, since the condition is embedded inside a signature, for the outsiders and miners the signature with a condition is indistinguishable from a regular one. This fungibility property is especially useful to hide payment channel network transactions among any other transactions [4]. Moreover, adaptor signatures enhance the functionality of blockchains with a limited scripting language. Since the condition embedded within the signature is not verified by miners, it is not limited by the blockchain's scripting language. These advantages have been utilized in payment channel networks [2, 4], atomic swaps [5], and discrete log contracts [6].

None of these works, however, provide security against powerful quantum computers as they rely on discrete-log-related assumptions. As evident, e.g., from NIST's efforts for standardization of *post-quantum* (i.e., quantum-resistant) algorithms [7], there is a major need for designing quantum-secure alternatives of currently deployed schemes. In fact, in the blockchain community, there are already significant efforts and considerations towards migrating to post-quantum cryptography. For example, Ethereum 2.0 Serenity upgrade [8] is planned to have an option for a post-quantum signature, Zcash developers plan to update their protocol with post-quantum alternatives when they are mature enough [9], and Hcash is building a post-quantum privacy-preserving blockchain [10].

Lattice-based cryptography, studied extensively in the last decades, is a promising candidate for post-quantum security. For example, Dilithium [11], which is based on standard lattice assumptions, is among the 2nd round signature candidates in NIST's post-quantum standardization process. Beyond basic cryptographic schemes such as encryption and signature, lattice-based cryptography also supports advanced schemes such as zero-knowledge proofs (ZKP), which play a crucial role in blockchain applications. For example, advanced ZKPs have recently been studied in [12, 13] and there are even recent efforts in constructing blockchain-specific applications based on lattice assumptions [14, 15].

Our contributions. In this work, we introduce the *first post-quantum* adaptor signature, LAS, in support of the efforts towards migration to post-quantum cryptography. Our construction relies on standard lattice assumptions, namely Module-LWE and Module-SIS, and is essentially as efficient as an ordinary lattice-based signature scheme based on the same assumptions. In particular, the signature scheme underlying LAS is a simplified version of Dilithium [11].

We further show how to realize post-quantum payment channel networks and atomic swaps using LAS. Our results show that these applications can be realized in the post-quantum setting without incurring an additional on-chain cost. The on-chain cost is effectively the cost of an ordinary lattice-based signature.

The main technical difficulties in constructing lattice-based adaptor signatures, as well as atomic swaps and payment channel networks, stem from the following two related facts. First, hard-to-find pre-images of lattice-based one-way functions, and in general user's secret keys, are required to have *small* coefficients in comparison to the system modulus q. In this case, a common technique used to hide user's secrets is rejection sampling, which is applied depending on the secret. As a result, in the setting of a payment channel network where a multi-party interaction is required with each user having his/her own secret, the realization of a secure construction demands a more careful analysis.

Secondly, *efficient* lattice-based zero-knowledge proofs underlying the (ordinary) signature scheme we employ have an inherent *knowledge (soundness) gap* (see, for example, [12, 16, 17]). That is, a witness extracted from a protocol interaction satisfies an *extended* relation R' whereas an honest user's secret satisfies a *stronger* relation R such that $R \subseteq R'$. Therefore, we need to adjust the security model carefully and also show that the extended guarantees are still meaningful and sufficient for practical applications. To this end, we extend the formal model of adaptor signatures introduced recently in [2], and show how to overcome the technical difficulties in our applications.

Organization of the paper. In Section 5.2, we present our security assumptions, lattice-based signatures and the rejection sampling technique as well as our extended formal definition for adaptor signatures. We introduce LAS, our adaptor signature, in Section 5.3, where the security and performance analyses and the effect of the knowledge gap are also given. We discuss the application of LAS to atomic swaps and payment channel networks in Section 5.4.

5.2. PRELIMINARIES

We define $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$ to be a cyclotomic ring of power-of-2 degree d for an odd modulus q. We denote by $\mathbb{S}_c$ the set of polynomials in $\mathcal{R}_q$ whose maximum absolute coefficient is at most $c \in \mathbb{Z}^+$. Similarly, $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$.

We denote by $\vec{I}_n$ the n-dimensional identity matrix. Vectors and matrices over $\mathcal{R}$ are denoted by lower-case and capital bold letters such $\vec{a}$ and $\vec{A}$, respectively. For a polynomial $f = f_0 + f_1 X + \cdots + f_{d-1} X^{d-1} \in \mathscr{R}$, we define the norms in the typical way: $\|f\| = \sqrt{\sum_{i=0}^{d-1} f_i^2}$, $\|f\|_\infty = \max_i |f_i|$ and $\|f\|_1 = \sum_{i=0}^{d-1} |f_i|$. For a vector $\vec{v} = (v_0, \dots, v_{s-1}) \in \mathscr{R}^s$ of polynomials with $s \geq 1$, we further define $\|\vec{v}\| = \sqrt{\sum_{i=0}^{s-1} \|v_i\|^2}$, $\|\vec{v}\|_1 = \sum_{i=0}^{s-1} \|v_i\|_1$, $\|\vec{v}\|_\infty = \max_i \|v_i\|_\infty$.

5.2.1. Security Assumptions: Module-SIS and Module-LWE

The security assumptions on which our constructions rely are the two well-known lattice problems, namely Module-SIS (M-SIS) and Module-LWE (M-LWE) [18]. They are generalizations of SIS [19] and LWE [20] problems, respectively. These problems are widely believed to resist attacks against powerful quantum adversaries. As in [11–13], we define below M-SIS in “Hermite normal form”, which is as hard as M-SIS with a completely random matrix $\vec{A}$.

Definition 5.1 (M-SIS$_{n,m,q,\beta_{\text{SIS}}}$)**.** Let $\vec{A}' \xleftarrow{\$} \mathscr{R}_q^{n\times(m-n)}$ and $\vec{A} = [\vec{I}_n \| \vec{A}']$. Given $\vec{A}$, M-SIS problem with parameters $m > n > 0$ and $0 < \beta_{\text{SIS}} < q$ asks to find a *short non-zero* $\vec{v} \in \mathscr{R}_q^m$ such that $\vec{A}\vec{v} = \vec{0}$ over $\mathscr{R}_q$ and $\|\vec{v}\| \leq \beta_{\text{SIS}}$.

5

We use a standard variant of M-LWE where both the error and secret coefficients are sampled uniformly from $\{-1, 0, 1\}$. This variant is commonly used in many recent proposals such as [12–14].

Definition 5.2 (M-LWE$_{\ell,m,q}$)**.** M-LWE problem with parameters $\ell, m > 0$ asks to distinguish between the following two cases: 1) $(\vec{A}, \vec{b}) \xleftarrow{\$} \mathscr{R}_q^{m\times\ell} \times \mathscr{R}_q^m$, and 2) $(\vec{A}, \vec{A}\vec{s} + \vec{e})$ for $\vec{A} \xleftarrow{\$} \mathscr{R}_q^{m\times\ell}$, a secret vector $\vec{s} \xleftarrow{\$} \mathbb{S}_1^\ell$ and an error vector $\vec{e} \xleftarrow{\$} \mathbb{S}_1^m$.

It is well-known that if the error and the secret coefficients are sampled from $\mathbb{S}_\gamma$ for $\gamma > 1$, then M-LWE problem gets harder. Therefore, M-LWE$_{\ell,m,q}$ hardness assumption implies that $\vec{t} = \vec{A}\vec{s} + \vec{e}$ is (computationally) indistinguishable from a uniformly random element of $\mathscr{R}_q^m$ when $\vec{s} \xleftarrow{\$} \mathbb{S}_\gamma^\ell$ and $\vec{e} \xleftarrow{\$} \mathbb{S}_\gamma^m$ for any $\gamma \geq 1$.

5.2.2. Lattice-Based Signature and Rejection Sampling

The (ordinary) signature part of our construction can be seen as a simplified version of Dilithium [11], which is a 2nd round signature candidate in NIST's post-quantum standardization process. This signature scheme itself is based on Lyubashevsky's signatures [16, 17]. In our construction, we do not employ the optimizations in Dilithium in order to simplify the presentation.

To make sure that the signature does not leak information about the secret key, we employ the rejection sampling technique from [16] as also done in Dilithium. The idea for this works as follows. Let $\vec{s} \in \mathscr{R}_q^k$ be a secret-dependant vector with $\|\vec{s}\|_\infty \leq p \in \mathbb{Z}^+$. In order to tie the security to M-SIS, we require the masked vector $\vec{z} = \vec{y} + \vec{s}$ to be short relative to q. Therefore, $\vec{y}$ cannot be sampled uniformly at random from $\mathscr{R}_q^k$. Instead, we sample $\vec{y} \xleftarrow{\$} \mathbb{S}_\gamma^k$ for $\gamma \approx kd \cdot p$. Then, we restart signing (i.e., reject $\vec{z} = \vec{y} + \vec{s}$) if $\|\vec{z}\|_\infty > \gamma - p$. It is easy to see that conditioned on $\vec{z}$ being accepted, the distribution of $\vec{z}$ is identical to the uniform distribution on $S_{\gamma-p}^k$. That is, the distribution of $\vec{z}$ is forced to be uniform in a box, and thus is (perfectly) simulatable using public information.

5.2.3. ADAPTOR SIGNATURES

In [2], an adaptor signature $\Pi_{R,\Sigma}$ is defined with respect to a hard relation R and a signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. A relation R with a language $L_R := \{Y \mid \exists y : (Y, y) \in R\}$ is said to be hard [21] if: (i) there exists a probabilistic polynomial time (PPT) generator $\mathsf{Gen}(1^n)$ that outputs $(Y, y) \in R$, (ii) for every PPT algorithm $\mathscr{A}$, given $Y \in L_R$, the probability of $\mathscr{A}$ outputting y is negligible. A signature scheme Σ is defined by three algorithms: (i) KeyGen generates a public-secret key pair (pk, sk), (ii) Sign produces a signature σ using the key (pk, sk) and message M, (iii) Verify verifies the correctness of a signature σ on a message M using a public key pk. Our underlying signature, Dilithium [11], is SUF-CMA (Strong existential unforgeability under chosen message attacks) secure.

In the lattice setting, we need to define two relations R, R' with $R \subseteq R'$. Here, R constitutes the relation for the statement-witness pairs output by Gen (i.e., those used by honest users) whereas R' is an *extended* relation that defines the relation for *extracted* witnesses. The reason for this extension is detailed in Section 5.3, and stems from the *knowledge/soundness gap* inherent in *efficient* lattice-based zero-knowledge proofs (see, e.g., the soundness definition in [13, Section 2.3]). We denote an adaptor signature scheme in this setting by $\Pi_{R,R',\Sigma}$, which extends the definition given in [2], and elaborate further below the reason why this extension is necessary.

Definition 5.3 (Adaptor Signature Scheme). An adaptor signature scheme $\Pi_{R,R',\Sigma}$ consists of four algorithms $(\mathsf{PreSign}, \mathsf{PreVerify}, \mathsf{Adapt}, \mathsf{Ext})$ defined below.

$\mathsf{PreSign}((pk, sk), Y, M)$: on input a key pair (pk, sk), a statement $Y \in L_R$ and a message $M \in \{0,1\}^*$, outputs a pre-signature $\hat{\sigma}$.

$\mathsf{PreVerify}(Y, pk, \hat{\sigma}, M)$: on input a statement $Y \in L_R$, a pre-signature $\hat{\sigma}$, a public key pk and a message $M \in \{0,1\}^*$, outputs a bit b.

$\mathsf{Adapt}((Y, y), pk, \hat{\sigma}, M)$: on input a statement-witness pair $(\hat{\sigma}, y)$, a public key pk, a pre-signature $\hat{\sigma}$ and a message $M \in \{0,1\}^*$, outputs a signature σ.

$\mathsf{Ext}(Y, \sigma, \hat{\sigma})$: on input a statement $Y \in L_R$, a signature σ and a pre-signature $\hat{\sigma}$, outputs a witness y such that $(Y, y) \in R'$, or $\perp$.

Note that an adaptor signature $\Pi_{R,R',\Sigma}$ also inherits KeyGen, Sign and Verify algorithms from the signature scheme Σ. The authors in [2] define the security properties for an adaptor signature: aEUF-CMA security, pre-signature adaptability and witness extractability. In addition, they extend the standard correctness definition of signature algorithms with pre-signature correctness, which states that an honestly generated pre-signature of a statement $Y \in L_R$ passes PreVerify and can be completed into a signature where the witness y can be extracted. We extend further the formal definitions of the security properties in [2], where $R = R'$ yields the setting in [2].

Definition 5.4 (aEUF-CMA security). An adaptor signature scheme $\Pi_{R,R',\Sigma}$ is aEUF-CMA secure if for every PPT adversary $\mathscr{A}$ there exists a negligible function $\nu[\lambda]$ such that $\Pr[\mathsf{aSignForge}_{\mathscr{A},\Pi_{R,R',\Sigma}}(\lambda) = 1] \leq \nu[\lambda]$, where the experiment $\mathsf{aSignForge}_{\mathscr{A},\Pi_{R,R',\Sigma}}$ is defined as follows:

```
$\mathsf{aSignForge}_{\mathcal{A},\Pi_{R,R',\Sigma}}(\lambda)$
1: $\mathcal{Q} := \emptyset$
2: $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3: $M^* \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot),\mathcal{O}_{pS}(\cdot,\cdot)}(pk)$
4: $(Y, y) \leftarrow \mathsf{Gen}(1^\lambda)$
5: $\hat{\sigma} \leftarrow \mathsf{PreSign}((pk, sk), Y, M^*)$
6: $\sigma \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot),\mathcal{O}_{pS}(\cdot,\cdot)}(\hat{\sigma}, Y)$
7: $\textbf{return } \big(M^* \notin \mathcal{Q} \wedge \mathsf{Verify}(pk, \sigma, M^*)\big)$

$\mathcal{O}_S(M)$
1: $\sigma \leftarrow \mathsf{Sign}((pk, sk), M)$
2: $\mathcal{Q} := \mathcal{Q} \cup \{M\}$
3: $\textbf{return } \sigma$

$\mathcal{O}_{pS}(M, Y)$
1: $\hat{\sigma} \leftarrow \mathsf{PreSign}((pk, sk), Y, M)$
2: $\mathcal{Q} := \mathcal{Q} \cup \{M\}$
3: $\textbf{return } \hat{\sigma}$
```

Definition 5.5 (Weak pre-signature adaptability)**.** An adaptor signature scheme $\Pi_{R,R',\Sigma}$ is *weak pre-signature adaptable* if for any message $M \in \{0,1\}^*$, any statement/witness pair $(Y, y) \in R$, any key pair $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and any pre-signature $\hat{\sigma} \leftarrow \{0,1\}^*$ with $\mathsf{PreVerify}(Y, pk, \hat{\sigma}, M) = 1$, we have $\Pr[\mathsf{Verify}(pk, \mathsf{Adapt}((Y, y), pk, \hat{\sigma}, M), M) = 1] = 1$.

We call our pre-signature adaptability definition *weak* because only statement-witness pairs satisfying R are guaranteed to be adaptable, and not those satisfying R'. This is similar to the *knowledge gap* of the ZKP underlying Dilithium, where the soundness only guarantees extraction of a witness from an *extended* relation. Therefore, pre-signature adaptability does not guarantee, for example, that an *extracted* witness can be used to adapt a pre-signature successfully (see Remark 5.10). This issue becomes effective in the applications of our adaptor signature, and we show how to overcome it in Section 5.4. Note that still the pre-signature $\hat{\sigma}$ in the above definition can be adversarially generated as in [2].

Definition 5.6 (Witness extractability)**.** An adaptor signature scheme $\Pi_{R,R',\Sigma}$ is *witness extractable* if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\nu[\lambda]$ such that the following holds: $\Pr[\mathsf{aWitExt}_{\mathcal{A},\Pi_{R,R',\Sigma}}(\lambda) = 1] \leq \nu[\lambda]$, where the experiment $\mathsf{aWitExt}_{\mathcal{A},\Pi_{R,R',\Sigma}}$ is defined as follows

```
$\mathsf{aWitExt}_{\mathcal{A},\Pi_{R,R',\Sigma}}(\lambda)$
1: $\mathcal{Q} := \emptyset$
2: $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3: $(M^*, Y) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot),\mathcal{O}_{pS}(\cdot,\cdot)}(pk)$
4: $\hat{\sigma} \leftarrow \mathsf{PreSign}((pk, sk), Y, M^*)$
5: $\sigma \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot),\mathcal{O}_{pS}(\cdot,\cdot)}(\hat{\sigma})$
6: $y' := \mathsf{Ext}(Y, \sigma, \hat{\sigma})$
7: $\textbf{return } (M^* \notin \mathcal{Q} \wedge (Y, y') \notin R'$
8:     $\wedge\, \mathsf{Verify}(pk, \sigma, M^*))$

$\mathcal{O}_S(M)$
1: $\sigma \leftarrow \mathsf{Sign}((pk, sk), M)$
2: $\mathcal{Q} := \mathcal{Q} \cup \{M\}$
3: $\textbf{return } \sigma$

$\mathcal{O}_{pS}(M, Y)$
1: $\hat{\sigma} \leftarrow \mathsf{PreSign}((pk, sk), Y, M)$
2: $\mathcal{Q} := \mathcal{Q} \cup \{M\}$
3: $\textbf{return } \hat{\sigma}$
```

Note that, in the above witness extractability definition, the adversary's winning condition is restricted to the extracted witness not being in R'. Since $R \subseteq R'$, $(Y, y') \notin R'$ implies that $(Y, y') \notin R$. Therefore, it is sufficient to ensure that R' is a hard relation, which

Table 5.1: Identifiers for LAS.

Notation	Explanation	Value
d	a power-of-2 ring dimension	256
$\mathcal{R}_q$	cyclotomic ring of degree d: $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$	$\log q \approx 24$
$\mathbb{S}_c$	the set of polynomials $f \in \mathcal{R}_q$ with $\|f\|_\infty \leq c$ for $c \in \mathbb{Z}^+$	
n	M-SIS rank	4
ℓ	M-LWE rank	4
$\mathcal{C}$	the challenge set and range of H: $\{c \in \mathcal{R} : \|c\|_1 = \kappa \wedge \|c\|_\infty = 1\}$	$\kappa = 60$
γ	maximum absolute coefficient of a masking randomness	$\kappa d(n+\ell)$
$(Y,y) \in \mathfrak{R}_{\vec{A}}$	the base relation with $[\vec{I}_n \,\|\, \vec{A}'] = \vec{A} \in \mathcal{R}_q^{n\times(n+\ell)}$: $(Y,y) = (\vec{t},\vec{r}) \in \mathfrak{R}_{\vec{A}}$ if $\vec{t} = \vec{A}\vec{r}$ and $\|\vec{r}\|_\infty \leq 1$	
$(Y,y) \in \mathfrak{R}'_{\vec{A}}$	the extended relation with $[\vec{I}_n \,\|\, \vec{A}'] = \vec{A} \in \mathcal{R}_q^{n\times(n+\ell)}$: $(Y,y) = (\vec{t},\vec{r}) \in \mathfrak{R}_{\vec{A}}$ if $\vec{t} = \vec{A}\vec{r}$ and $\|\vec{r}\|_\infty \leq 2(\gamma-\kappa)$	$\gamma > \kappa$

itself implies that R is also a hard relation. As a result, in our security assumptions, we make sure that R' is a hard relation.

5.3. LAS: An Efficient Adaptor Signature from Lattices

In this section, we describe our lattice-based adaptor signature, LAS. Let $\vec{A} = [\vec{I}_n \,\|\, \vec{A}'] \in R_q^{n\times(n+\ell)}$ for $\vec{A}' \xleftarrow{\$} R_q^{n\times\ell}$ and $\mathsf{H} : \{0,1\}^* \to \mathcal{C}$ be a hash function (modelled as a random oracle). We assume that the public parameters $pp = (\vec{A}, \mathsf{H})$ are publicly available and can be used by any algorithm. In practice, $\vec{A}'$ can be generated from a small seed using an extendable output function (modelled as a random oracle) as done in Dilithium [11]. The function $f_{\vec{A}}(\vec{x}) = \vec{A}\vec{x}$ over $\mathcal{R}_q$ is Ajtai's hash function [19] defined over module lattices where the matrix $\vec{A}$ is in Hermite normal form (HNF). It is clear that the function is additively homomorphic, and Ajtai [19] showed that it is one-way in the setting of SIS. In our case, the security is based on M-SIS (in HNF). Collision-resistance is also clear as a collision $(\vec{x},\vec{x}')$ yields an immediate M-SIS solution: $\vec{A}(\vec{x}-\vec{x}') = 0$.

In Table 5.1, we first summarize the identifiers used for LAS, where the hard relations R, R' are given by $\mathfrak{R}_{\vec{A}}, \mathfrak{R}'_{\vec{A}}$ with $\mathfrak{R}_{\vec{A}} \subseteq \mathfrak{R}'_{\vec{A}}$. The statement-witness generation Gen for $\mathfrak{R}_{\vec{A}}$ runs exactly as KeyGen. It is easy to see that if M-SIS$_{n,n+\ell+1,q,\beta}$ for $\beta = 2\gamma d(n+\ell)$ is hard, then $\mathfrak{R}_{\vec{A}}$ and $\mathfrak{R}'_{\vec{A}}$ are hard relations. This is because if one can find $\vec{r}$ such that $(\vec{t},\vec{r}) \in \mathfrak{R}'_{\vec{A}}$ for a random $\vec{t}$, then $[\vec{A} \,\|\, \vec{t}] \cdot \begin{pmatrix} \vec{r} \\ -1 \end{pmatrix} = 0$. Hence, $\begin{pmatrix} \vec{r} \\ -1 \end{pmatrix}$ is a solution to M-SIS$_{n,n+\ell+1,q,\beta}$ for $\beta = 2\gamma d(n+\ell)$ since $\|\vec{r}\| \leq \beta$.

We present the ordinary signature procedures in Algorithm 5, and then the procedures for the adaptor signature in Algorithm 6. The idea for the signature is similar to the Schnorr signature [22] with the main difference being the use of rejection sampling at Step 11. This is the so-called "Fiat-Shamir with Aborts" technique [16, 17].

In the adaptor signature part in Algorithm 6, PreSign and PreVerify operate very similar to Sign and Verify, respectively. The main issue is that the signer may not know

Algorithm 5 Lattice-Based Signature

```
1: procedure KeyGen():                              ▷ same as Gen
2:     $\vec{r} \xleftarrow{\$} \mathbb{S}_1^{n+\ell}$
3:     $\vec{t} = \vec{A}r$
4:     return $(pk, sk) = (\vec{t}, \vec{r})$
5: end procedure

6: procedure Sign($(pk, sk), M$):
7:     $\vec{y} \xleftarrow{\$} \mathbb{S}_\gamma^{n+\ell}$
8:     $\vec{w} = \vec{A}y$
9:     $c = \mathsf{H}(pk, \vec{w}, M)$
10:    $\vec{z} = \vec{y} + c\vec{r}$ where $\vec{r} := sk$
11:    if $\|\vec{z}\|_\infty > \gamma - \kappa$, then Restart
12:    return $\sigma = (c, \vec{z})$
13: end procedure

14: procedure Verify($pk, \sigma, M$):
15:    Parse $(c, \vec{z}) := \sigma$
16:    if $\|\vec{z}\|_\infty > \gamma - \kappa$, then return 0
17:    $\vec{w}' = \vec{A}z - c\vec{t}$ where $\vec{t} := pk$
18:    if $c \neq \mathsf{H}(pk, \vec{w}', M)$, then return 0
19:    return 1
20: end procedure
```

(at the time of running PreSign) the witness y to the statement Y, and yet for many applications in practice (such as payment channel networks), one would want to make sure that having access only to the signature (but not the pre-signature) does not reveal any information on the witness y.

To this end, we need to modify the rejection sampling step. Even though the signer does not know the witness y, he does know how it is supposed to be generated in an honest run. Therefore, he knows that the maximum absolute coefficient of any honestly-generated witness is at most 1 (recall that Gen runs exactly as KeyGen). Since we have $\vec{z} = \vec{y} + c\vec{r} + \vec{r}'$ for $\vec{r}' := y$ in an honestly-generated full signature, we know that the secret-dependant part $c\vec{r} + \vec{r}'$ has infinity norm at most $\kappa + 1$. Therefore, the signer artificially performs a stronger rejection sampling step in PreSign, where $\|\hat{z}\|_\infty \leq \gamma - \kappa - 1$ is required. This ensures that even when the witness is added to the response in Adapt, the response $\vec{z}$ still satisfies the rejection sampling condition in Sign, and thus remains publicly simulatable, i.e., no secret information including the witness is revealed.

In fact, there are further reasons for this important modification. One is in regards to adaptability. If the rejection sampling in PreSign is done exactly as in Sign, then verification of an adapted pre-signature (i.e., output of Adapt) via Verify may not succeed as the infinity norm condition may be violated due to the addition of $\vec{r}' := y$. Another reason comes from the security analysis. In order to be able to simulate the outputs of both Sign and PreSign, this change to rejection sampling plays a crucial role.

Let us summarize the following two facts as we will make use of them repeatedly in the security proofs.

Fact 5.7. We can see that $\|c\vec{r}\|_\infty \leq \kappa$ since $\|c\|_1 \leq \kappa$ and $\|\vec{r}\|_\infty \leq 1$. Therefore, both $\hat{z}$ in PreSign and $\vec{z}$ in Sign can be simulated publicly as they follow uniform distributions on $\mathbb{S}_{\gamma-\kappa-1}^{n+\ell}$ and $\mathbb{S}_{\gamma-\kappa}^{n+\ell}$, respectively, due to the rejection sampling.

Fact 5.8. Assuming the hardness of M-LWE$_{\ell,n,q}$, the result of $\vec{A}x$ is (computationally) indistinguishable from a uniformly random element in $\mathcal{R}_q^n$ whenever $\vec{x} \xleftarrow{\$} \mathbb{S}_c^{n+\ell}$ for some

Algorithm 6 LAS: Lattice-Based Adaptor Signature

```
1: procedure PreSign((pk, sk), Y, M):
2:     $\vec{y} \xleftarrow{\$} \mathbb{S}_{\gamma}^{n+\ell}$
3:     $\vec{w} = \vec{A}y$
4:     $c = \mathsf{H}(pk, \vec{w} + \vec{t}', M)$ for $\vec{t}' := Y$
5:     $\vec{z} = \vec{y} + c\vec{r}$ where $\vec{r} := sk$
6:     if $\|\vec{z}\|_\infty > \gamma - \kappa - 1$, then Restart
7:     return $\hat{\sigma} = (c, \vec{z})$
8: end procedure

9: procedure PreVerify(Y, pk, σ̂, M):
10:    Parse $(c, \vec{z}) := \hat{\sigma}$ and $\vec{t}' := Y$
11:    if $\|\vec{z}\|_\infty > \gamma - \kappa - 1$ then
12:        return 0
13:    end if
14:    $\vec{w}' = \vec{A}\vec{z} - c\vec{t}$ where $\vec{t} := pk$
15:    if $c \neq \mathsf{H}(pk, \vec{w}' + \vec{t}', M)$ then
16:        return 0
17:    end if
18:    return 1
19: end procedure

20: procedure Adapt((Y, y), pk, σ̂, M):
21:    if PreVerify$(Y, pk, \hat{\sigma}, M) = 0$ then
22:        return ⊥
23:    end if
24:    Parse $(c, \vec{z}) := \hat{\sigma}$ and $\vec{r}' := y$
25:    return $\sigma = (c, \vec{z} + \vec{r}')$
26: end procedure

27: procedure Ext(Y, σ, σ̂):
28:    Parse $(c, \vec{z}) := \sigma$ and $(\hat{c}, \vec{\hat{z}}) := \hat{\sigma}$
29:    Parse $\vec{t}' := Y$
30:    $\vec{s} = \vec{z} - \vec{\hat{z}}$
31:    if $\vec{t}' \neq \vec{A}s$, then return ⊥
32:    return $\vec{s}$
33: end procedure
```

$c \geq 1$. We can see this by realizing that $\vec{A}x = [\vec{I}_n \,\|\, \vec{A}'] \cdot \begin{pmatrix} \vec{x}_0 \\ \vec{x}_1 \end{pmatrix} = \vec{x}_0 + \vec{A}'\vec{x}_1$. This is an M-LWE instance with the secret vector $\vec{x}_1 \in \mathbb{S}_c^{\ell}$ and the error vector $\vec{x}_0 \in \mathbb{S}_c^{n}$.

Note that there is a *knowledge gap* between a witness used by an honest user and a witness extracted by Ext for a statement Y. In particular, an honest user's witness $y = \vec{r}$ satisfies $\|\vec{r}\|_\infty \leq 1$ (i.e., $(Y, y) \in \mathfrak{R}_{\vec{A}}$), whereas an extracted witness $y' = \vec{r}'$ is only guaranteed to satisfy $\|\vec{r}'\|_\infty \leq 2(\gamma - \kappa)$ (i.e., $(Y, y') \in \mathfrak{R}'_{\vec{A}}$). Such a knowledge gap is inherent in the existing *efficient* lattice-based zero-knowledge proofs such as the one underlying Dilithium. However, we emphasize that this knowledge gap does not raise a security concern as our hardness assumptions require that finding even a witness as big as an extracted witness is still hard, which itself implies that finding an honest user's witness is also hard. In the next section, we study the security aspects more rigorously.

5.3.1. SECURITY ANALYSIS

Pre-signature correctness follows via a straightforward investigation. In the following sequence of lemmas, we prove the security properties.

Lemma 5.9 (Weak pre-signature adaptability)**.** LAS satisfies weak pre-signature adaptability with respect to the relation $\mathfrak{R}_{\vec{A}}$ given in Table 5.1.

Proof. Let $\hat{\sigma} = (c, \vec{z})$ be a valid pre-signature with PreVerify$(Y, pk, \hat{\sigma}, M) = 1$ and $y = \vec{r}' \in \mathbb{S}_1^{n+\ell}$ be a witness corresponding to Y. Note that $\|\vec{z}\|_\infty \leq \gamma - \kappa - 1$ since $\hat{\sigma}$ is valid. Then,

$\mathsf{Adapt}((Y,y),pk,\hat{\sigma},M) = (c,\vec{\hat{z}}+\vec{r}') =: (c,\vec{z}) = \sigma$. Now, we have

$$\|\vec{z}\|_\infty = \|\vec{\hat{z}}+\vec{r}'\|_\infty \le \|\vec{\hat{z}}\|_\infty + \|\vec{r}'\|_\infty = (\gamma-\kappa-1)+1 = \gamma-\kappa. \tag{5.1}$$

We further have

$$\begin{aligned}\mathsf{H}(pk,\vec{A}z-c\vec{t},M) &= \mathsf{H}(pk,\vec{A}(\vec{\hat{z}}+\vec{r}')-c\vec{t},M) = \mathsf{H}(pk,\vec{A}\vec{\hat{z}}-c\vec{t}+\vec{A}\vec{r}',M)\\ &= \mathsf{H}(pk,\vec{A}\vec{\hat{z}}-c\vec{t}+\vec{t}',M) = c.\end{aligned} \tag{5.2}$$

From (5.1) and (5.2), it follows that σ is valid, i.e., $\mathsf{Verify}(pk,\sigma,M) = 1$. □

Remark 5.10. Observe in the proof of Lemma 5.9 that we crucially rely on the fact that for a witness $y = \vec{r}'$ in $\mathfrak{R}_{\vec{A}}$, we have $\|\vec{r}'\|_\infty \le 1$. An *extracted* witness $\vec{s}$ does not necessarily obey this rule as the relation $\mathfrak{R}'_{\vec{A}}$ only requires $\|\vec{s}\|_\infty \le 2(\gamma-\kappa)$. Therefore, extra care needs to be taken when dealing with the cases where an extracted witness is used to adapt a pre-signature.

Lemma 5.11 (Witness extractability)**.** If M-LWE$_{\ell,n,q}$ and M-SIS$_{n,n+\ell+1,q,\beta}$ for $\beta = 2\gamma\sqrt{d(n+\ell)}$ are hard, then LAS is witness extractable in the random oracle model.

Proof. Here, we only investigate the case that the signature output by the adversary shares the same challenge with the pre-signature. The other case (where the two challenges are distinct) can be proven exactly as in **Case 2** of the proof of Lemma 5.12 because how Y is generated is irrelevant for that case.

For a given pair of public key and statement $(pk,Y) = (\vec{t},\vec{t}')$ and a message M, let $\hat{\sigma} = (c,\vec{\hat{z}})$ and $\sigma = (c,\vec{z})$ be a valid pre-signature and a valid signature, respectively. Then, from the corresponding verification algorithms (i.e., Verify and PreVerify), we have $\mathsf{H}(pk,\vec{A}z - c\vec{t},M) = \mathsf{H}(pk,\vec{A}\vec{\hat{z}}-c\vec{t}+\vec{t}',M)$. Since H is modelled as a random oracle, this holds only when $\vec{A}z-c\vec{t} = \vec{A}\vec{\hat{z}}-c\vec{t}+\vec{t}'$, which implies that $\vec{A}z-\vec{A}\vec{\hat{z}} = \vec{A}(\vec{z}-\vec{\hat{z}}) = \vec{t}'$. It is easy to see that $\|\vec{z}-\vec{\hat{z}}\|_\infty \le 2(\gamma-\kappa)$. Therefore, for the output $\vec{s} = \vec{z}-\vec{\hat{z}}$ of $\mathsf{Ext}(Y,\sigma,\hat{\sigma})$, we have $(\vec{t}',\vec{s}) \in \mathfrak{R}'_{\vec{A}}$. Note also that $\vec{s}$ is non-zero since $\vec{t}'$ is non-zero except for a negligible probability. □

Lemma 5.12 (Unforgeability)**.** If M-SIS$_{n,n+\ell+1,q,\beta}$ for $\beta = 2\gamma\sqrt{d(n+\ell)}$ and M-LWE$_{\ell,n,q}$ are hard, then LAS is aEUF-CMA secure in the random oracle model.

Proof. First, from the assumptions in the statement, we know that

1. both $\mathfrak{R}_{\vec{A}}$ and $\mathfrak{R}'_{\vec{A}}$ are hard relations,
2. any public key output by KeyGen and any statement output by Gen is indistinguishable from a uniformly random element in $\mathcal{R}_q^n$ due to Fact 5.8.

Let $\mathscr{F}$ be a PPT adversary who wins the aEUF-CMA security game with non-negligible probability. We will build an adversary $\mathscr{S}$ that solves M-SIS$_{n,n+\ell+1,q,\beta}$. Let $\beta = 2\gamma\sqrt{d(n+\ell)}$ and $\vec{B} = [\vec{I}_n \,\|\, \vec{A}' \,\|\, \vec{a}] \in \mathcal{R}_q^{n\times(n+\ell+1)}$ for $\vec{A}' \xleftarrow{\$} \mathcal{R}_q^{n\times\ell}$ and $\vec{a} \xleftarrow{\$} \mathcal{R}_q^n$. Assume that $\mathscr{S}$ wants to solve M-SIS w.r.t. $\vec{B}$. Let $\vec{A}$ denote $[\vec{I}_n \,\|\, \vec{A}']$.
Setup. $\mathscr{S}$ sets $\vec{A}$ together with some hash function H as the public parameters. It is clear that $\vec{A}$ has the correct distribution. Then, it sets $pk = \vec{t} = \vec{B}r$ where $\vec{r} = \begin{pmatrix}\vec{r}'\\ 1\end{pmatrix}$ for

$\vec{r}' \xleftarrow{\$} \mathbb{S}_1^{n+\ell}$. $\mathscr{S}$ sends pk to $\mathscr{F}$. By M-LWE$_{\ell,n,q}$, pk is indistinguishable from a public key output by KeyGen since $\vec{B}r = \vec{A}r' + \vec{a}$ looks uniformly random as $\vec{A}r'$ does. Note also that $\vec{t} = pk$ is non-zero with overwhelming probability.

Oracle simulation. For $\mathscr{O}_{\mathsf{S}}(M)$, $\mathscr{S}$ picks $\vec{z} \xleftarrow{\$} \mathbb{S}_{\gamma-\kappa}^{n+\ell}$ and $c \xleftarrow{\$} \mathscr{C}$, and programs the random oracle such that $c = \mathsf{H}(pk, \vec{A}z - c\vec{t}, M)$. If the input of H has been queried before, $\mathscr{S}$ aborts. Otherwise, $\mathscr{S}$ returns $\sigma = (c, \vec{z})$. The simulated output is indistinguishable from a real one due to Fact 5.7.

For $\mathscr{O}_{\mathsf{pS}}(M, Y)$, the simulator picks $\vec{\hat{z}} \xleftarrow{\$} \mathbb{S}_{\gamma-\kappa-1}^{n+\ell}$ and $c \xleftarrow{\$} \mathscr{C}$, and programs the random oracle such that $c = \mathsf{H}(pk, \vec{A}\hat{z} - c\vec{t} + \vec{t}', M)$ for $\vec{t}' := Y$. If the input of H has been queried before, $\mathscr{S}$ aborts. Otherwise, the simulator returns $\hat{\sigma} = (c, \vec{\hat{z}})$. The simulated output is indistinguishable from a real one due to Fact 5.7.

In both cases, the probability of an abort is negligible as $\mathscr{F}$ can make at most polynomially many queries to H.

Forgery. $\mathscr{F}$ returns the target message M^* to $\mathscr{S}$. $\mathscr{S}$ sets $Y = -\vec{a}$ and computes a pre-signature $\hat{\sigma}^* = (c^*, \vec{\hat{z}}^*)$ using the simulation method above. $\mathscr{S}$ sends $(Y, \hat{\sigma}^*)$ to $\mathscr{F}$. Again, note that Y is indistinguishable from a real output by Gen, and $\hat{\sigma}^*$ is indistinguishable from a real output of PreSign. Finally, $\mathscr{F}$ returns a forged signature $\sigma = (c, \vec{z})$ on M^*.

Case 1 $(c^* = c)$**:** If this is the case, then as shown in the proof of Lemma 5.11, $\mathscr{S}$ can extract a witness to $\mathfrak{R}'_{\vec{A}}$. That is, $\mathscr{S}$ gets $(Y, y) \in \mathfrak{R}'_{\vec{A}}$ with $\vec{s}' := y$, which implies that $\vec{A}s' = -\vec{a}$ (since $Y = -\vec{a}$) and $\|\vec{s}'\|_\infty \le 2(\gamma - \kappa)$. This is equivalent to $\vec{B}\vec{s} = \vec{0}$ for $\vec{s} = \begin{pmatrix} \vec{s}' \\ 1 \end{pmatrix}$. Note that $\|\vec{s}\| \le \beta$. Hence, $\mathscr{S}$ finds a solution to M-SIS$_{n,n+\ell+1,q,\beta}$.

Case 2 $(c^* \ne c)$**:** In this case, we know that the forged signature's challenge comes from a random oracle query output (with overwhelming probability). Therefore, we can use a standard rewinding argument as in [23], where $\mathscr{S}$ rewinds $\mathscr{F}$ to get another forgery $\sigma' = (c', \vec{z}')$ such that $c' \ne c$ and $\mathsf{H}(pk, \vec{A}\vec{z}' - c'\vec{t}, M^*) = \mathsf{H}(pk, \vec{A}\vec{z} - c\vec{t}, M^*)$. Therefore, we have

$$\vec{A}\vec{z}' - c'\vec{t} = \vec{A}\vec{z} - c\vec{t} \quad \Longleftrightarrow \quad \vec{A}\left(\vec{z}' - \vec{z}\right) = (c' - c)\vec{t}. \tag{5.3}$$

Since $c' \ne c$, we have $\vec{z}' - \vec{z} \ne 0$. The above equation (5.3) can be equivalently written as

$$\vec{B}\begin{pmatrix} \vec{z}' - \vec{z} \\ 0 \end{pmatrix} = (c' - c)\vec{t}. \tag{5.4}$$

Now recalling that $\vec{t} = \vec{B}r$, we also have

$$(c' - c)\vec{t} = \vec{B} \cdot (c' - c)\vec{r}. \tag{5.5}$$

Subtracting (5.3) from (5.5), we get

$$\vec{B}\left[(c' - c)\vec{r} - \begin{pmatrix} \vec{z}' - \vec{z} \\ 0 \end{pmatrix}\right] = \vec{0}. \tag{5.6}$$

Recalling that the last coordinate of $\vec{r}$ is 1, i.e., non-zero, the above gives a non-trivial solution to M-SIS$_{n,n+\ell+1,q,\beta}$. Here note that $\|\vec{z}' - \vec{z}\| \le 2(\gamma - \kappa)\sqrt{d(n+\ell)} < \beta$ and $\|(c' - c)\vec{r}\| \le 2\kappa\sqrt{d(n+\ell+1)}$. Since $\gamma \gg \kappa$, the total norm of the M-SIS solution remains below $\beta = 2\gamma\sqrt{d(n+\ell)}$. □

5.3.2. Parameter Setting and Performance Analysis

First, we set $\gamma = \kappa d(n+\ell)$ so that the average number of restarts in Sign and $\mathsf{PreSign}$ is about $e < 3$. Then, we set $d = 256$ and $\kappa = 60$, which ensures that the challenge set $\mathscr{C}$ has more than 2^{256} elements. Finally, in order to meet the M-SIS$_{n,n+\ell+1,q,\beta}$ and M-LWE$_{\ell,n,q}$ security requirements for $\beta = 2\gamma\sqrt{d(n+\ell)}$, we set $n = \ell = 4$ and $q \approx 2^{24}$. Only the size of the modulus q is important, and therefore the concrete value can be chosen to allow fast computation such as Number Theoretic Transformation (NTT).

In estimating the practical security of M-SIS and M-LWE, we follow the methodology outlined in [24, Section 3.2.4] and measure the practical hardness in terms of "root Hermite factor" δ. This parameter setting yields $\delta < 1.0045$ for both M-SIS and M-LWE. $\delta \approx 1.0045$ has been used in recent works, e.g., [12–14] for targeting 128-bit post-quantum security. From here, we can compute the concrete signature length as

$$|\sigma| = d(n+\ell)\log(2\gamma)/8 + 32 \text{ bytes} \approx 3210 \text{ bytes}. \tag{5.7}$$

This length is slightly larger than the size of Dilithium (2701 bytes) [11] with recommended parameters. The main reason is because we do not employ the optimizations for ease of presentation.

In terms of the computational efficiency, the operations performed in LAS are almost identical to those in Dilithium. Thus, hundreds of signing (and even more verification) can be done per second on a standard PC as shown in [11, Table 2].

5.4. Applications

In this section, we present two blockchain applications of our adaptor signature, namely atomic swaps and payment channel networks. To match with the existing adaptor signature applications, we assume an Unspent Transaction Output (UTXO)-based blockchain like Bitcoin where the signature algorithm is replaced with a lattice-based signature scheme given in Algorithm 5 . In the UTXO model, coins are kept in addresses where each address consists of the amount and the spending condition. The spending condition is defined by the scripting language and the most common ones are signature and hash preimage verifications, and timing conditions. For our applications, we also assume that the underlying blockchain supports these scripts.

5.4.1. Atomic Swaps

An atomic swap can be defined between two users u_1 and u_2 who want to exchange two different cryptocurrencies c_1 and c_2. The crucial point of the exchange is ensuring fairness, i.e., either both parties receive their expected output or none do. In [25], an atomic swap protocol is presented with the following steps.

Setup. First, u_1 shares a hash value $h_1 := \mathsf{H}(r_1)$ of a secret r_1 to u_2. Then, u_1 creates a transaction on the coins c_1 such that it can be spendable by u_2 only if the preimage of h_1 is presented. Similarly, u_2 also creates a transaction on the coins c_2 with the same preimage condition for u_1. Here, both transactions have timeouts t_i such that, once t_i elapses, u_i can redeem c_i if the counterparty does not continue to the exchange. Also, the timelock, t_2, on u_2's transaction is shorter (i.e., $t_2 < t_1$) to ensure that u_2 would have enough time to react. First, u_1 publishes her transaction on-chain, then u_2 does the same.

Swap. Once both transactions are on-chain, u_1 can obtain c_2 by revealing r_1, which yields to u_2 obtaining c_1. Note that this protocol requires both scripting languages of the cryptocurrencies to have preimage conditioned scripts. Later on, in [5], the scriptless version of the protocol is presented where the hash condition is embedded into the signature algorithm.

Let us explain how to achieve atomic swaps using LAS, which requires careful analysis because of the aforementioned knowledge gap. In the scenario below, an *extracted* witness, which satisfies an *extended* relation (i.e., $\mathfrak{R}'_{\vec{A}}$, but not necessarily $\mathfrak{R}_{\vec{A}}$), will constitute the opening condition to receive coins.

Let (pk_i, sk_i) be the public-secret key pair for user u_i for $i = 1, 2$. First, u_1 generates a statement-witness pair $(Y, y) = (\vec{t}, \vec{r}) \in \mathfrak{R}_{\vec{A}}$ as in Section 5.3, and sends Y to u_2 along with a proof π of knowledge of a witness $\vec{r}$ such that $\vec{t} = \vec{A}r$ and $\|\vec{r}\|_\infty \leq 1$. Such a proof can be realized using the recent Esgin-Nguyen-Seiler proof system [26]. Then, u_1 also creates a pre-signature $\hat{\sigma}_1 \leftarrow \mathsf{PreSign}((pk_1, sk_1), Y, \mathsf{tx}_1)$ for tx_1 spending the coins c_1 to u_2. After verifying the proof π, u_2 similarly creates a pre-signature $\hat{\sigma}_2 \leftarrow \mathsf{PreSign}((pk_2, sk_2), Y, \mathsf{tx}_2)$ for tx_2 spending the coins c_2 to u_1. Then, the two pre-signatures are exchanged between the parties. Now u_1 adapts the pre-signature $\hat{\sigma}_2$ as $\sigma_2 \leftarrow \mathsf{Adapt}((Y, y), pk_2, \hat{\sigma}_2, \mathsf{tx}_2)$, and aborts if $\sigma_2 = \perp$. Otherwise, he publishes the full signature σ_2 on the second cryptocurrency's blockchain in order to receive the coins c_2. Then, seeing σ_2, u_2 runs $y' = \vec{s} \leftarrow \mathsf{Ext}(Y, \sigma_2, \hat{\sigma}_2)$ and $\sigma_1 \leftarrow \mathsf{Adapt}((Y, y'), pk_1, \hat{\sigma}_1, \mathsf{tx}_1)$. If any of them returns $\perp$, u_2 aborts. Otherwise, u_2 publishes σ_1 on the first cryptocurrency's blockchain to receive the coins c_1. This interaction is depicted in Figure 5.1.

Let us now analyze whether u_1 receives c_2 if and only if u_2 receives c_1. If u_1 does not receive c_2, i.e., u_1 aborts, then u_2 clearly cannot receive c_1 due to the aEUF-CMA security of LAS as u_2 only has the pre-signature $\hat{\sigma}_1$ and the statement Y (without a witness to Y). On the other hand, if u_1 does receive c_2, this means that σ_2 is valid signature published on a blockchain, i.e., accessible by u_2. Therefore, by the witness extractability of LAS, u_2 can extract a witness $\vec{s}$ to $Y = \vec{t}$ such that $\vec{t} = \vec{A}s$. Recall that u_1 proved knowledge of a witness $\vec{r}$ to $Y = \vec{t}$ such that $\|\vec{r}\|_\infty \leq 1$. By the hardness of M-SIS, it must be the case that $\vec{s} = \vec{r}$ as otherwise $\vec{A}(\vec{s} - \vec{r}) = 0$ gives a solution to M-SIS$_{n,n+l,q,\beta}$ for $\beta = 2\gamma\sqrt{d(n+\ell)}$. As a result, we have that $\|\vec{s}\|_\infty = \|\vec{r}\|_\infty \leq 1$. Therefore, $\vec{s} \in \mathfrak{R}_{\vec{A}}$ and the pre-signature adaptability works, and hence the signature σ_1 adapted by u_2 passes the verification. Note that without the proof of knowledge π, we cannot guarantee that the extracted witness $\vec{s}$ will satisfy $\|\vec{s}\|_\infty \leq 1$, and hence pre-signature adaptability would not have been guaranteed without π. In other words, π is essential to make sure that u_2 receives the coins c_1.

We also note that even though a lattice-based proof of knowledge, π, is relatively costly in terms of communication in practice (but very efficient in computation), this proof is only exchanged between the parties, and not published on blockchain. Therefore, it does not incur additional on-chain storage costs.

5.4.2. PAYMENT CHANNEL NETWORKS

Payment channel networks (PCNs) [4, 27–29] are one of the promising solutions to the scalability issues of blockchains. More specifically, many blockchains have poor transaction throughput compared to alternatives like credit card networks because of their consensus mechanisms, where every party (miner) approves and stores every transaction.

$u_1((pk_1, sk_1), pk_2, c_1)$		$u_2((pk_2, sk_2), pk_1, c_2)$
$(Y, y) = (\vec{t}, \vec{r}) \leftarrow \mathsf{Gen}()$		
$\pi \leftarrow \mathscr{P}\left((\vec{t}; \vec{r}), \{\exists \vec{r} : \vec{A}r = \vec{t} \wedge \|\vec{r}\|_\infty \le 1\}\right)$		
Generate tx_1 for spending c_1 to u_2		
$\hat{\sigma}_1 \leftarrow \mathsf{PreSign}((pk_1, sk_1), Y, \mathsf{tx}_1)$		
	$\xrightarrow{Y, \pi, \hat{\sigma}_1, \mathsf{tx}_1}$	
		If verif. of π or $\hat{\sigma}_1$ fails, Abort
		Generate tx_2 for spending c_2 to u_1
		$\hat{\sigma}_2 \leftarrow \mathsf{PreSign}((pk_2, sk_2), Y, \mathsf{tx}_2)$
	$\xleftarrow{\hat{\sigma}_2, \mathsf{tx}_2}$	
$\sigma_2 \leftarrow \mathsf{Adapt}((Y, y), pk_2, \hat{\sigma}_2, \mathsf{tx}_2)$		
If $\sigma_2 = \perp$, Abort		
Publish σ_2 on blockchain		
	$\xrightarrow{\sigma_2}$	
		$y' \leftarrow \mathsf{Ext}(Y, \sigma_2, \hat{\sigma}_2)$
		$\sigma_1 \leftarrow \mathsf{Adapt}((Y, y'), pk_1, \hat{\sigma}_1, \mathsf{tx}_1)$
		Publish σ_1 on blockchain if $\sigma_1 \neq \perp$

Figure 5.1: Atomic swap protocol using LAS.

PCNs improve the throughput by moving some transactions off-chain while relying on the security of the blockchain. In a PCN, two parties can lock coins into a channel where they can make instant and arbitrarily many transactions between each other so long as they have enough balances. One of the most popular PCNs built on Bitcoin is the Lightning network [29]. The overall structure of our post-quantum PCN resembles the Lightning network.

A payment channel consists of three steps: create, update, and close. In the creation phase, parties deposit some coins into the channel and create a funding transaction that spends the input addresses into a single output of the channel. The funding transaction is published on the blockchain and afterward, all of the updates are done off-chain until the closing part. The output condition of funding is spendable only if both parties sign it, which ensures an agreement by both parties. The condition can be implemented by a two-party multi-signature.

In realizing a two-party multi-signature, a straightforward option is to simply combine two individual signatures. Alternatively, there is a lattice-based multi-signature in [30], which can be used in the two-party setting. The underlying signature uses the same "Fiat-Shamir with Aborts" technique, and as stated in [30], the multi-signature can be realized over module lattices as in our work.

When parties want to send/receive coins in the channel, they make off-chain transactions and update the channel balances. In each update, parties create new commit transactions that spend the output of the funding transaction into the two new addresses of the parties with their corresponding balances. Also, parties revoke the previous commits by sharing the signing keys with each other. The revocation can be seen as a punishment mechanism to prevent a malicious party from publishing an old commitment. Once parties are done with the channel, they can close it and obtain their coins by publishing the latest commitment on-chain. A payment channel creation, update, and closing can be done in the same manner as the Lightning network. Now, we investigate how to achieve *multi-hop* payments with our adaptor signature scheme.

A *network* of channels allows parties to make multi-hop payments. More specifically, parties, who do not have a direct channel, can route a payment using the channels of some intermediary nodes. In these multi-hop payments, it is crucial to synchronize each channel on the route so that either all of them update accordingly or no one does. The Lightning network achieves this by using HTLC (hash-time lock contract). However, in [4], the authors presented privacy concerns as well as the *wormhole attack* for the HTLC mechanism. In this manner, we adopt the AMHL (anonymous multi-hop lock) technique [4] for the multi-hop payments. Also, it is stated that AMHLs are sufficient to construct a payment channel network [4, Theorem 4]. In a scenario where sender S (or I_0) wants to send payment through the intermediary nodes $I_1, \dots, I_{k-1}$ to the receiver R (or I_k), AMHL-based multi-hop payment works as follows (for simplicity, we omit the fees given to the intermediary nodes).

Setup. S chooses random strings $\ell_0, \ell_1, \dots, \ell_{k-1}$, and computes $y_j := \sum_{i=0}^{j} \ell_i$ and $Y_j := \mathsf{G}(y_j)$ for $j = 0, \dots, k-1$ where G is an additively homomorphic one-way function. Then, S shares (Y_{j-1}, Y_j, ℓ_j) with each intermediary I_j for $i = 1, \dots, k-1$ and (Y_{k-1}, y_{k-1}) with R. Each intermediary party I_j validates the correctness of values by using the homomorphism, i.e., checking that $\mathsf{G}(\ell_j) \oplus Y_{j-1} = \mathsf{G}(y_j) = Y_j$, where $\oplus$ denotes the operation in the range of G.

Payment. S makes a conditional payment to I_1 requiring preimage of Y_0, while each intermediary party I_j, for $j = 1, \dots, k-1$, makes a payment of the same amount to I_{j+1} with a condition on preimage of Y_j after they receive a similar payment from I_{j-1}. Once all conditional payments are placed, S reveals the preimage y_{k-1} to $R = I_k$ showing that she can redeem the payment. This creates a chain reaction as follows. When an intermediary party I_j receives y_j from I_{j+1}, he can compute $y_{j-1} = y_j - \ell_j$ and redeem the payment by revealing y_{j-1} to I_{j-1}. The procedure is completed once all the channels are updated accordingly.

We can realize AMHL in the post-quantum setting using LAS, but again a special care is required due to the knowledge gap and the use of rejection sampling. First of all, we assume that the length of the PCN is at most $K \ll q$ (i.e., $k \le K \ll q$) and update the norm check at Steps 6 and 11 in Algorithm 6 by $\|\vec{z}\|_\infty > \gamma - \kappa - K$. Now, S samples $\vec{r}_j \xleftarrow{\$} \mathbb{S}_1^{n+\ell}$, and computes $\vec{s}_j = \sum_{i=0}^{j} \vec{r}_i$ and $\vec{t}_j = \vec{A}s_j$ for $j = 0, \dots, k-1$. Observe that we have $\|\vec{s}_j\|_\infty \le k \le K$ for all $j = 0, \dots, k-1$. Then, S treats $Y_j = \vec{t}_j$, $y_j = \vec{s}_j$ and $\ell_j = \vec{r}_j$ for $j = 0, \dots, k-1$. The additively homomorphic function is $f_{\vec{A}}(\vec{x}) = \vec{A}x$ (over $\mathscr{R}_q$) mentioned in Section 5.3. Then, the Setup phase of AMHL described above is run. Additionally, for each $j = 0, \dots, k-2$, S sends I_{j+1} a NIZK proof π_{j+1} that she knows a witness $y_j = \vec{s}_j$ to

$Y_j = \vec{t}_j$ such that

$$\|\vec{s}_j\|_\infty \leq K. \tag{5.8}$$

After this setup, payment phase begins. Let (pk_j, sk_j) be I_j's public-secret key pair used in his channel with I_{j+1}, and tx_j be the transaction transferring the relevant coins from I_j to I_{j+1}. S creates a pre-signature $\hat{\sigma}_0 \leftarrow \mathsf{PreSign}((pk_0, sk_0), Y_0, \mathsf{tx}_0)$ and sends it to I_1. Then, for $j = 1, \ldots, k-1$, each user I_j creates a pre-signature $\hat{\sigma}_j \leftarrow \mathsf{PreSign}((pk_j, sk_j), Y_j, \mathsf{tx}_j)$ after receiving the pre-signature $\hat{\sigma}_{j-1}$ from I_{j-1}. Once all pre-signatures are generated and transferred, S reveals y_{k-1} to R, which allows R to adapt the pre-signature $\hat{\sigma}_{k-1}$ to σ_{k-1} in order to receive the relevant coins from I_{k-1}. R sends σ_{k-1} to I_{k-1}. From here, I_{k-1} extracts a witness y'_{k-1} to Y_{k-1}. Then, she computes $y''_{k-2} = y'_{k-1} - \ell_{k-1}$ and uses it to complete the pre-signature $\hat{\sigma}_{k-2}$. Continuing this way, completion of a pre-signature by I_j enables I_{j-1} to obtain a witness to Y_{j-1} and then compute a witness to Y_{j-2} using ℓ_j. The process ends with S receiving σ_0. This anonymous multi-hop payment procedure is depicted in Figure 5.2.

Let us analyze the details now. First of all, each party I_j has a proof that S knows a witness $y_{j-1} = \vec{s}_{j-1}$ to Y_{j-1} satisfying (5.8). Due to the M-SIS hardness as before, no party I_j can obtain another witness to Y_{j-1}, but y_{j-1} generated by S. Therefore, each party I_j is ensured that the witness he extracts will have infinity norm at most K. As a result, each party I_j will be able to adapt the pre-signature $\hat{\sigma}_{j-1}$ successfully and claim his coins thanks to the aforementioned change to Steps 6 and 11 in Algorithm 6.

We emphasize again the importance of the proof π_j's that guarantee pre-signature adaptability. These proofs are only communicated off-chain and thus do not incur any additional on-chain cost, and can be realized using the techniques in [26]. Moreover, the change to Steps 6 and 11 in Algorithm 6 is also important as, in this setting, even honestly-generated witnesses have potentially absolute coefficients greater than 1, but still at most K. Note that this change does not affect the security assumptions as still the original conditions (and even stronger ones) in Algorithms 5 and 6 hold. The only effect is that PreSign may have more restarts, but for most practical settings of, say, $K \leq 50$ (i.e., the length of the PCN is at most 50), the effect will be minimal. In practice, for example, in Lightning Network, the route search algorithm typically stops after $K = 20$ [31].

5.5. Conclusion

In this work, we constructed the first post-quantum adaptor signature based on standard lattice assumptions. We also showed that our construction, LAS, leads to efficient atomic swaps and payment channel networks in the post-quantum world. In particular, our applications do not incur additional costs on the blockchain, other than the cost of an ordinary lattice-based signature.

```
$S(pk_0, sk_0)$                          [SETUP]        $I_j(pk_j, sk_j)$    $j = 1, \dots, k$
for $j = 0, \dots, k-1$
    $\vec{s}_j = \sum_{i=0}^{j} \vec{r}_i$ for $\vec{r}_j \xleftarrow{\$} \mathbb{S}_1^{n+\ell}$
    $\vec{t}_j = \vec{A}s_j$
    $Y_j := \vec{t}_j,\ y_j := \vec{s}_j,\ \ell_j := \vec{r}_j$
for $j = 0, \dots, k-2$
    $\mathsf{st}_j = \left\{\exists \vec{s} : \vec{A}s = \vec{t}_j \wedge \|\vec{s}\|_\infty \le K\right\}$
    $\pi_{j+1} \leftarrow \mathscr{P}\left((\vec{t}_j; \vec{s}_j), \mathsf{st}_j\right)$
    $T_{j+1} := (Y_j, \ell_{j+1}, \pi_{j+1})$
$T_k := (Y_{k-1}, y_{k-1})$               $\xrightarrow{T_j}$
                                                  if $j \neq k$ then
                                                      $T_j =: (Y_{j-1}, \ell_j, \pi_j) =: (\vec{t}_{j-1}, \vec{r}_j, \pi_j)$
                                                      if $\mathcal{V}\left(\vec{t}_{j-1}, \mathsf{st}_{j-1}, \pi_j\right) = 0$, Abort
                                                      $Y'_j := \vec{t}'_j = \vec{A}r_j + \vec{t}_{j-1}$
                                                      $Z_j := (Y'_j, Y_{j-1}, \ell_j)$
                                         [PAYMENT]
$I_j((pk_j, sk_j), pk_{j-1}, Z_j)$                         $I_{j+1}((pk_{j+1}, sk_{j+1}), pk_j, Z_{j+1})$
Parse $Z_j = (Y'_j, Y_{j-1}, \ell_j)$
Obtain $\hat{\sigma}_{j-1}$ from $I_{j-1}$
Generate $\mathsf{tx}_j$, spending coins to $I_{j+1}$
$\hat{\sigma}_j \leftarrow \mathsf{PreSign}((pk_j, sk_j), Y'_j, \mathsf{tx}_j)$     $\xrightarrow{\hat{\sigma}_j}$
                                                      Parse $Z_{j+1} = (Y'_{j+1}, Y_j, \ell_{j+1})$
                                                      Obtain $\sigma_{j+1}$ from $I_{j+2}$
                                                      $y'_{j+1} \leftarrow \mathsf{Ext}(Y'_{j+1}, \sigma_{j+1}, \hat{\sigma}_{j+1})$
                                                      // Note $\hat{\sigma}_{j+1}$ is created by $I_{j+1}$
                                                      $y''_j = y'_{j+1} - \ell_{j+1}$
                                                      $\sigma_j \leftarrow \mathsf{Adapt}((Y_j, y''_j), pk_j, \hat{\sigma}_j, \mathsf{tx}_j)$
                                          $\xleftarrow{\sigma_j}$
$y'_j \leftarrow \mathsf{Ext}(Y'_j, \sigma_j, \hat{\sigma}_j)$
$y''_{j-1} = y'_j - \ell_j$
$\sigma_{j-1} \leftarrow \mathsf{Adapt}((Y_{j-1}, y''_{j-1}), pk_{j-1}, \hat{\sigma}_{j-1}, \mathsf{tx}_{j-1})$
```

Figure 5.2: Anonymous multi-hop payments using LAS. We assume that (i) T_j's are transmitted confidentially, (ii) pre-signature transmission from I_j to I_{j+1} happens only if that from I_{j-1} to I_j already happened, and (iii) signature transmission from I_{j+1} to I_j happens only if that from I_{j+2} to I_{j+1} already happened.

REFERENCES

[1] A. Poelstra, *Scriptless scripts,* Presentation Slides (), https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-05-milan-meetup/slides.pdf.

[2] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi, *Generalized bitcoin-compatible channels,* Cryptology ePrint Archive, Report 2020/476 (2020), https://eprint.iacr.org/2020/476.

[3] L. Fournier, *One-time verifiably encrypted signatures a.k.a. adaptor signatures,* (2019), https://github.com/LLFourn/one-time-VES/blob/master/main.pdf.

[4] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, *Anonymous multi-hop locks for blockchain scalability and interoperability,* in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019* (2019).

[5] A. Poelstra, *Adaptor signatures and atomic swaps from scriptless scripts,* (), https://github.com/ElementsProject/scriptless-scripts/blob/master/md/atomic-swap.md.

[6] T. Dryja, *Discreet log contracts,* https://adiabat.github.io/dlc.pdf.

[7] NIST, *Post-quantum cryptography – call for proposals,* (2017), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization/Call-for-Proposals, accessed on April 20, 2020.

[8] V. Buterin, *Understanding serenity, part i: Abstraction,* (2015), https://blog.ethereum.org/2015/12/24/understanding-serenity-part-i-abstraction/, accessed on April 20, 2020.

[9] Zcash, *Frequently asked questions,* https://z.cash/support/faq/#quantum-computers, accessed on April 20, 2020.

[10] Hcash, *Hcash features,* https://h.cash/#section4, accessed on April 20, 2020.

[11] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, *Crystals–Dilithium: Digital signatures from module lattices,* in *CHES*, Vol. 2018-1 (2018) https://eprint.iacr.org/2017/633.pdf.

[12] M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu, *Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications,* in *CRYPTO (1)*, LNCS, Vol. 11692 (Springer, 2019) pp. 115–146, (Full version at https://eprint.iacr.org/2019/445).

[13] M. F. Esgin, R. Steinfeld, A. Sakzad, J. K. Liu, and D. Liu, *Short lattice-based one-out-of-many proofs and applications to ring signatures,* in *ACNS*, LNCS, Vol. 11464 (Springer, 2019) pp. 67–88, (Full version at https://eprint.iacr.org/2018/773).

[14] M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu, *MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol,* in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19 (ACM, 2019) pp. 567–584, (Full version at https://eprint.iacr.org/2019/1287).

[15] W. A. A. Torres, V. Kuchta, R. Steinfeld, A. Sakzad, J. K. Liu, and J. Cheng, *Lattice RingCT v2.0 with multiple input and multiple output wallets,* in *ACISP*, LNCS, Vol. 11547 (Springer, 2019) pp. 156–175.

[16] V. Lyubashevsky, *Fiat-shamir with aborts: Applications to lattice and factoring-based signatures,* in *ASIACRYPT* (Springer, 2009) pp. 598–616.

[17] V. Lyubashevsky, *Lattice signatures without trapdoors,* in *EUROCRYPT* (Springer, 2012) pp. 738–755, (Full version).

[18] A. Langlois and D. Stehlé, *Worst-case to average-case reductions for module lattices,* Designs, Codes and Cryptography **75**, 565 (2015).

[19] M. Ajtai, *Generating hard instances of lattice problems (extended abstract),* in *STOC* (ACM, 1996) pp. 99–108.

[20] O. Regev, *On lattices, learning with errors, random linear codes, and cryptography,* J. ACM **56**, 34:1 (2009), preliminary version in STOC 2005.

[21] I. Damgård, *On Σ-protocols,* Lecture Notes, University of Aarhus, Department for Computer Science (2002), https://www.cs.au.dk/~ivan/Sigma.pdf.

[22] C. Schnorr, *Efficient identification and signatures for smart cards,* in *CRYPTO*, Lecture Notes in Computer Science, Vol. 435 (Springer, 1989) pp. 239–252.

[23] D. Pointcheval and J. Stern, *Security arguments for digital signatures and blind signatures,* J. Cryptology **13**, 361 (2000).

[24] M. F. Esgin, *Practice-Oriented Techniques in Lattice-Based Cryptography*, Ph.D. thesis, Monash University (2020), https://bridges.monash.edu/articles/Practice-Oriented_Techniques_in_Lattice-Based_Cryptography/12279728.

[25] T. Nolan, *Alt chains and atomic transfers,* https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949.

[26] M. F. Esgin, N. K. Nguyen, and G. Seiler, *Practical exact proofs from lattices: New techniques to exploit fully-splitting rings,* in *ASIACRYPT (2)*, Lecture Notes in Computer Science, Vol. 12492 (Springer, 2020) pp. 259–288.

[27] C. Decker and R. Wattenhofer, *A fast and scalable payment network with bitcoin duplex micropayment channels,* in *Symposium on Self-Stabilizing Systems* (Springer, 2015).

[28] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, *Sok: Off the chain transactions.* IACR Cryptology ePrint Archive **2019**, 360 (2019).

[29] J. Poon and T. Dryja, *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments,* (2016), draft version 0.5.9.2, available at https://lightning.network/lightning-network-paper.pdf.

[30] R. E. Bansarkhani and J. Sturm, *An efficient lattice-based multisignature scheme with applications to bitcoins,* in *CANS*, LNCS, Vol. 10052 (2016) pp. 140–155.

[31] *Basis of lightning technology,* Available at: https://github.com/lightningnetwork/lightning-rfc/blob/master/00-introduction.md.

6 DISCUSSION

Since "Bitcoin: A peer-to-peer electronic cash system" paper has been publicly available, it has attracted the attention of both research and industry communities. The recognition of Bitcoin has introduced the notion of cryptocurrencies and, in general, blockchain technology. In the last decade, blockchain has become one of the exciting developments where academia and industry jointly initiate real-life applications and implementations of cutting-edge technology.

Notwithstanding the interest and great effort, blockchain is still a new and evolving technology, and there are numerous challenges that need to be addressed. To name a few, security, privacy, scalability, smart contracts, and economic aspects with their manifold sub-challenges can be mentioned. Among the research challenges, we investigate three crucial ones for the long-term functionality of the Bitcoin-like blockchains, which are security, scalability, and economic aspects.

Bitcoin has been in the center of blockchain research as it is the first commonly accepted cryptocurrency and still owns more than 60 percent of the total market capitalization of cryptocurrencies. Also, many of the cryptocurrencies and blockchain protocols have utilized Bitcoin's components, such as consensus protocol or transaction structure. Therefore, our research and improvements over Bitcoin can be applied to several blockchain protocols.

In this chapter, we present our findings together with their limitations and potential future works. We explained the research directions and open questions in Section 1.3, and listed the ones investigated in the thesis in Section 1.4. Here, we present our research in two domains: transaction advertisement and payment channel networks.

6.1. TRANSACTION PROPAGATION

In this section, we explain the first two research questions that are related to the transaction propagation and our solutions given in Chapter 2. Firstly, we investigate the lack of incentives for the transaction propagation and provide an incentive mechanism for peer-to-peer mining networks. Secondly, we focus on the inefficient routing of the transactions and propose a solution that is suitable for an improved new version of Bitcoin.

6.1.1. INCENTIVES

The first problem we investigate in Chapter 2 is

Q1: *How to design an incentive-compatible and Sybil-proof transaction propagation protocol that is suitable for well-connected peer-to-peer mining networks?*

Transaction propagation or advertisement is crucial for the proper functionality of the blockchain since only the miners who have the transaction can add it to the blockchain. The existing blockchains, including Bitcoin, do not have an incentive mechanism for transaction propagation. The fact that they work without the propagation incentive does not conflict with the necessity of the incentives in the long-term functionality of cryptocurrencies. There are two main reasons for the current, short-term, functionality: altruism and centralized mining. Early miners of Bitcoin were altruistic and propagated transactions in the mining network regardless of their individual gain [1, 2]. Later on, the mining network became rational but centralized mining pools [3–6]. In the current centralized setting, clients reach a miner of the pools, and then the pool handles the internal propagation to its miners. Consequently, the interim systems do not exhibit decentralized and peer-to-peer mining networks.

A way of rewarding the propagation can utilize the existing incentive instrument, transaction fee. The idea is that the parties in the propagation path from the client to the miner who adds it to the block, miner of the block, can receive a partial reward of the transaction fee. There are two challenges in propagation incentive mechanisms: ensuring that honest propagating parties are rewarded and preventing Sybil/fake propagating parties. Firstly, the parties who propagate the transaction to the miner of the block should be rewarded, and no one, including the miner, should be able to alter the list of the propagation path. Secondly, the propagating parties should not gain more fees by creating fake identities and pretending that they propagated through them. Not that these fake parties would not contribute to the propagation to the miners but increase the workload in the process.

There have been proposals that aim to solve the aforementioned challenges [7, 8]. The first challenge is solved with a chain of signatures where each propagating party suffixes the public key of the next party and signs it before sending the transaction. Thanks to the chaining mechanism, the path from the client to the miner who adds to the block would be secured, and any removal or addition of an intermediary party would break the chain of signatures. The second challenge requires game-theoretical analysis of the propagation reward. The existing Sybil-proof rewarding mechanisms work for specific network structures such as tree-structured networks, which do not reflect a common peer-to-peer mining network.

In our work, we analyze all network structures for our incentive model that shares the fee among the propagating parties and the miner of the block. In our model, we prove that for poorly-connected networks that can be disconnected by the removal of a party, namely 1-connected networks, there is no Sybil-proof rewarding mechanism. Note that this type of networks would not be encountered in a real mining network since the mining networks are well-connected. Also, we present a Sybil-proof mechanism that is suitable for any well-connected networks. Our analysis is based on a simultaneous move game where we compute the fee gain of the miners if they propagate the transaction or not. We show that if the probability of a miner creating the next block (solving the puzzle in Bitcoin) is less than a *factor* of its neighbors, then it would propagate. This factor is related to the network connectivity, and a detailed analysis of it is left as future work.

Limitations and Future Work: Long-term Analysis in a Real Network

As mentioned above, the network connectivity factor is one of the interesting open questions of our work. The factor determines how much of the transaction fee share is given to each propagating party. If the factor is high, then the intermediary parties would have high rewards and incentives to propagate the transaction, yet the miner of the block would have less reward and incentive to add it to the block, and vice versa. That is why it is crucial to choose a moderate value that suits both intermediary parties and the miner of the block. Determination of this factor requires rigorous analysis of the mining network regarding the number of connections of each party and their mining capacities. Because of the security and privacy concerns, the corresponding network information of Bitcoin is not available.

Another network-related open question is the effects of the incentive mechanism in the network. Our model for the incentive mechanism assumes a stable mining network meaning that the connections of the miners do not change dramatically. However, if there will be a propagation incentive, then some of the miners may try to increase their connections to participate more in the propagations. The post effects of the propagation incentives in the mining network is left as future work.

Now, we discuss practical concerns of our incentive mechanism. In the existing blockchains, the transactions are stored with their witnesses, which are usually the signatures of the owners. However, to implement the propagation incentive mechanism, we need to store the signatures of the propagating parties as well. These additional signatures would increase the transaction size and thereby reduce the number of transactions stored in a single block of fixed size. Also, the propagation rewards would be relatively small since they are a portion of a transaction fee, and the current blockchains may not be suitable for spending these micropayments. In this regard, an interesting open problem is designing an off-chain rewarding mechanism for the propagation that would prevent the additional storage cost on the blockchain, and it would be more suitable for micropayments.

Our Follow-Up

In our follow-up work [9], we propose the first Sybil-proof rewarding mechanism that works for any network, including poorly connected ones, by modifying the rewarding model. The challenge in the poorly connected networks is that there can be only one path that connects some client-miner pairs. The intermediary parties in that path have the monopoly effect, whereas, in other cases, alternative paths have to compete. Our previous model shares the transaction fee among the propagating parties and the miner who adds to the block. This is inherited from Bitcoin because the reward mechanism of Bitcoin gives all of the block reward and transaction fees to the miner of the block. In the new model, we add the miner of the next block into our reward mechanism, which has been the shareholder in Bitcoin-NG protocol [10] to prevent selfish mining. Our new model allows us to design a path length-dependent rewarding mechanism for the intermediary parties, discouraging them from introducing Sybil parties even in a single path scenario. After formulating all the conditions, we obtain the following rewarding function

$$f_{[i]}^{\ell}: \begin{cases} F \cdot \sum_{j=i}^{\ell} \left((-1)^{j-i} \cdot \binom{\ell-i}{j-i} \cdot \prod_{k=1}^{j-1} (1-\Gamma_k) \right) & \text{for } 1 \leq i < \ell, \\ F \cdot \prod_{k=1}^{\ell-1} (1-\Gamma_k) & \text{for } i = \ell. \end{cases}$$

where F is the transaction fee, $f_{[i]}^{\ell}$ is the reward of the i^{th} node in the propagation path, $f_{[NRL]}^{\ell} = F - \sum_{i=1}^{\ell} f_{[i]}^{\ell}$ is the reward of the miner of the next block, and Γ_k's are the network connectivity variables. It can be seen that $f_{[i]}^{\ell} = f_{[i]}^{\ell+1} + f_{[i+1]}^{\ell+1}$ satisfied for all ℓ and i values, which shows that the Sybil parties do not increase the reward of an intermediary party.

6.1.2. EFFICIENT ROUTING

The second problem we analyze in Chapter 2 is the following:

Q2: *How to get rid of the redundancy of the transaction propagation if the miner of the block is known in advance?*

In a mining network, the transactions are broadcast to the miners, and later on, a miner would create a block including them and broadcast the block. Here, we have the redundant propagation of the transactions. More specifically, the same transaction is broadcast twice: for the advertisement of the transaction itself and for the advertisement of the block including it. In some existing blockchains, the redundancy is reduced by a mechanism that first checks if the receiving miner already has the transaction or not by sharing the hash of it. However, these check messages may cost more bandwidth usage than the size of the transaction itself since it would come from multiple neighbors [11].

In our work, we reduce the redundancy of the propagation by reducing the cost of the transaction advertisement. We note that the block advertisement is necessary since every miner should have the same view of the blockchain. However, the transaction advertisement is only necessary for the miner who will create the next block. Our idea is to create a path from each miner to the one who is responsible for the next block, namely round leader. Then, each client would reach a set of miners, and these miners, instead of broadcasting the transaction, will route to the leader using the previously created paths. Since the leader is changing every round, the paths will also be dynamically updated. The mechanism is applicable if the leader is known before the block of transactions is created.

To create the paths to the round leader, we propose a smart routing algorithm with two phases: recognition and transaction. In the recognition phase, the round leader makes itself known to the network by sharing proof of its identity. Each party in the network would store the first sender of the proof as the gradient party and propagate the proof to its neighbors. Hereby, each party learns their close connection to the leader. This connection, gradient party, would be the closest party to the leader under the assumption that the delay is approximately the same in the network. In the transaction phase, the clients send their transaction to a couple of their neighboring miners, and then each miner only forwards it to their gradient party. The reason for having multiple paths is to avoid a single point of failure. The experiments on a simulated network of 1-10K miners show that our protocol reduces the redundant transaction propagation by up to 99%.

Limitations and Future Work: Compatibility with Existing Blockchains

There are three concerns regarding the recognition phase of the routing protocol: availability, latency, and privacy. The availability issue is that the round leader needs to be known before the transactions are propagated. We define this type of blockchain consensus protocols as first-leader-then-block (FLTB), where the round leader can be validated before he creates the block of transactions. The standard proof-of-work of Bitcoin does not satisfy this requirement because the proof requires the block with transactions. A recently proposed advanced version of it, given in Bitcoin-NG [10], separates the transactions from the proof, which allows us to realize the smart routing algorithm. In our follow-up work, we present a framework applying this separation in any blockchain protocol.

The latency concerns of the routing protocol are related to the network delay. First, in our model, we assume that the network delay is approximately the same in any two parties, which makes the gradient party is the closest neighbor to the leader. Yet, in a real network, the delay would variate. Thus, our protocol may not create the optimum routes in terms of the network distance. Secondly, our transaction propagation phase will add time delay to the block creation. In the existing blockchains, since the transactions are propagated as they are created, the round leader would already have most of the new transactions. However, in our routing protocol, they are collected after the recognition phase. Analysis of the effects of the delay in practice is left as future work.

The privacy concern of the recognition phase is the proximity leakage of the round leader. Each party would know their gradient node that is closer to the round leader. Note that the proximity information does not provide the exact position, even to the neighbors of the leader. However, there are de-anonymization techniques where IP addresses in the peer-to-peer network can be matched with the public keys on the blockchain [12, 13]. A powerful adversary may locate the round leader using these techniques and try to deploy a denial-of-service (DoS) attack. We should emphasize that our smart routing protocol does not cause this vulnerability in the Bitcoin-NG protocol. The routing protocol does not leak any more information of the round leader than the existing protocol; it just uses the information for efficient routing. A solution for the proximity problem would be using the anonymization techniques for locational privacy, e.g., Dandelion protocol [14] where the leader's proof of identity is first sent in a line of parties and then broadcast from there. This would cause the addition of a couple of parties into the routing paths.

Our Follow-Up

In our follow-up work [15], we present a blockchain framework that separates block creation and transaction collection. In more detail, we introduce a new actor called *transaction collector* (TC) that is responsible for the collection and broadcast of the transactions. In each round, several TCs are selected with the follow-the-satoshi algorithm [16], which assigns the hash of the round quantity to a coin owner. The round quantity is updated by the round leader using the verifiable random function given in [17]. For the block creation, the round leader creates the block as before, but instead of including transactions, it will refer to the transactions sets published by TCs. Thus, since the round leader selection process is irrelevant for the routing protocol, the routing can be applied to any consensus protocol with the new framework. Another advantage of the framework is that the locational privacy issue is moved from the round leader to the TCs, and it would be harder to deploy the DoS attack since there would be multiple TCs in a round.

6.2. PAYMENT CHANNEL NETWORKS

In this section, we explain our solutions for the research questions related to payment channel networks (PCN) and their limitations. We analyzed the incentives of the multi-hop payments in Chapter 3 and proposed virtual channels that remove the need for interaction with an intermediary for each payment in Chapter 4. Finally, we presented the first post-quantum PCN using adaptor signatures in Chapter 5.

6.2.1. INCENTIVES

In Chapter 3, we address the following question:

Q3: *How to determine the fee of a channel that maximizes the profit and encourages the owner of the channel to create new channels yielding to the growth of the network?*

In a PCN, multi-hop payments allow parties to create a payment route using intermediary parties' channels. The process requires the intermediary parties to lock their coins in their channel, and they would receive a fee for their participation. The determination of these fee values is an open question, and the fee values used in the existing implementations are not informative. First of all, most of the PCN protocols are in the prototype phase. Secondly, our analysis on the only actively used PCN, Lightning Network (LN), indicates that most of the parties use the default fee parameters to define their fees. This is because, currently, the parties in LN are mostly altruistic or enthusiastic. However, as the network becomes more popular and profitable, the parties would be more rational and optimize their parameter selection. Note that a similar transition from altruistic to rational behavior has already happened in the Bitcoin mining network.

In our work, we present a fee strategy for a party **A** who wants to maximize her overall profit by creating new channels and tuning the fee parameters of her channels. For our analysis, we used the fee model of LN and its network data. In addition, since there is no publicly available data on the multi-hop payments regarding the amounts and the sender and receiver pairs, we assume that each party in the network is equally likely to be the sender or receiver of a payment, and we analyze our strategy for a few fixed payment amounts. In LN, the sender of a multi-hop payment would choose the payment path with the cheapest fee in total. Thus, if the intermediary party charges a high fee for the multi-hop payment, then the sender would most likely choose another path that is cheaper. Also, if the intermediary charges a small fee, then the profit from each payment would be small. In other words, while the party would like to be on the cheapest path between the sender and the receiver, she would not want to earn an insignificant fee the each payment. We aim to find the best fee that maximizes the total profit of all transactions.

We formulate the profit function of **A** that computes the total gain from her channels by considering all possible fee parameters and sender-receiver pairs. The problem of maximizing profit is related to a graph metric: betweenness centrality, which counts the number of shortest paths passing through a node. If we convert the network into a graph where the channel fees are seen as the weights of the edges, then the profit of a party from payment is equal to the probability of being in the shortest path times the weight of the edge. Using this correlation, we show that the profit-maximizing problem is NP-hard by reducing it to the maximizing betweenness centrality problem, which is NP-hard [18].

We propose a greedy algorithm for the profit-maximizing problem where **A** connects to a new party in the network that maximizes the total profit. At each step, we try all parties in the network, and for all possible channels, we choose the best fee parameters. Note that trying all fee parameters for all channels would require enormous computation. To improve our exhaustive search, we present an efficient fee search algorithm that reduces the search space in the order of 10-100. The experiments on the Lightning Network data show that our greedy algorithm improves the profit significantly. If **A** is creating new channel connections, our strategy at least a factor of two better than the other strategies that connect to the parties with the best graph centrality metrics. Moreover, **A** can still improve her profit by using our fee search algorithm on her existing channels.

Limitations and Future Work: Long-term Analysis in a Dynamic Network

Here, we explain the limitations of the model used in our work and the open challenges. Firstly, in our analysis, we analyze the fee parameters of **A** who wants to maximize her profit while we do not consider the effect on other parties. As already mentioned, in the current setting, most of the parties use the default fee parameters. However, this would change as the network become more profitable. In that case, each party would try to optimize their fee parameters. In our model, we assume the other channels' fee parameters are stable and do not change during the updates of **A**. The analysis of the scenario where every party in the network tries to maximize their profit while considering the other parties' actions is left as future work.

The second limitation of our model concerns the coins of **A** while creating the new channels. We assume that **A** would have enough coins to create channels with balances more than the fixed payment amounts. Yet, for a more realistic model, the coins owned by **A** should be bounded. Analysis of the constrained version of our greedy algorithm is an open question. Moreover, another interesting open question of the constrained model would be the distribution of the limited coins among the channels.

The third limitation of our model is that the effect of a payment in the channel balance is not taken into account. In more detail, after each payment in a channel, the balances of both parties would be updated while preserving the total amount in the channel. The balance on one side of the channel would determine the upper limit of the payment amount that can be carried in that direction. The balance distributions of channels in LN are not publicly available because of privacy concerns. In our model, we assume that the transactions can be routed as long as the total capacity of a channel is more than the transaction amount, which is not the case in practice. Also, we do not consider the changes in the balance of a channel after a payment is processed. A long-term analysis of the channel profit that takes into account the balances is an open question.

The analysis of the channel balances has become an important concern for healthy network functionality as well. This is because most of the channels are presumably used in one direction, which eventually leads to the accumulation of the channel capacity on one side of the channel. These one-sided channels can participate in the multi-hop payments only in the direction from the accumulated side. Therefore, it is crucial to have well-balanced channels for the multi-hop payments in the network. One way to achieve a healthy functioning network is to incentivize balancing of the channels with the fee parameters. The design of the fee strategy that improves both the channel's balance distribution and the profit of the parties is an interesting open challenge. Unlike the previous ones, a solution to this challenge would assist the whole network rather than a single party and thereby would have a significant impact on the network functionality.

6.2.2. VIRTUAL CHANNELS

In Chapter 4, we address the following question:

Q4: *How to design a virtual channel between two parties separated with an intermediary that does not require the intermediary involvement in each payment?*

As mentioned earlier, a multi-hop payment in a PCN requires the involvement of the intermediary parties on the payment path. Imagine a scenario where Alice and Bob are connected via an intermediary party Ingrid, i.e., Ingrid has payment channels with both parties. Whenever Alice and Bob want to exchange coins, they create a multi-hop payment that requires both channels' updates. This has several drawbacks: (i) Ingrid needs to be online for the updates, (ii) Alice and Bob need to pay a fee for every payment individually, (iii) the payment procedure requires more message exchanges compared to a channel update between two parties. A simple solution would be creating a new payment channel between Alice and Bob, which would cost two on-chain transactions. Recently, an alternative solution has been introduced in [19] where a *virtual channel* is created over the existing channels with only off-chain message exchanges. The intermediary party, Ingrid, is only involved in the creation and closing of the virtual channel, and the end parties, Alice and Bob, can exchange arbitrarily many transactions between themselves, just like in a regular channel. Unfortunately, the virtual channel construction given in [19] is suitable for Turing-complete scripting language and account-based model. The construction of a virtual channel with a limited scripting language has been an open question.

Our work presents the first virtual channel constructions that can be realized in any blockchain, including Bitcoin, with the UTXO model and scripting language supporting digital signatures and timelocks. We formulate the ideal functionality of the virtual channels in the Universal Composability framework and propose two virtual channel constructions, namely, virtual channels with validity (VC-V) and without validity (VC-NV) that realize the ideal functionality. Our constructions are built over generalized (payment) channels [20], and we also explain how to build them on Lightning Network's payment channels. To create a virtual channel, we construct a special founding transaction that takes the outputs of the underlying payment channels as its inputs. Note that the founding transaction is not published on the blockchain unless a party behaves maliciously. For the updates of a virtual channel, we rely on the same revocation mechanisms used in a payment channel. Finally, the closure of the virtual channel is done by updating the underlying payment channels according to the corresponding balance distribution.

The main challenge in virtual channel construction is that it does not rely on on-chain transactions like regular payment channels. Instead, the end parties create a channel over the intermediary. In a sense, the intermediary takes the role of the blockchain, yet it is not trusted. In a payment channel, the balance security of a party is provided by the on-chain funding transaction and revocation mechanism. In other words, an honest party would be reimbursed via the revocation mechanism if the counter-party misbehaves.

Unlike payment channels, the balance security of an honest party in a virtual channel construction cannot rely on only the revocation over the blockchain because the end parties do not have control over the payment channel between the intermediary and the other party. Therefore, the balance security of the end parties requires additional measures. We preserve the balance security of the end parties with the collaterals of the intermediary, Ingrid. Thus, an honest end party would be reimbursed in her payment channel if the other parties misbehave. For Ingrid, it can be assumed that she has control over both underlying payment channels since she is part of them. However, we also need to make sure that she would not lose her collaterals if she is honest and follows the protocol. This is important if both end parties maliciously try to reimburse via their payment channels.

In this work, we provide two constructions VC-V and VC-NV, that preserve each honest party's balance security. The constructions differ in their founding transaction structure and the validity time constraint of the channel. In VC-V, the virtual channel is built over one of the payment channels, let say on Alice and Ingrid's channel, and the other payment channel can be seen as the collateral of the virtual channel. Here, Alice is responsible for the closure of the channel before the validity time. If she does not close it on time, then Ingrid would punish her by getting all the coins in their channel, and Bob would be reimbursed with his channel with Ingrid. In VC-NV, the virtual channel and the collateral are built over both payment channels with a single transaction, and there is no time constraint for the channel. Ingrid can close the virtual channel whenever she wants. Also, the end parties can jointly close the channel or enforce the channel's closure by the intermediary. In both constructions, when the virtual channel is closed, the underlying payment channels are updated accordingly and can be used for payments as before.

Limitations and Future Work: Multiple Intermediaries

Here, we explain potential improvements and future works on our virtual channel constructions. So far, we explained virtual channel constructions for two end parties connected with an intermediary party. There are two dimensions that can be improved in the construction. The first one is the distance between the end parties of a virtual channel, i.e., the number of intermediary parties in between. In our paper, we conjuncture the possibility of the recursive construction of the virtual channels on top of each other that yields to virtual channels between parties having a distance of more than one. Nevertheless, the detailed construction and security analysis under the Universal Composability framework is left as future work. The second dimension that can be enhanced is the number of parties involved in a virtual channel. There exists multi-party virtual state channel construction [21] built over Ethereum and make use of its Turing-complete language. However, a multi-party virtual payment channel built over the UTXO model and limited scripting language is an interesting open problem.

6.2.3. Post-Quantum Security

In Chapter 5, we address the following question:

Q5: *How to design a post-quantum secure adaptor signature, and thereby payment channel network?*

The fifth research question concerns adaptor signatures and their application in the payment channel networks. Adaptor signatures have been introduced with the name of scriptless scripts [22] to improve the capabilities of Bitcoin-like blockchains. Bitcoin's scripting language has very limited operations that hinder deploying conditional payments sophisticated than the few available scripts. Yet, with the help of adaptor signatures, the conditions of the payments can be beyond the scripting conditions. This is because, in an adaptor signature, a condition can be embedded into the signature. As the condition is not seen or validated by the miners, it is not restricted to the blockchain's scripting language. The condition is defined over a hard relation, and the full signature over the condition reveals a witness of it. An adaptor signature is built over a digital signature scheme, and they have been proposals over Schnorr [22], ECDSA [23] signature schemes, which are vulnerable to post-quantum attacks. An adaptor signature with an underlying post-quantum signature scheme has been an open problem until our work.

In this work, we present the first post-quantum adaptor signature scheme based on standard lattice assumptions, Module-SIS and Module-LWE. The underlying signature scheme is a simplified version of Dilithium [24], which is among the third round finalists of digital signature algorithms in NIST's Post-Quantum Cryptography standardization process [25]. We utilize the formalization of the adaptor signature schemes given in [20] to prove the security of our construction. Compared to Schnorr and ECDSA based solutions, adaptor signature construction from Dilithium has two technical difficulties. Firstly, the lattice-based scheme has a *rejection sampling* step to avoid leakage of information on the secret key. In this step, the infinity norm of the signature is checked, and if it is higher than the threshold, then the signing process restarts with new randomness. In adaptor signature construction, since the condition is not known to the signing party, we need to consider the size of the witness in the rejection step. Secondly, there is a *knowledge gap* in the zero-knowledge proof system underlying Dilithium, which means that there is a gap between the relations satisfied by an honest user's witness and a witness that can be extracted from the proof. We take into account these technical difficulties in our applications using the adaptor signatures and we extend the adaptor signature model introduced in [20] accordingly.

We build two applications using our post-quantum adaptor signature scheme, namely atomic swap and PCN. The first application of our scheme, atomic swap, is an exchange of two different cryptocurrencies among two parties where either both parties receive the expected coins or none do. For the post-quantum atomic swap protocol, we utilize the protocol given in [26], scriptless version of [27], and carefully handle the issues arising from the knowledge gap. Our second application is the design of the first post-quantum PCN. Using our adaptor signature scheme, we show how to realize anonymous multi-hop lock (AMHL) construction [23] which is adequate to build a PCN. Here, we also consider the effect of the number of hops in the witness size, thereby in the rejection sampling step. We elaborate on the limitation of this issue in the following section.

LIMITATIONS AND FUTURE WORK: WITNESS-INDEPENDENT APPROACHES

As we already mentioned, our construction inherits several technical difficulties from the underlying lattice-based signature scheme, Dilithium. We carefully design the applications to overcome these difficulties that may lead to some limitations in the applications.

First of all, because of the knowledge gap, we need to make sure that the adopted witness does not violate the rejection step. For that reason, in both applications, we provide proof of knowledge of the witness being sufficiently small, more precisely, having a infinity norm less than or equal to 1. These proofs would add computational and communication costs to the applications, and the size of such a proof is about 50KB [28]. Secondly, in our PCN, because of the AMHL construction, the size of the witness linearly increases with the length of the multi-hop payment. Thus, we limit the length of the multi-hop payment to have an upper bound on the size of the witness and update the rejection sampling step accordingly. In practice, this limit may not affect the functionality of the PCN as long as it is higher than the distance between parties. Indeed, Lightning Network implementations use the limit of 20 hops [29] though they do not have the same technical issue. Finally, as mentioned in a follow-up work [30], the witness size can leak information about the position of an intermediary party in the payment path. For example, if the infinity norm of the witness is about 1, then there is a high chance that the previous party is the sender of the payment. In [30], the authors present another post-quantum adaptor signature scheme to overcome the previously mentioned limitations. Their scheme relies on isogeny-based assumptions.

6.3. Conclusion

Blockchain, within less than a decade, became one of the most exciting technological developments. Among several interesting use cases and projects, there has been an inevitable hype in the industry as well. While in the research community, it has opened an interdisciplinary research field among cryptography, distributed systems, and economics. Blockchain is an evolving technology, and its integration into our daily life depends on improvements made by industry and research partners as well as necessary adjustments concerning legal and regulatory perspectives. The thesis has focused on the former improvements, and the latter concerns are out of the scope.

In this thesis, we investigated cryptocurrencies and addressed several open questions regarding security, scalability, and economical aspects. Our works can be divided into two subjects: transaction propagation and payment channel networks (PCN). First, we analyzed the existing transaction propagation mechanisms and proposed incentive-compatible and bandwidth-efficient ones suitable for peer-to-peer mining networks. The second subject of the thesis is the PCNs, which are promising layer-2 protocols aiming to improve the scalability of blockchains. As cryptocurrencies became more popular and active, scalability has become one of the most urgent research quests of the blockchain domain. In this thesis, we presented three works on the PCNs. We investigated the incentives to participate in multi-hop payments and proposed a profit strategy that would encourage the use of PCNs. Then, we proposed the first virtual channel constructions on payment channels that improve the efficiency and availability of multi-hop payments. Finally, we introduced the first post-quantum PCN utilizing our post-quantum adaptor signature scheme. Our works mainly focused on Bitcoin and its PCN, Lightning Network, yet they can be applied to the permissionless blockchains and cryptocurrencies having similar characteristics.

REFERENCES

[1] T. J. Housel, W. Baer, and J. Mun, *A new theory of value,* Intellectual Capital in Organizations: Non-Financial Reports and Accounts **1**, 16 (2014).

[2] E. G. Sirer, *Bitcoin runs on altruism,* (2015).

[3] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein, *Bitcoin mining pools: A cooperative game theoretic analysis,* in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, edited by G. Weiss, P. Yolum, R. H. Bordini, and E. Elkind (ACM, 2015) pp. 919–927.

[4] J. A. Kroll, I. C. Davey, and E. W. Felten, *The economics of bitcoin mining, or bitcoin in the presence of adversaries,* in *Proceedings of WEIS*, Vol. 2013 (2013).

[5] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, *Is bitcoin a decentralized currency?* IEEE Security & Privacy **12**, 54 (2014).

[6] A. E. Gencer, S. Basu, I. Eyal, R. van Renesse, and E. G. Sirer, *Decentralization in bitcoin and ethereum networks,* in *Financial Cryptography*, Lecture Notes in Computer Science, Vol. 10957 (Springer, 2018) pp. 439–457.

[7] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, *On bitcoin and red balloons,* in *ACM Conference on Electronic Commerce, EC '12, Valencia, Spain, June 4-8, 2012*, edited by B. Faltings, K. Leyton-Brown, and P. Ipeirotis (ACM, 2012) pp. 56–73.

[8] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, *Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus,* CoRR **abs/1612.02916** (2016), arXiv:1612.02916 .

[9] O. Ersoy, Z. Erkin, and R. L. Lagendijk, *Decentralized incentive-compatible and sybil-proof transaction advertisement,* in *MARBLE* (Springer, 2019) pp. 151–165.

[10] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, *Bitcoin-NG: A scalable blockchain protocol,* in *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, edited by K. J. Argyraki and R. Isaacs (USENIX Association, 2016) pp. 45–59.

[11] Statoshi Team, *Statoshi info,* `http://statoshi.info` (retreived 13 Feb. 2018).

[12] A. Biryukov, D. Khovratovich, and I. Pustogarov, *Deanonymisation of clients in bitcoin P2P network,* in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, edited by G. Ahn, M. Yung, and N. Li (ACM, 2014) pp. 15–29.

[13] P. Koshy, D. Koshy, and P. D. McDaniel, *An analysis of anonymity in bitcoin using P2P network traffic,* in *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, Lecture Notes in Computer Science, Vol. 8437, edited by N. Christin and R. Safavi-Naini (Springer, 2014) pp. 469–485.

[14] S. B. Venkatakrishnan, G. C. Fanti, and P. Viswanath, *Dandelion: Redesigning the bitcoin network for anonymity,* POMACS **1**, 22:1 (2017).

[15] O. Ersoy, Z. Erkin, and R. L. Lagendijk, *TULIP: A fully incentive compatible blockchain framework amortizing redundant communication,* in *EuroS&P Workshops* (IEEE, 2019) pp. 396–405.

[16] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, *Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract],* SIGMETRICS Performance Evaluation Review **42**, 34 (2014).

[17] S. Micali, *ALGORAND: the efficient and democratic ledger,* CoRR **abs/1607.01341** (2016), arXiv:1607.01341 .

[18] E. Bergamini, P. Crescenzi, G. D'Angelo, H. Meyerhenke, L. Severini, and Y. Velaj, *Improving the betweenness centrality of a node by adding links,* ACM Journal of Experimental Algorithmics **23** (2018), 10.1145/3166071.

[19] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, *Perun: Virtual payment hubs over cryptocurrencies,* in *IEEE Symposium on Security and Privacy* (IEEE, 2019) pp. 106–123.

[20] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi, *Generalized bitcoin-compatible channels,* IACR Cryptol. ePrint Arch. **2020**, 476 (2020).

[21] S. Dziembowski, L. Eckey, S. Faust, J. Hesse, and K. Hostáková, *Multi-party virtual state channels,* in *EUROCRYPT (1),* Lecture Notes in Computer Science, Vol. 11476 (Springer, 2019) pp. 625–656.

[22] A. Poelstra, *Lightning in scriptless scripts,* Mailing list post (), https://lists.launchpad.net/mimblewimble/msg00086.html.

[23] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, *Anonymous multi-hop locks for blockchain scalability and interoperability,* in *NDSS* (The Internet Society, 2019).

[24] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, *Crystals–Dilithium: Digital signatures from module lattices,* in *CHES*, Vol. 2018-1 (2018) https://eprint.iacr.org/2017/633.pdf.

[25] NIST, *Post-quantum cryptography – call for proposals,* (2017), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization/Call-for-Proposals, accessed on March 25, 2021.

[26] A. Poelstra, *Adaptor signatures and atomic swaps from scriptless scripts,* (), https://github.com/ElementsProject/scriptless-scripts/blob/master/md/atomic-swap.md.

[27] T. Nolan, *Alt chains and atomic transfers,* https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949.

[28] M. F. Esgin, N. K. Nguyen, and G. Seiler, *Practical exact proofs from lattices: New techniques to exploit fully-splitting rings,* in *ASIACRYPT (2)*, Lecture Notes in Computer Science, Vol. 12492 (Springer, 2020) pp. 259–288.

[29] *Basis of lightning technology,* Available at: https://github.com/lightningnetwork/lightning-rfc/blob/master/00-introduction.md.

[30] E. Tairi, P. Moreno-Sanchez, and M. Maffei, *Post-quantum adaptor signature for privacy-preserving off-chain payments,* IACR Cryptol. ePrint Arch. **2020**, 1345 (2020).

Acknowledgements

Here comes the end of my Ph.D. journey. It was not easy for me to come to a new country and be away from my family and friends for such a long time. In addition to my personal issues, I experienced Covid-19 and lockdown in the last part of my Ph.D. Luckily, I had my family, friends, and colleagues that supported me the whole time. I would like to thank the people I mention here and many others who made this journey quite enjoyable.

First, I would like to thank my promotor Inald Lagendijk. It was a great opportunity to work with you. The way you question my research problems helped me to improve my critical thinking. I hope you also enjoyed my unexpected jokes during our meetings. I will try not to joke during my defense ceremony.

I would like to deeply thank my daily supervisor Zeki Erkin. He was not only my supervisor but also my mentor and a close friend. I could not imagine that our casual chat at Crypto Days Symposium would lead me here. I have to confess now that I fell asleep and missed your talk at the symposium :) I was quite lucky that you were looking for a Ph.D. student while I was looking for a position in Europe. Thank you for the amazing four years. I really enjoyed the vacations, sorry conferences, that we went to great venues. I hope you will continue to mentor me.

I would like to thank committee members Matteo Maffei, Christian Cachin, Michel van Eeten, Dick Epema, and Jos Weber for taking their time to review my thesis and being part of my defense ceremony. Also, I thank Job and Jelle for helping me with the translation of the summary into Dutch.

During my Ph.D., I had the chance to collaborate with many researchers. I would like to thank Zhijie Ren, Stefanie Roos, Lukas Aumary, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, Siavash Riahi, Muhammed Esgin, Ziya Alper Genç and Mauro Conti. I appreciate the valuable discussions and interesting collaborations. I want to particularly thank Pedro Moreno-Sanchez and Matteo Maffei for the research visit opportunity in TU Wien. It was a great research experience, but also I had amazing three months in Vienna together with the Security and Privacy group.

I would like to thank members of the Cyber Security group Stjepan, Sicco, Christian, Phil, Peter, and Jan for our interesting discussions near the coffee corner during the pre-Covid period. I also would like to thank our secretary Sandra for her help.

Our Ph.D. office and the floor were full of joy and cake before the lockdown. Thanks to Zeki, at some point, we were eating cake every day (of course to celebrate his birthday). I would like to thank my officemates Gamze, Chibuike, Majid. Gamze, thank you for being a great friend. Having a similar background and mindset helped us understand each other quite well and led to many interesting discussions in our Turkish group. You were (and still are) organizing exciting and fun activities that made our PhDs quite enjoyable. Chibuike, I guess I will never know how early you were coming to the office; you were always there before me. I thank you and Niki for letting me play with Adanna and Mmasinachi. Majid,

our PhDs did not overlap that much, but it was enough to have knowledgeable discussions with you. Also, thank you for helping me move with your car.

Other than our small Ph.D. circle, I had the chance to know great colleagues in our group. Harm thank you for introducing me the great sandwich place in the city center. I do not know how I missed it for so many years. Also, I would like to thank you and Rienke for including me with several Dutch traditions like Christmas cards and birthday circles. The other members of the other Ph.D. office, Vincent, Mark, Qin, thank you for the interesting discussions near the coffee machine. Azqa, Huimin, Marina, Miray, Tianyu, Felix, Clinton and Luca unfortunately, our office sharing was during the Covid days, and we could not spend enough time together in the office. I hope we can meet more often for non-work-related occasions. I am sure that Marina can come up with extreme, weird, and fun activities that Azqa will hesitate to join because of the crowd (even after the Covid), and Miray will, of course, find something to complain about. I also have to mention my favorite twins, Daniel, Jelle. I remember your bachelor group working with Zeki at the beginning of my Ph.D. Now, you are both doing a Ph.D. which also indicates that I should move on to new adventures.

There were also short visitors of our group: master students. Since I have been there for ages, I had the chance to meet with several generations. From ancients to the newest, I was great pleasure to meet you Asta, Bjorn, Christian, Daphne, Dirk, Felix, Ginger, Hari, Jehan, Lars, Leon, Martin, Mathijs, Mourad, Pieter, Rasmus, Roemer, Rogier, Sjors, Victor. Christian, thank you for constantly beating me on chess after so many years. Hari, thank you for your amazing company and your delicious food. Now, I am more tolerant of spicy food. Also, thank you for introducing many great animes and movies to my life.

During my Ph.D., I had the chance to supervise 5 master students, Mourad, Bjorn, Rasmus, Breus, and Jehan. Thank you all, it was a great experience for me, and I learned a lot together with you. Mourad, I am sure that you are still the funny guy of your new group, but please let them work as well. Bjorn, it was nice working with an organized person. Rasmus, you were a unique person; I had a lot of fun with you, especially with go-karting. Jehan, you are a great person but also extremely chaotic. I am glad that I had this fun challenge; you were my Nirvana with supervision.

In addition to our research group, I was lucky to have many interesting friends. Here, I should mention Osman and Taygun. Our gatherings were mostly connected with food and Turkish tea. Thank you for the unique and interesting discussions with the combination of Taygun's knowledge and Osman's pop culture and comedy references. When it comes to discussion, I have to acknowledge Baran. It is nice to know such a provocative character who fed from heated debates. Together with Salim and Melis, we had amazing food (of course, the food was always delicious) and passionate discussions on everything. I thank you for all, and particularly for my cozy Covid-proof 30th birthday. I would also like to thank my soccer, bouldering and table tennis friends, Ekin, Jose, Ramin, Tamim, and many others. Thanks to you, even I scored hat-trick in multiple matches :) I hope soon we can start again. Finally, I would like to mention my new postdoc group: Can, Jeremy, Martijn, Masoud, Satwik, and many other colleagues. I am quite happy and excited to work with you.

I would like to especially thank Bilal and Zeynep. When I moved to the Netherlands, you were the only two friends that I had here. I enjoyed a lot our trips to many parts of the

country. It is unfortunate that you moved to another country now. I also thank my old virtual friends from the bachelor period, Hüseyin, Erhan, Erdem, Furkan, Sinan, Özgür, Raşit and Murat. Even though we were not physically together, we keep teasing each other all the time. Erhan, Erdem, and Furkan thank you for visiting me here, and I hope to see you soon. Hüseyin, thank you for trying to visit me, maybe next time you will manage :) Also, thank you for teaching me Catan and letting me win once in a while. I also would like to thank my old colleagues, Muhammed and Ziya. Even though we were doing PhDs in different countries and our daytime did not match, we were finding time for short 5 mins chats that usually took an hour. Our non-work-related discussions always ended up with research and eventually resulted in academic collaborations.

Last but the most, I would like to thank my family for everything. I could not achieve anything without their constant support. First, I deeply thank my parents Ayşe Yaşar and İbrahim for all the sacrifices and commitments they made for us to have these opportunities. I wish my mother was able to see the results of her efforts. I deeply thank my father for his unending support in this journey. Thank you for calling me every single day and for long discussions on the news, sports, or family concerns that help me distract myself from the Ph.D. I guess you are now an expert on the procedures of a Ph.D. since you were tracking my progress very closely. I am also extremely lucky to have two amazing siblings: Esra and Mehmet Akif. I thank my sister for always being there for me. I am sure you will be happy to have another Dr. in our family, but this one will not be medical. I also thank you and my brother-in-law, Fetih, for raising two amazing kids, Melih Kerem and Alper Semih. Though I am mostly away from my nephews, we always have our video calls. Finally, I thank my twin brother. I cannot explain how grateful I am for having a twin like you. It is a great feeling to know that no matter what, you will be there for me. You are also the funniest person I know, and I am lucky to get all of your references. Until my Ph.D., we were always together, I could not convince you for a Ph.D., but I hope you will come nearby for your work.

Finally, Ph.D. was a period that I rediscovered myself. I tried to become a better and happier person in addition to the research part. I got a couple of new hobbies, including chess, painting, and bouldering, and yes, I am still terrible at them. I made many great friends who also gave me many nicknames. To name a few: Qubit for my indecisiveness, Big-O, The Wizard (of Oz) because why not, and finally Ozzy, which is the one I am keeping for now. Overall, I tried to get the best out of this journey, and I believe that it was a success. Thank you for being part of it :)

Curriculum Vitæ

Oğuzhan Ersoy

Oğuzhan Ersoy was born on August 20, 1990 in Erzurum, Turkey. He obtained his B.Sc. degrees with a double major in Electrical & Electronics Engineering and Mathematics in 2012 and an M.Sc. degree in Electrical & Electronics Engineering in 2015 at Boğaziçi University, Istanbul, Turkey. In 2012, he started working as a researcher at the department of Cryptographic Architecture and Algorithms at TÜBİTAK BİLGEM UEKAE (National Research Institute of Electronics and Cryptology), Kocaeli, Turkey. Later on, in 2016, he became a senior researcher in the same department. In 2017, he joined the Cyber Security Group of Delft University of Technology as a Ph.D. student.

He was ranked in the top 100 in the national high school entrance exam among 650K participants in 2004. He obtained a silver medal in the national primary mathematics olympiad in 2004 and a bronze medal in the national mathematics olympiad in 2006. He was ranked in the top 300 in the national university entrance exam among 1.6M participants in 2007. He received B.Sc. honor degrees in Electrical & Electronics Engineering and Mathematics in 2012.

During his Ph.D., he worked on blockchain technology under the supervision of Prof.dr.ir. R.L. Lagendijk and Dr. Z. Erkin. He supervised five master students and several bachelor students for their end projects. He contributed to Security and Cryptography, Blockchain Engineering, and Bachelor Seminar courses as a teaching assistant. He visited TU Wien in Austria as a guest researcher and worked under the supervision of Prof. M. Maffei. Currently, he is continuing his research career as a post-doctoral researcher in the Distributed Systems Group of Delft University of Technology.

List of Publications

Journal

3. **Ersoy, O.**, Pedersen, T. B., and Anarim, E. Homomorphic extensions of CRT-based secret sharing. *Discrete Applied Mathematics 285* (2020), 317–329.

2. **Ersoy, O.**, Kaya, K., and Kaskaloglu, K. Multilevel threshold secret sharing based on the chinese remainder theorem. *International Journal of Information Security Science 8*, 2 (2019), 17–29.

1. **Ersoy, O.**, Pedersen, T. B., Kaya, K., Selçuk, A. A., and Anarim, E. A CRT-based verifiable secret sharing scheme secure against unbounded adversaries. *Security and Communication Networks 9*, 17 (2016), 4416–4427.

Conference and Workshop

12. **Ersoy, O.**, Genç, Z. A., Erkin, Z., and Conti, M. Practical Exchange for Unique Digital Goods. In *IEEE DAPPS* (2021), (in press).

11. Aumayr, L., **Ersoy, O.**, Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., and Riahi, S. Bitcoin-compatible virtual channels. *IEEE Symposium on Security and Privacy* (2021), (in press).

10. Esgin, M. F., **Ersoy, O.**, and Erkin, Z. Post-quantum adaptor signatures and payment channel networks. In *ESORICS (2)* (2020), vol. 12309 of *Lecture Notes in Computer Science*, Springer, pp. 378–397.

9. **Ersoy, O.**, Roos, S., and Erkin, Z. How to profit from payments channels. In *Financial Cryptography* (2020), vol. 12059 of *Lecture Notes in Computer Science*, Springer, pp. 284–303.

8. **Ersoy, O.**, Erkin, Z., and Lagendijk, R. L. TULIP: A fully incentive compatible blockchain framework amortizing redundant communication. In *EuroS&P Workshops* (2019), IEEE, pp. 396–405.

7. **Ersoy, O.**, Erkin, Z., and Lagendijk, R. L. Decentralized incentive-compatible and sybil-proof transaction advertisement. In *MARBLE* (2019), Springer, pp. 151–165.

6. el Maouchi, M., **Ersoy, O.**, and Erkin, Z. DECOUPLES: a decentralized, unlinkable and privacy-preserving traceability system for the supply chain. In *SAC* (2019), ACM, pp. 364–373.

5. VAN DER LAAN, B., **ERSOY, O.**, AND ERKIN, Z. MUSCLE: authenticated external data retrieval from multiple sources for smart contracts. In *SAC* (2019), ACM, pp. 382–391.

4. **ERSOY, O.**, REN, Z., ERKIN, Z., AND LAGENDIJK, R. L. Transaction propagation on permissionless blockchains: Incentive and routing mechanisms. In *CVCBT* (2018), IEEE, pp. 20–30.

3. EL MAOUCHI, M., **ERSOY, O.**, AND ERKIN, Z. Trade: A transparent, decentralized traceability system for the supply chain. In *Proceedings of 1st ERCIM Blockchain Workshop 2018* (2018), European Society for Socially Embedded Technologies (EUSSET).

2. KARAKOÇ, F., SAGDIÇOGLU, Ö. M., GÖNEN, M. E., AND **ERSOY, O.** Impossible differential cryptanalysis of 16/18-round khudra. In *LightSec* (2016), vol. 10098 of *Lecture Notes in Computer Science*, Springer, pp. 33–44.

1. BAY, A., **ERSOY, O.**, AND KARAKOÇ, F. Universal forgery and key recovery attacks on ELmD authenticated encryption algorithm. In *ASIACRYPT (1)* (2016), vol. 10031 of *Lecture Notes in Computer Science*, pp. 354–368.

PREPRINT

1. AUMAYR, L., **ERSOY, O.**, ERWIG, A., FAUST, S., HOSTAKOVA, K., MAFFEI, M., MORENO-SANCHEZ, P., AND RIAHI, S. Generalized Bitcoin-compatible channels. *IACR Cryptol. ePrint Arch. 2020* (2020), 476.