MONASH University

Layer-2 Solutions for Bitcoin Scalability

by

Arash Mirzaei,

Thesis

Submitted by Arash Mirzaei

for fulfillment of the Requirements for the Degree of

Doctor of Philosophy (0190)

Supervisor: Dr. Amin Sakzad

Associate Supervisor: A/Prof. Ron Steinfeld, Dr. Jiangshan Yu

Faculty of Information Technology
Monash University

July, 2023

Layer-2 Solutions for Bitcoin Scalability

Arash Mirzaei,

Monash University, 2023

Supervisor: Dr. Amin Sakzad
Associate Supervisor: A/Prof. Ron Steinfeld, Dr. Jiangshan Yu

Abstract

Payment channels are an effective layer-2 approach to improve the scalability of blockchain-based cryptocurrencies. A payment channel allows two parties to execute transactions off-chain. But since each party's share of coins in a channel changes over time, one party might attempt to close the channel with an old state to maximise her profit. In practice, when a channel party publishes a channel state on the blockchain, a period called *dispute period* starts, in which their counter-party can provide some evidence that proves the invalidity of the published state and hence prevents the channel from getting finalised with an old state. Several payment channels have been proposed for Bitcoin, each with its own limitations. Some issues with existing payment channels include: (1) *state duplication* meaning each party has its own version of transactions leading to complex transaction flows, (2) lack of a penalisation mechanism against dishonest parties and hence incentive incompatibility, (3) incompatibility of the payment channel with some important digital signature algorithms such as BLS which limits the payment channel usage in blockchains with such underlying digital signature algorithms, and (4) increase in the storage requirements of the channel parties over time.

Since the dispute period length is limited, channel parties must be always online and synced with the blockchain. Otherwise, they might fail to react to their counter-parties' misbehaviour in time. However, it is impossible for ordinary users to stay always online. Thus, parties might delegate this task to a third party, called the *watchtower*. Multiple watchtower schemes have been proposed in the literature, each with its own limitations; Some are unfair against channel parties or the watchtower (i.e. an honest channel party or a responsive watchtower might lose some funds in the channel); Some do not provide privacy preserving properties and some cannot be deployed on Bitcoin.

Therefore, there are some limitations in existing payment channels and their watchtower schemes, particularly in those that have been designed for blockchains with limited scripting languages. Thus, we focus on analysing the mentioned limitations and designing payment channels and watchtower schemes that outperform the existing ones. Hence, our contributions are as follows:

- We formalise watchtowers and their properties including *agility*, *privacy*, *fairness* and *coverage*. Furthermore, we prove some trade-offs between the above-mentioned properties. We also focus on different properties of a payment channel and examine the limitations of existing payment channels in achieving the required properties.

- We design a new watchtower scheme for Bitcoin, called FPPW, which is fair with respect to both channel parties and the watchtower. Therefore, the honest party's funds in the channel are safe and the honest watchtower's rewards are also guaranteed. Furthermore, FPPW provides privacy against the watchtower as the watchtower obtains no data on the way channel funds are redistributed among channel parties. Focusing on storage costs, we also design a second watchtower scheme for Bitcoin, called Garrison, where the storage costs of channel parties and their watchtower increase logarithmically with the number of channel updates. Additionally, Garrison avoids state duplication. Both FPPW and Garrison can be implemented without any update in Bitcoin blockchain.

- Relying on the deployment of the `ANYPREVOUT` signature type in Bitcoin, we also present a payment channel with an unlimited lifetime for Bitcoin, called Daric, that achieves optimal storage. Daric is the first payment channel that provides penalisation against the dishonest channel party and simultaneously avoids state duplication without relying on any particular property for the underlying digital signature. We also prove the security of Daric in the Universal Composability model.

Layer-2 Solutions for Bitcoin Scalability

Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Arash Mirzaei
July 9, 2023

Publications

Published works (included in the thesis):

- Arash Mirzaei, Amin Sakzad, Jiangshan Yu, Ron Steinfeld. FPPW: A Fair and Privacy Preserving Watchtower For Bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 151-169. Springer, 2021.
- Arash Mirzaei, Amin Sakzad, Jiangshan Yu, Ron Steinfeld. Garrison: A Novel Watchtower Scheme for Bitcoin. In *Australasian Conference on Information Security and Privacy*, pages 489-508. Springer, 2022.
- Arash Mirzaei, Amin Sakzad, Jiangshan Yu, Ron Steinfeld. Daric: A Storage Efficient Payment Channel With Penalization Mechanism. In *International Conference on Information Security*, pages 229-249. Springer, 2022.

Other works during my PhD (not included in the thesis):

- Arash Mirzaei, Amin Sakzad, Ron Steinfeld, Jiangshan Yu. Algorand blockchain, in *Blockchains - A Handbook on Fundamentals, Platforms and Applications*, Lecture Notes in Computer Science (LNCS).
- Xianrui Qin, Shimin Pan, Arash Mirzaei, Oguzhan Ersoy, Amin Sakzad, Muhammed Esgin, Joseph K. Liu, Jiangshan Yu, Tsz Hon Yuen. BlindHub: Bitcoin-Compatible Privacy-Preserving Payment Channel Hubs Supporting Variable Amounts, In *IEEE Symposium on Security and Privacy*, 2023.

Acknowledgments

I would like to express my heartfelt gratitude to my main supervisor Amin Sakzad for all his endless support, guidance and encouragement in various aspects both on a professional and personal level. His support started even before the commencement of my PhD and went beyond my expectations throughout my research. I also thank my co-supervisors Ron Steinfeld and Jiangshan Yu for their help and support. Their expertise, insights, and valuable feedback helped me to achieve my academic goals. I am also grateful to my wife Lili for her unwavering support, patience, and love throughout this journey. Her understanding, encouragement, and sacrifice have been invaluable in enabling me to balance my academic pursuits with my personal life. I could not have achieved this without her support and I am forever thankful.

Moreover, I am thankful to my panel committee members Carsten Rudolph, Joseph Liu, Rafael Baiao Dowsley and Shifeng Sun for their constructive feedback. I thank the Faculty of Information Technology and the Graduate Research Office at Monash University for providing me the scholarship to cover the tuition fee and my cost of living during my PhD study. I also owe great thanks to my fellow PhD students Ahmad, Sara and Rumpa for their moral support.

Finally, I would like to take this opportunity to acknowledge the bravery and resilience of Iranian men and women who have fought for their rights and freedom. During my PhD study, I was moved by the story of Mahsa Amini, a young Iranian woman who was killed while in police custody. Her tragic death, along with the deaths of others including kids who were killed during the "*Women Life Freedom*" protests, the blinding of many individuals, and the imprisonment of countless others, is a stark reminder of the courage and determination of their unwavering commitment to the principles of human rights, democracy, and gender equality. May their sacrifice and courage never be forgotten, and may their struggle continue to inspire and guide us in the fight for justice, freedom, and equality.

Arash Mirzaei

Monash University
July 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Payment Channel: Bitcoin [2], the pioneering cryptocurrency, has emerged as a significant innovation with far-reaching implications. Its decentralized nature, secure transactions, and potential to reshape the financial landscape highlight its importance as a transformative force in the digital era. Scalability is of paramount importance in the context of Bitcoin as it directly affects its ability to function as a global, mainstream currency. With the growing adoption and transactional demands, ensuring a scalable infrastructure becomes crucial for Bitcoin to maintain efficiency, low fees, and widespread usability, ultimately shaping its potential for broader acceptance and impact.

However, Bitcoin scalibility is still a grand open challenge limiting the adoption of blockchain technologies [3, 4, 5]. For instance, Bitcoin can only process about 10 transactions per second on average [6, 7], which is much lower than the figures for centralised payment systems such as Visa [8]. This issue has led to several solutions: (1) modifications in consensus protocols [9, 10], (2) sharding [11, 12], (3) side-chains [13], and (4) layer-2 or off-chain protocols [14, 4, 15], where this thesis targets the last solution.

Layer-2 protocols are so named because they operate on top of the base layer of the Bitcoin blockchain. These solutions aim to alleviate scalability issues by conducting off-chain transactions, reducing the burden on the main blockchain. By implementing layer-2 solutions, Bitcoin can achieve faster and more cost-effective transactions, increased capacity, and improved scalability while maintaining the security and decentralization features of the underlying blockchain. Notably, layer-2 protocols do not require any modifications in the blockchain and its consensus mechanism. Layer-2 protocols are deployed using the scripting language of the underlying blockchain and rely on two main properties of the first layer blockchain [16]: *integrity* and *eventual synchronicity with an upper time-bound*, where the former means only valid transactions can be published on

the blockchain and the latter implies every submitted valid transaction is finally added to the blockchain before a limited timeout.

There are various types of off-chain solutions: (1) Payment channel [15, 4, 14] allow participants to create a secure off-chain channel for conducting multiple transactions without each transaction being recorded on the main blockchain (2) State channels [17] expand upon the idea of payment channels to encompass the execution of arbitrary applications, extending their functionality beyond simple payment transactions, and (3) Commit-chains [18, 19] allow transactions to be facilitated through the involvement of a centralized yet untrusted intermediary, enabling communication between participating parties. Given their simplicity, widespread adoption and significance in addressing scalability challenges, payment channels take center stage in this thesis.

A payment channel between two parties Alice (or A) and Bob (or B) allows them to perform a number of transactions without publishing every single transaction on the blockchain. To create a channel, A and B respectively deposit a and b coins into a joint address that is controlled by both parties. Parties can privately update their balance in the channel by exchanging off-chain transactions and agreeing on a new distribution of channel funds. Each party can close the channel at any time by enforcing the latest channel state on the blockchain. Payment channels can also be linked to form a Payment Channel Network (PCN), where payments between users with no direct payment channel can be routed via intermediaries.

Since the channel parties are generally untrustworthy and blockchain miners are unaware of the off-chain transactions, a mechanism must be adopted to prevent potentially cheating parties from publishing an old state. To achieve that, Lightning Network [15], as the most popular payment channel network, adopts a punishment mechanism to prevent parties from acting dishonestly. So upon authorising a new state, channel parties exchange some revocation secrets to revoke the previous state. Then, if a party publishes a revoked state, her counterparty who is supposed to be always online uses the corresponding revocation secrets and reacts within a limited time period, called *dispute period*, to take all the channel funds.

Although elegantly designed, the Lightning Network has some shortcomings. Firstly, since channel parties must store all the revocation secrets, received from their counterparties, their storage amount increases with the number of channel updates. Moreover, to detect and punish the misbehaving party, the channel state is duplicated meaning each party has its own copy of the state. To solve the former issue, eltoo [1] removes the punishment mechanism causing incentive incompatibility. To solve the second issue, Generalized channel [14] uses a dedicated design of adaptor signatures which introduces compatibility issues with BLS [20] or most post-quantum digital signatures.

Watchtower The dispute process works based on the assumption that the parties are always online and synced with the blockchain to detect malicious behaviour. This requirement has always been a drawback for payment channels because it can be practically violated due to crash failures or by performing DoS attacks on channel parties. *Watchtower* was introduced to relax this strong assumption by allowing users to delegate the watching tasks to watchtower services [15]. Watchtowers are always-online services that monitor the blockchain and act on behalf of their clients to secure their funds.

Monitor [21] is the first watchtower scheme for the Lightning Network which mainly focuses on channel privacy against watchtower. However, Monitor has two main issues, both of which are related to *fairness*. Firstly, honest watchtowers might be rewarded for fraud (i.e. broadcast of an old state on-chain), which is unfair with respect to (w.r.t.) the watchtower. Secondly, honest parties cannot penalise the unresponsive watchtower, which is unfair towards an honest hiring party. Moreover, the storage costs of the watchtower in Monitor increase linearly with each channel update. Outpost [22] solves the issue of fairness towards the watchtower by paying her per channel update. It also improves the storage requirements of the watchtower.

Cerberus [23] and PISA [24] elegantly provide fairness w.r.t. the hiring party by enforcing the watchtower to lock some collateral that is given to their clients given that the watchtower is unresponsive upon dispute. However, PISA fails to be deployed on Bitcoin and Cerberus sacrifices privacy. In particular, the Cerberus watchtower learns the distribution of funds in the channel. Moreover, since watchtowers in PISA and Cerberus have to lock some collateral per channel, their *coverage* or equivalently their capability in watching all the existing payment channels on a fixed Blockchain is limited. Therefore, each watchtower scheme focuses on some particular properties among privacy, storage requirements of the watchtower or fairness. However, none of the existing Bitcoin-compatible watchtower schemes can achieve all the desired features.

Therefore, the main objective of this thesis is the analysis of different limitations of existing payment channels and their watchtowers and then moving towards improving different aspects of the existing schemes. Accordingly, this thesis answers the following research questions (RQ):

RQ 1 How to formally define different properties of a watchtower scheme? Why do existing watchtower schemes fail to meet all the required properties?

RQ 2 How to design a watchtower scheme for existing payment channels that mitigates the limitations of existing schemes?

RQ 3 What are other limitations of the existing payment channels? How to design a payment channel solution that mitigates the limitations of existing payment channel proposals? How to analyse the security of the proposed payment channel?

1.1 Contributions

In this thesis, we analyse the existing payment channel and watchtower schemes. Then, focusing on the shortcomings of the current schemes, we move towards design of new payment channel and watchtower schemes with better properties. In more detail, the contributions of this thesis are as follows:

- **Formal Definition of a Watchtower and its Required Properties:** For RQ 1, we formalise the definition of a watchtower and its different properties (See Chapter 4). Those properties are as follows:
  - Agility: flexibility of a watchtower to start and terminate its service,
  - Privacy: the amount of knowledge the watchtower (or respectively any third party) obtains about the payment channel (or respectively about the hiring status of the watchtower).
  - Fairness: the level of guarantee that the watchtower (or respectively the hiring party) provides to the hiring party (or respectively to the watchtower) on its service (or respectively on its payment), and
  - Coverage: the capability of a watchtower (on a scale between 0 to 1) in watching all the existing payment channels on a fixed Blockchain.

  We also evaluate the existing schemes regarding these properties and show that existing proposals fail to provide acceptable results in all aspects. Furthermore, we prove that there is a trade-off between the level of fairness that a watchtower provides to its clients and the coverage it can achieve.

- **Design and Analysis of Two New Watchtower Schemes:** For RQ 2, we design two watchtower schemes: FPPW (See Chapter 5) and Garrison (See Chapter 6). FPPW focuses on the properties we introduced earlier and provides fairness w.r.t. all channel participants including both channel parties and the watchtower. It means that the funds of any honest channel participant are safe even assuming that the other two channel participants are corrupted and/or collude with each other. Furthermore, the watchtower in FPPW learns no information about the off-chain transactions and hence the channel privacy is preserved. Furthermore, we

show that FPPW's coverage is higher than other existing schemes. FPPW can be implemented without any update in the Bitcoin script.

While FPPW offers desirable properties, it is essential to consider that the storage costs for channel parties and watchtowers increase linearly with each channel update. Thus, we design Garrison to achieve lower storage costs. Notably, the storage cost for both channel parties and their watchtower in Garrison would be $\mathcal{O}(\log N)$ with N being the number of channel updates. Furthermore, using properties of the adaptor signature, Garrison avoids state duplication. It means both parties store the same version of transactions for each state and hence the number of off-chain transactions does not exponentially increase with the number of applications built on top of each other in the channel. Moreover, independent of the complexity of the published revoked state, the honest party or his watchtower in a Garrison channel publishes a single transaction to react upon dispute. Garrison can also be implemented without any update in the Bitcoin script.

- **Deisgn and Analysis of a New Payment Channel Scheme:** For RQ 3, we discuss the required properties of a payment channel and analyse the limitations of existing payment channels in achieving the required properties, i.e. unlimited lifetime, fixed storage costs, punishment mechanism, compatibility with any digital signature algorithms, state duplication avoidance and *bounded closure* where the latter enables channel parties to close the channel within a bounded time. Then, we introduce Daric (See Chapter 7), a payment channel with an unlimited lifetime for Bitcoin that achieves optimal storage (constant storage for both channel parties and their watchtower) and bounded closure. Moreover, Daric implements a punishment mechanism and simultaneously avoids the methods other schemes commonly used to enable punishment: (1) state duplication which leads to an exponential increase in the number of transactions with the number of applications on top of each other or (2) dedicated design of adaptor signatures which introduces compatibility issues with BLS or most post-quantum digital signatures. We also formalise Daric and prove its security in the Universal Composability model.

1.2 Thesis Structure

In Chapter 2, we review the existing payment channel and watchtower schemes. Chapter 3 presents the background, preliminaries and notations. Chapter 4 formalises the watchtower and its different properties. Two proposed watchtower schemes, i.e. FPPW and Garrison, will be presented in Chapter 5 and Chapter 6, respectively. Chapter 7 presents our proposed payment channel scheme, i.e. Daric. We conclude this thesis and discuss the potential future works in Chapter 8.

Chapter 2

Background and Literature Review

2.1 Background

Bitcoin [2], as the first cryptocurrency, is a collection of concepts and technologies that can be used to make payments without relying on any central authorities like banks or financial institutes. Bitcoin users who are connected over the Internet can transfer units of currency, called bitcoin, in exchange for some goods or services. However, unlike traditional payment systems, bitcoins are fully virtual. It means that there are not any physical coins to be transferred between the sender and the recipient. So units of value in the Bitcoin system are transferred among users by exchanging some digital transactions that transfer bitcoin values from the sender to the recipient. To prevent users from spending each other's coins, each coin in this system corresponds with a key pair of a public key cryptosystem whose private key is only possessed by the coin owner. So, to create a transaction and send some coins to a new owner, the current owner needs to prove the ownership of his coins by signing the transaction using his corresponding private key. This transaction transfers the coins to a new owner with his own key pair whose public key is recorded in the transaction.

Bitcoin is a fully distributed, peer-to-peer system. Thus, there is not any central authority to validate and store the users' transactions. Therefore, the important question is what can prevent a user from double spending, i.e. sending the same coins more than once. For fiat currencies, notes or coins carry some special features that can only be made by a central authority. So counterfeited money would be detectable. Also, when we use our mobile banking app to transfer money over a digital medium, one or multiple central authorities like banks record the corresponding transactions. So they prevent double spending. However, due to its decentralised nature, Bitcoin cannot use these mechanisms to avert double spending. As will be explained in the following, blockchain technology is the solution to this issue [2].

The Bitcoin blockchain [2] is an immutable public distributed ledger that records Bitcoin transactions. Let's explain these terms further. Each block is a limited-size list of Bitcoin transactions that are added to the blockchain data structure one at a time. The term public distributed ledger refers to the fact that rather than storing in a central database, blockchain data is distributed to a large number of independent nodes (also known as miners) and everyone is also capable to join the node community. Transactions in the blockchain are immutable meaning that once a transaction is added to the blockchain, it is computationally impossible to change its value or remove it from the ledger unless the block containing the transaction and all subsequent blocks are altered. This property is achieved as each block, along with its list of transactions, contains a cryptographic hash of the previous block. In this way, each block in the blockchain is chained to all previous blocks up to the first block in the blockchain.

Now that the ledger in the Bitcoin blockchain is distributed to many nodes, these nodes need a way to reach an agreement on the data that is going to be added to the ledger. In more detail, Bitcoin users constantly send their transactions to a publicly available peer-to-peer network. Then, miners validate these transactions and list them to form a block. However, due to the fact that these nodes are geographically distributed, they receive these transactions with different orders. Sometimes, due to network issues, some transactions might not be received by some blockchain nodes. Thus, their view of the submitted transactions and the latest created block would be different from each other. Therefore, another important building block in blockchain technology is how blockchain nodes reach a consensus. The consensus mechanism that is used in Bitcoin is called Proof-of-Work (PoW) [2, 25]. In this method, after creating the block, each node solves a computation-heavy cryptographic puzzle. The first node that succeeds in solving this puzzle would win and add the corresponding block to the blockchain. In order to incentivise the nodes to perform such heavy computations, the winner is also rewarded with some newly generated Bitcoins as well as some transaction fees.

The Bitcoin blockchain cryptographic puzzle is set such that each block is added to the blockchain every 10 minutes. Due to the fact that the size of each block is also limited to 1 MB, Bitcoin is significantly slower than centralised payment systems such as Visa [15]. A payment channel [15, 26, 4] is a promising solution to this issue where two channel parties open a channel on the blockchain by publishing a transaction on the blockchain through which they send their coins to a joint account that needs signatures by both parties to be spent. Then, they exchange off-chain transactions outside of the Bitcoin blockchain where those transactions send the coins in the joint account to each party. Each party can submit the latest channel transaction to the Bitcoin network to close the channel. Channel parties can also use their channels to create a network of payment channels, called Payment Channel Network (PCN), to perform indirect payments

through intermediaries [15, 14]. In practice, some routing algorithms are required to find the appropriate intermediaries for this purpose. In more detail, the indirect payment would be feasible if all channels along the route have enough balances in the desired direction. Otherwise, the payment fails. Thus, the routing algorithms intend to find the best existing paths from the payer to the payee for a given payment.

2.2 Payment Channel

A payment channel is a state machine run by two participants and its lifetime consists of three main phases including *create*, *update* and *close* (See Fig. 2.1). In the channel create phase, parties lock their funds in a joint account whose value can only be spent upon both parties' agreement. In the channel update phase, parties agree on a new channel state. Finally, in the channel close phase, parties close the channel by recording the latest channel state on-chain. Different payment channel proposals use different techniques to replace the channel state. In the following, we will explain these techniques [16].

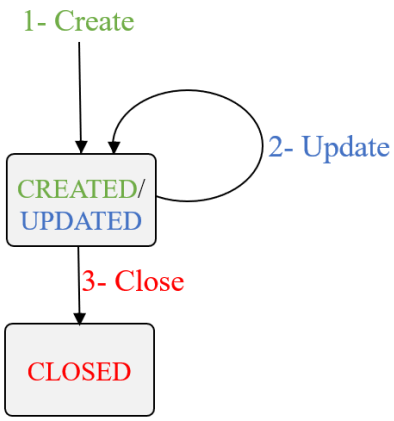

Figure 2.1: Lifetime of a payment channel.

- *Replace by Incentive (RbI).* This mechanism was proposed by Hearn and Spilman [26] and led to the first payment channel. This channel is funded by one of the channel parties who is the payer in the channel. Then, for each channel update, a partially signed transaction is given from the payer to the payee. The channel can be closed before a predetermined deadline by the payee who is incentivised to do that using the latest channel state because it gives the payee the highest deserved value. This type of channel has two limitations: (1) The lifetime of the channel must be determined at the time the channel is created and cannot be extended, and (2) The channel is unidirectional meaning that payment can only be done in one direction.

- *Replace by Time-Lock (RbT).* This mechanism allows both channel parties to pay each other [4]. Each agreed-on state in such a channel has an absolute time-lock whose value decreases with each channel update. Thus, the latest state can be published on the blockchain earlier than all other states. But due to its decrementing time-lock, the number of channel updates is limited.

- *Replace by Revocation (RbR).* If upon authorising a new state, the previous state is revoked, the payment channel is categorised as an RbR type. To revoke the previous state, parties exchange some revocation secrets that can be used for fraud. In other words, if a party tries to close the channel with a revoked state, her counterparty uses the corresponding revocation secret and publishes a transaction, called *revocation* transaction and penalises the dishonest party by taking all her funds in the channel. This idea was first proposed by Poon and Dryja in [15] for Bitcoin and has been actively used in Lightning Network with a current network capacity of around 5,000 BTC (as of November 2022).

  Although elegantly designed, the Lightning channel has some shortcomings. Firstly, since all the revocation secrets must be stored by channel parties, their storage requirements increase with each channel update. Secondly, each party has its own version of the channel state. This *state duplication* is required to distinguish a dishonest party from her counterparty. To add an application (e.g. *Virtual channel* [27]) on top of the channel, parties have to split their channel into sub-channels. The state of each sub-channel is duplicated and it must propagate on both duplicates of the parent channel. This causes the number of transactions to exponentially rise with the number of applications built on top of each other [14].

  Recently, Aumayr et al. [14] proposed a new design called *Generalized channel* which uses *adaptor signatures* to distinguish the publisher of a revoked state from her counterparty, i.e. once the revoked state is published, a *publishing secret* is revealed that can be used to penalise the dishonest party. In this way, state duplication is avoided. Towards a different direction, Aumayr et al. [28] proposed a channel, called "sleepy channel", that allows channel parties to go offline for prolonged periods.

- *Replace by Version (RbV).* Another method for replacing an old state is the usage of a monotonic counter that represents the version of each state. Channel parties store the channel state with the highest version and use it upon fraud. This idea has been used in several proposals on Turing complete blockchains (e.g. Ethereum) [29, 30, 17]. To extend this idea to Bitcoin, Decker et al. [1] introduced `ANYPREVOUT` as a new signature type for Bitcoin whose deployment requires a soft fork in the Bitcoin protocol [31]. This new signature type allows Bitcoin to support *floating*

transactions, i.e. transactions that can spend any Unspent Transaction Output (UTXO) with matching scripts. The proposal eltoo [1] uses this idea to deploy the concept of version.

Channels of type RbI and RbT are closed once the latest state is broadcast on-chain. However, for RbR and RbV channels, once a channel state is published by a channel party, a period, called the *dispute period*, starts. If the published state is invalid, the honest party proves its invalidity within the dispute period. For RbR channels, the invalidity of the published state is proved by publishing another transaction called the *revocation transaction* which can only be done by the honest party who knows the value of the corresponding revocation secret. For RbV channels, invalidity proof is done by broadcasting a state with a higher version.

Since the length of the dispute period is limited, RbR and RbV channels are based on the assumption that channel parties are always online. However, since this strong requirement cannot be met by most ordinary users, this task is delegated to a third-party service provider called the *watchtower*.

2.3 Watchtower

Monitor [21] is a watchtower scheme for Lightning Network. In this scheme, the hiring party provides the watchtower with IDs of revoked transactions as well as their corresponding revocation transactions. Then, the watchtower is supposed to look for a transaction with the matching ID on the blockchain. When a match is found, the watchtower can immediately publish its corresponding revocation transaction. To improve the efficiency, just some parameters such as addresses, signatures, and other metadata which are required for constructing the revocation transaction can be given to the watchtower [21]. Then, a full revocation transaction can be built by the watchtower if it is necessary.

To improve privacy, the hiring channel party encrypts the revocation transaction (or its corresponding parameters) using the second half of the revoked transaction ID and provides the watchtower with the first half of the ID along with the encrypted version of the revocation transaction. Then, the watchtower monitors the blockchain looking for every transaction whose ID matches the IDs received from the hiring party. If the watchtower finds a match, it decrypts its corresponding encrypted transaction using the second half of the found transaction ID and if the result is a valid transaction, transmits it on the blockchain. To reward the watchtower, the hiring party pays the watchtower through the revocation transactions, meaning that the watchtower is paid given that a revoked transaction is broadcast and its corresponding revocation transaction is published by the watchtower. To reduce the storage costs and hence operational costs of

the watchtower, Khabbazian et al. [22] proposed Outpost which proposes storing each revocation transaction as part of its corresponding channel state. Therefore, the watchtower just needs to watch for specific revoked transaction IDs on the blockchain. Then, the watchtower extracts the signed revocation transaction directly from the published revoked channel state that has appeared on the blockchain.

One issue with Monitor and Outpost is that the watchtower's client cannot completely trust that the watchtower remains online or does not collude with their counterparties. To address this concern specifically for Turing complete blockchains, DCWC and for Bitcoins, DCWC* [32] propose the implementation of a network of watchtowers that are financially motivated to faithfully cooperate with each other. This incentivised cooperation increases the probability that at least one honest watchtower prevents the channel from getting finalised with an old state. However, even for these schemes, channel parties cannot still be completely certain that they do not lose any money in their channels. The reason is that watchtowers are unaccountable or unfair to their clients. In other words, watchtowers are not penalised if they crash or deliberately do not act upon malicious behaviour. To encourage watchtowers to well-behave, [33] proposes a reputation system that works based on a cryptographic hash-based proof-of-work algorithm [34], called hashcash. This algorithm was originally designed to mitigate the email spamming issue.

Fail-safe [35] is a watchtower scheme for Turing complete blockchains and tries to resolve the issue of watchtower failure. A Fail-safe watchtower verifies all the off-chain channel updates and stores the latest state. Whenever a channel party, let's say A, attempts to close the channel, the watchtower can immediately reject or validate the committed state and receives its reward. However, if for any reason the watchtower is offline, a long timeout is triggered. Party B is supposed to get online before this long timeout is expired. Then, if the published state is an old one, B has this opportunity to prove the invalidity of the published state.

Another issue with Monitor, DCWC and DCWC* is that the watchtower is paid only if it observes a revoked channel state on the blockchain and broadcasts its corresponding revocation transaction. However, given that both parties of a given channel act honestly, the honest watchtower does not receive any reward although it has consumed some resources to remain online and monitor that channel. Such a reward mechanism is unfair to the watchtower and might disincentivise entities to run such services.

One simple solution to this issue (lack of fairness with respect to the watchtower), proposed by Outpost [22], is unconditional payment to the watchtower upon each channel update. However, this reward mechanism somehow exacerbates the consequences of the first mentioned problem (lack of fairness with respect to the hiring party) as it is probable for a channel party not only to lose some funds in the channel but also to pay

an unconditional reward to one or more watchtowers which actually did not prevent from fraudulent channel closure.

Therefore, having a watchtower scheme with a fair reward mechanism that also guarantees the safety of its channel parties' funds in the channel is desired. This is the main motivation behind the design of PISA which is a watchtower scheme proposed by Mccorry et al. [24]. The watchtower's client in PISA receives a signed receipt from the watchtower using which they can prove the watchtower's wrongdoing. In such situations, the large security deposit of the watchtower is forfeited. The watchtower is also paid per channel update.

However, PISA cannot be used for more limited scripting languages, and hence cannot be utilised for Bitcoin payment channels. Avarikioti et al. [23] proposed Cerberus which is a fair watchtower scheme for Bitcoin. In particular, the watchtower in Cerberus locks some collateral per channel that is taken by the channel party given that the watchtower is unresponsive upon fraud. However, it is not privacy-preserving as the watchtower learns how channel funds are redistributed amongst channel parties at each channel update.

Furthermore, there are two other schemes whose security assumptions are different from the ones mentioned earlier. Unlike other watchtower schemes, TEE Guard [36] relies on features of Trusted Execution Environments to build watchtowers. TEE Guard can be deployed for Lightning Network and its storage costs are constant per channel. Towards a different direction, Brick [37] is actually a state channel construction in which the dispute period is replaced with a committee of $n = 3f + 1$ members with at most f byzantine members where signatures of $t = 2f + 1$ members of the committee are required for each channel update. Each committee member in Brick locks some funds as collateral and cheating committee members are penalised by losing their collateral. Also, unlike other watchtower schemes, if one of the channel parties wants to unilaterally close the channel, the involvement of $t = 2f + 1$ committee members is required. This mitigates the synchrony requirement of other schemes which work based on a dispute period. In other words, watchtowers are no longer necessary in Brick.

2.4 Channel Synchronisation

To establish a payment channel, parties must lock some funds in the channel. If a user wants to perform transactions with many other users, a naive idea would be to establish payment channels with all of them. This idea clearly does not scale well. Thus, in practice payment channels are linked to form a Payment Channel Network (PCN) where each payment can be routed via multiple intermediaries [15]. Thus, if Alice wants to

send some funds to David whom does not share any channels, Alice finds a path to David where intermediate nodes (Bob and then Charlie) relay the payment from Alice to David. As an alternative, a Payment Channel Hub (PCH) [38, 39, 29, 30, 40] deploys a star topology where users can pay each other via a single intermediary (called the *tumbler*).

Hashed Time-Lock Contracts (HTLC) [41], is the main method to synchronise payments in different channels in PCN or PCH deployments. Now the most prominent PCNs, i.e. Lightning Network [15] on Bitcoin [2] and Raiden network [42] on Ethereum [43] are working based on HTLC. HTLC in a transaction output locks x coins in a contract with two parameters: a timeout t and a hash value $y := \mathscr{H}(R)$ where $\mathscr{H}$ is a collision-resistant hash function and R is a randomly selected value. The HTLC contract's condition can be fulfilled as follows: (1) If Bob provides a pre-image R^* with $y = \mathscr{H}(R^*)$ before time t, then Alice pays Bob x coins, (2) Otherwise, Alice is refunded with x coins after time t. Now assume that Alice wants to pay David via Bob and Charlie who agree to forward this payment in exchange for f coins as the fee. Then, Alice conditionally pays Bob $x + 2f$ coins with $3t$ and y as HTLC parameters. Similarly, Bob conditionally pays Charlie $x + f$ coins with $2t$ and y as HTLC parameters. Finally, Charlie conditionally pays David x coins with t and y as HTLC parameters. Now if David provides R before time t, he is paid x coins by Charlie. Having the value of R, Charlie and then Bob are also paid $x + f$ and $x + 2f$ coins by Bob and Alice, respectively. If R is not provided by David, then Charlie, Bob and finally Alice get back their coins after time t, $2t$ and $3t$, respectively.

The timeout value in HTLC mechanism is linear to the length of the path. Sprites [17] focuses on making the timeout value independent of the path length. However, it can only be deployed in Turing complete blockchains such as Ethereum. AMCU [44] resolves the same issue but in a way that is deployable on Bitcoin. Additionally, it extends the functionality by enabling concurrent payments in channels that are not necessarily along a path from the sender to the receiver. This could be useful for some applications such as crowdfunding. Payment Trees [45] presents an attack against AMCU and also provides a secure solution similar to Sprites but compatible to Bitcoin.

HTLC mechanism also suffers from two other problems: (1) The hash value y can be used by an adversary to compromise privacy by finding out who is paying to whom and (2) Malicious intermediaries might steal the fees from honest intermediaries in the same payment path [46]. [47] proposes a Multi-hop Hash Time-Lock Contract (MHTLC) protocol to improve HTLC privacy. However, its proposed protocol is quite expensive in terms of computation and communication. [48] proposes a less complex solution, called Chameleon Hash Time-Lock Contract (CHTLC) to solve the same privacy problem. However, it is incompatible with Bitcoin. AMHL [46] resolves both issues by replacing the HTLC contract with a novel cryptographic lock. However, since this novel idea is

incompatible with some important digital signatures such as BLS [20], [49] presents a new primitive called *lockable signature* which is compatible with any digital signature algorithm.

HTLC, MHTLC, CHTLC, AMHL and lockable signature follow a 2-phase-commit paradigm where in the first phase, HTLC contracts are set up one by one from the payer to the payee (i.e. from Alice to David in the previously mentioned scenario). Then in the second scenario, locks are released by passing the pre-image value (i.e. R) from the payee to the payer via intermediates before HTLC time-locks are expired. Blitz [50] provides a 1-phase protocol with provable security for multi-hub payments.

In all previously introduced solutions, all intermediate nodes along the path between the payer and the payee should be actively involved in each and every single payment. This makes the whole PCN network less reliable as offline users cannot contribute to payments. Also, this active involvement adds to each intermediary's service fee. To mitigate this issue, [29] provides a novel solution for Ethereum called virtual channels. A virtual channel is like a bridge between two users who do not share a direct payment channel. Intermediaries become involved in the process of virtual channel creation but once the virtual channel is created, end users can perform arbitrarily many off-chain transactions without requiring any involvement by intermediaries. Aumayr et al. [51] also extended this idea to a Bitcoin-compatible virtual channel.

Chapter 3

Preliminaries and Notations

3.1 Preliminaries

This section introduces the different cryptographic primitives used in this thesis. For all the cryptographic primitives defined in this section, κ is the security parameter. Also, a negligible function is defined as below.

Definition 3.1. *A function $\nu : \mathbb{N} \to \mathbb{R}$ is negligible in κ if for every $n \in \mathbb{N}$, there exists $n_0 \in \mathbb{N}$ such that for every $\kappa \geq n_0$, $|\nu(\kappa)| \leq 1/\kappa^n$ holds.*

3.1.1 Digital Signature

A digital signature scheme Π includes three algorithms as following:

- **Key Generation.** $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$ on input 1^κ, outputs the public/private key pair (pk, sk).
- **Signing.** $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$ on inputs the private key sk and a message $m \in \{0,1\}^*$ outputs the signature σ.
- **Verification.** $b \leftarrow \mathsf{Vrfy}_{pk}(m;\sigma)$ takes the public key pk, a message m and a signature σ as input and outputs 1 if σ is a valid signature on message m created with the private key corresponding to pk. Otherwise, it outputs 0.

Correctness of a digital signature guarantees that for any honestly generated signature σ on the message m w.r.t. a public key pk, $\mathsf{Vrfy}_{pk}(m;\sigma)$ outputs 1. In this work, we assume that the utilised signature schemes are existentially unforgeable under an adaptive chosen-message attack. It guarantees that the probability that an adversary who has access to a signing oracle outputs a valid signature on any new message is negligible

in κ. We call such signature schemes secure. ECDSA [52] is a secure signature scheme that is currently being used in Bitcoin. Schnorr signature [53] is another important secure signature scheme that has been proposed to be introduced in Bitcoin due to its key aggregation and signature aggregation properties.

3.1.2 Hard relation

A relation $\mathscr{R}$ with statement/witness pairs (Y, y) is called a hard relation if (1) There exists a polynomial time generating algorithm $(Y, y) \leftarrow \mathsf{GenR}(1^\kappa)$ that on input 1^κ outputs a statement/witness pair $(Y, y) \in \mathscr{R}$; (2) The relation between Y and y can be verified in polynomial time, and (3) For any polynomial-time adversary $\mathscr{A}$, the probability that $\mathscr{A}$ on input Y outputs y is negligible. We also let $L_{\mathscr{R}} := \{Y \mid \exists y \; s.t. \; (Y, y) \in \mathscr{R}\}$. Statement/witness pairs of $\mathscr{R}$ can be public/private key of a signature scheme generated by Gen algorithm.

3.1.3 Adaptor Signature

Adaptor signatures appeared first in [14]. Adaptor signature is used in the Generalized channel to tie together the authorisation of a commit transaction and the leakage of a secret value. In what follows, we recall how an adaptor signature works. Given a hard relation $\mathscr{R}$ and a signature scheme Π, an adaptor signature protocol Ξ includes four algorithms as follows:

- **Pre-Signing.** $\tilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$ is a probabilistic polynomial time (PPT) algorithm that on input a private key sk, message $m \in \{0,1\}^*$ and statement $Y \in L_{\mathscr{R}}$, outputs a pre-signature $\tilde{\sigma}$.
- **Pre-Verification.** $b \leftarrow \mathsf{pVrfy}_{pk}(m, Y; \tilde{\sigma})$ is a deterministic polynomial time (DPT) algorithm that on input a public key pk, message $m \in \{0,1\}^*$, statement $Y \in L_{\mathscr{R}}$ and pre-signature $\tilde{\sigma}$, outputs a bit b.
- **Adaptation.** $\sigma \leftarrow \mathsf{Adapt}(\tilde{\sigma}, y)$ is a DPT algorithm that on input a pre-signature $\tilde{\sigma}$ and witness y, outputs a signature σ.
- **Extraction,** $\mathsf{Ext}(\sigma, \tilde{\sigma}, Y)$ is a DPT algorithm that on input a signature σ, pre-signature $\tilde{\sigma}$, and statement $Y \in L_{\mathscr{R}}$, outputs $\perp$ or a witness y such that $(Y, y) \in \mathscr{R}$.

Correctness of an adaptor signature guarantees that for an honestly generated pre-signature $\tilde{\sigma}$ on the message m w.r.t. a statement $Y \in L_{\mathscr{R}}$, we have $\mathsf{pVrfy}_{pk}(m, Y; \tilde{\sigma}) = 1$.

Furthermore, when $\tilde{\sigma}$ is adapted to the signature σ, we have $\mathsf{Vrfy}_{pk}(m;\sigma) = 1$ and $\mathsf{Ext}(\sigma, \tilde{\sigma}, Y)$ outputs y such that $(Y, y) \in \mathscr{R}$.

An adaptor signature scheme is secure if it is existentially unforgeable under chosen message attack (aEUF−CMA security), pre-signature adaptable and witness extractable. The aEUF−CMA security guarantees that it is of negligible probability that any PPT adversary who has access to signing and pre-signing oracles outputs a valid signature for any arbitrary new message m even given a valid pre-signature and its corresponding Y on m. Pre-signature adaptability guarantees that every pre-signature (possibly generated maliciously) w.r.t. Y can adapt to a valid signature using the witness y with $(Y, y) \in \mathscr{R}$. Witness extractability guarantees that it is of negligible probability that any PPT adversary who has access to signing and pre-signing oracles outputs a valid signature and a statement Y for any new message m such that the valid signature does not reveal a witness for Y even given a valid pre-signature on m w.r.t. Y. The ECDSA-based and Schnorr-based adaptor signature schemes were constructed and analysed in [14].

3.2 Notations

Throughout this work, we define different attribute tuples. Let U be a tuple of multiple attributes including the attribute attr. To refer to this attribute, we use U.attr. Our focus in this work is on Bitcoin or any other blockchains with *Unspent Transaction Output* (UTXO) model. In this model, units of value–which we call coins–are held in *outputs*. An output θ is a tuple of two attributes, $\theta = (\mathsf{cash}, \varphi)$, where θ.cash denotes the number of coins held in this output and $\theta.\varphi$ denotes the condition that needs to be fulfilled to spend the output θ. The condition $\theta.\varphi$ is encoded using any script supported by the underlying blockchain. If the condition $\theta.\varphi$ contains a user P's public key, we say that P controls or owns the output θ. If satisfying a condition requires authorisations by multiple parties, such a condition contains public keys of all the involved parties separated by $\wedge$ operation(s). Relative time-lock of T rounds in an output condition is denoted by T^+ and means the output cannot be spent unless at least T rounds passed since the output was recorded on the blockchain. A condition might also have several sub-conditions, one of which must be satisfied to spend the output. Different sub-conditions of output are separated by $\vee$ operation(s).

A transaction changes ownership of coins, meaning it takes a list of existing outputs and transfers their coins to a list of new outputs. To distinguish between these two lists, we refer to the list of existing outputs as *inputs*. A transaction TX is formally defined as the tuple (txid, Input, nLT, Output, Witness). The identifier TX.txid $\in \{0,1\}^*$ is computed as $\mathsf{TX.txid} := \mathscr{H}([\mathsf{TX}])$, where [TX] is called the *body* of the transaction defined as $[\mathsf{TX}] := (\mathsf{TX.Input}, \mathsf{TX.nLT}, \mathsf{TX.Output})$ and $\mathscr{H}$ is a hash function, which is modelled as a random

oracle. The attribute TX.nLT denotes the value of the parameter *nLockTime*, where TX is invalid unless its *nLockTime* is in the past. The attribute TX.Input is a list of identifiers for all inputs of TX. The attribute TX.Output is a list of new outputs. The attribute $\mathsf{TX.Witness} = (\mathsf{W}_1, \dots, \mathsf{W}_m)$ is a list of tuples where its i^{th} tuple authorises spending the output that is taken as the i^{th} input of TX. The tuple $\mathsf{W}_i = (\eta, \zeta)$ of the witness Tx.Witness contains two attributes where $\mathsf{W}_i.\zeta$ denotes the data, e.g. the signature(s), that is (are) required to meet the $\mathsf{W}_i.\eta^{\text{th}}$ sub-condition of the output that is taken as the i^{th} input of TX. The signature and pre-signature of party P on TX for $\mathsf{TX.Witness.W}_j.\zeta$ is denoted by $\sigma_{\mathsf{TX}}^{P,j}$ and $\tilde{\sigma}_{\mathsf{TX}}^{P,j}$, respectively, where j can be removed if TX has one input. The i^{th} entry of a list L is denoted by $L[i]$ with $i > 0$.

We use charts to illustrate transaction flows. As Fig. 3.1 shows, double-edge and single-edge rectangles represent published and unpublished transactions, respectively. Also, dotted rectangles represent transactions that are still unprepared to be propagated in the blockchain network. In other words, dotted rectangles denote transactions that lack some required elements (e.g. some signatures in the witness). Considering that TX contains two inputs, each with a value of a and b respectively, then since the output of TX with the value of $a + b$ has two sub-conditions, it is denoted by a diamond shape with two arrows. One of the sub-conditions can be fulfilled by both A and B and is relatively time-locked by T rounds. Another sub-condition can be fulfilled by C where rather than the public key pk_C, the user identity, i.e. C, has been used in the chart. The *nLockTime* parameter for TX and TX′ is 0, so it is not shown inside these transactions.

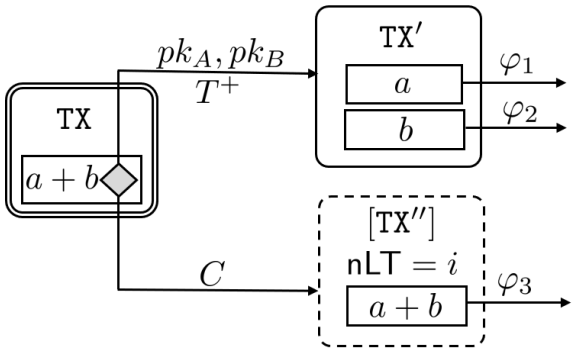

Figure 3.1: A sample transaction flow.

Table 3.1 summarises the mentioned notations.

Table 3.1: Summary of notations

Notation	Description
TX	Transaction $\mathsf{TX} = (\mathsf{txid}, \mathsf{Input}, \mathsf{nLT}, \mathsf{Output}, \mathsf{Witness})$
$[\mathsf{TX}]$	Tuple $(\mathsf{TX.Input}, \mathsf{TX.nLT}, \mathsf{TX.Output})$
$\mathsf{TX.txid}$	Identifier of the transaction TX
$\mathsf{TX.nLT}$	Parameter *nLockTime* of the transaction TX
$\mathsf{TX.Input}$	List of identifiers for all inputs of TX
$\mathsf{TX.Output}$	List of new outputs for TX
$\mathsf{TX.Witness}$	List of witnesses for TX
$\theta = (\mathsf{cash}, \varphi)$	Output with monetary value cash and condition φ
$W = (\eta, \zeta)$	Witness that meets η^{th} sub-condition of an output using data ζ
$\sigma_{\mathsf{TX}}^{P,j}$ (or $\tilde{\sigma}_{\mathsf{TX}}^{P,j}$)	Signature (or pre-signature) of P for j^{th} input of TX
T^{+}	The relative time-lock of T rounds
$L[i]$	i^{th} entry of a list L with $i \geq 1$

Chapter 4

Formal Treatment of Watchtower

4.1 Introduction

In this chapter, we formally define a payment channel and a watchtower and then work on our first research question, RQ1, which aims to formally define different properties of a watchtower scheme and analyse why existing watchtower schemes that fail to meet all the required properties. Monitor [15], DCWC[1] and DCWC* [32], Outpost [22] , Cerberus [23], PISA [24] and Fail-safe [35] are the watchtower schemes we will analyse in this chapter.

We construct some parts of this chapter based on our published paper, "FPPW: A Fair and Privacy Preserving Watchtower For Bitcoin" [54].

4.2 Formalisation of Payment Channel and Watchtower

A payment channel is defined as follows:

Definition 4.1 (Payment Channel)**.** *A payment channel γ of blockchain $\mathbb{B}$ is a state machine run by two participants. Let S_i with $i \geq 0$ be the i^{th} channel state. γ consists of three phases, namely* create, update *and* close, *as follows:*

- Channel create (state S_0)*: Channel participants commit to the initial state S_0 and record it on $\mathbb{B}$.*
- Channel update (from S_i to S_{i+1} for $i \in [0, n)$)*: Channel participants update the channel state from S_i to S_{i+1} and commit to S_{i+1}.*

[1] Disclose Cascade Watch Commit

- Channel close (state S_n): *Any channel participants can close the channel by recording the latest state S_n on $\mathbb{B}$.*

Remark 4.1. *Definition 4.1 is a generic one for a bi-directional payment channel. Moreover, to analyse the payment channel security in the universal composability framework [55], some ideal functionalities of payment channels can also be found in the literature [14, 28]. We will also present an ideal functionality for our designed payment channel, Daric, in Chapter 7.*

The capacity of a payment channel γ is defined below.

Definition 4.2 (Channel Capacity)**.** *For a payment channel γ with two participants A and B, let $x_{P,i}$ with $P \in \{A, B\}$ denote the balance of P at the channel state S_i. Then, the channel capacity is denoted by $X_{A,B}$ and equals $x_{A,0} + x_{B,0}$.*

Definition 4.3 (Payment Channel Security)**.** *A payment channel γ of blockchain $\mathbb{B}$ with two channel participants A and B is μ-secure if and only if given that γ with latest state of S_n is closed by recording the state S_i with $i < n$ on $\mathbb{B}$, an honest party $P \in \{A, B\}$ can claim at least $x_{P,n}$ by recording some evidence proving invalidity of S_i on $\mathbb{B}$ within μ blocks.*

The above definition states that when a μ-secure payment channel is closed using a transaction that corresponds with an old state, the honest party has still μ blocks time to get refunded or equivalently to invalidate the published channel state. After μ blocks, the honest party's funds are not guaranteed anymore. To invalidate the published channel state, the honest party must record some evidence on $\mathbb{B}$ which proves the published state is not the latest channel state that parties have committed to. For Replace by Revocation or RbR payment channels (e.g. [14, 15]), the evidence is knowledge of the revocation secret corresponding with the published channel state. This knowledge is practically proved by publishing a revocation transaction that gives all the funds in the channel to the honest party. For Replace by Version or RbV type channels, the evidence is a state with higher version [1, 17].

One or both parties of a payment channel can outsource the task of publishing the evidence to a third-party called watchtower defined as follows.

Definition 4.4 (Watchtower)**.** *For a μ-secure payment channel γ of $\mathbb{B}$ with two participants A and B, a watching contract $C = (W, \gamma, P, S_s, \mathscr{C})$ with $\mathscr{C} = \{\mathscr{C}_S, \mathscr{C}_U, \mathscr{C}_T\}$ is a contract between a watchtower W and a party $P \in \{A, B\}$ in which W commits to follow* start, update, termination *and* alarm, *as follows:*

- Watchtower start (state S_s, condition $\mathscr{C}_S$): *Watchtower starts when the condition $\mathscr{C}_S$ is met and P provides the evidence e to W. Let e invalidate a set of channel states $\mathcal{S}$.*

The start state S_s is defined by

$$S_s := \begin{cases} S_{\max\{i : S_i \in \mathcal{S}\}+1} & \mathcal{S} \neq \emptyset, \\ S_0 & \mathcal{S} = \emptyset. \end{cases}$$

- Watchtower update (from S_k to S_{k+1}, condition $\mathscr{C}_U$)*: With each channel update from S_k to S_{k+1}, watchtower updates if the condition $\mathscr{C}_U$ is met and party P provides W with some evidence proving the invalidity of S_k.*

- Watchtower terminate (condition $\mathscr{C}_T$)*: Watchtower terminates once the latest state S_n is recorded on $\mathbb{B}$ or once the condition $\mathscr{C}_T$ is met.*

- Watchtower react*: Let γ with latest state of S_n be closed by recording the state S_i with $i \neq n$ on $\mathbb{B}$. If this event occurs before the condition $\mathscr{C}_T$ is met, then watchtower W records some evidence proving the invalidity of S_i on $\mathbb{B}$ within μ blocks.*

The condition set $\mathscr{C}$ could be a watchtower scheme specific condition, possibly relying on different parameters. In the following, we will discuss the condition set for the existing watchtower schemes.

For DCWC, DCWC* [32] and monitor [15], $\mathscr{C}_S$ and $\mathscr{C}_U$ are $\emptyset$. However, these schemes have not discussed their watchtower termination conditions although channel closure by the hiring party P is definitely included in $\mathscr{C}_T$.

The conditions $\mathscr{C}_S$ and $\mathscr{C}_U$ for Outpost are payments to the watchtower. However, condition $\mathscr{C}_T$ is somehow complicated. An outpost watchtower stores a pre-defined number of evidence per user and deletes older items based on a FIFO order. Thus, when a user has several channels, it is possible that while a channel is still open, its older states are not monitored by the watchtower anymore. Thus, $\mathscr{C}_T$ for Outpost is met once the channel is closed by the hiring party P or with an old state whose corresponding evidence has been removed from the watchtower's evidence list.

The watchtower in Cerberus [23] locks some collateral at the watchtower start phase. This collateral is taken by the hiring party given that the watchtower does not follow the watchtower reaction rules. So, $\mathscr{C}_S$ is locking some collateral by the watchtower. The watchtower is rewarded upon each watchtower update. Also, as part of each watchtower update, the watchtower must provide the channel parties with signatures on two *penalty* transactions that are used by the cheated hiring party to penalise the unresponsive watchtower for not following the watchtower react rules. So, $\mathscr{C}_U$ is payment to the watchtower and signing the corresponding penalty transactions by the watchtower. $\mathscr{C}_T$ is met when the watchtower's collateral is redeemed or the channel is closed by the hiring party P.

Table 4.1: Summary of the Condition $\mathscr{C}_S$ for Different Watchtower Schemes.

	$\mathscr{C}_S$
Monitor [21]	$\emptyset$
DCWC [32]	$\emptyset$
Outpost [22]	$\emptyset$
Cerberus [23]	collateral
PISA [24]	custodian contract with security deposit, payment, signed receipt
Fail-safe [35]	tower contract, payment, signed receipt

Table 4.2: Summary of the Condition $\mathscr{C}_U$ for Different Watchtower Schemes.

	$\mathscr{C}_U$
Monitor [21]	$\emptyset$
DCWC [32]	$\emptyset$
Outpost [22]	payment
Cerberus [23]	payment, signatures of penalty transactions
PISA [24]	payment, signed receipt
Fail-safe [35]	payment, signed receipt

The watchtower in PISA first locks a large deposit in a *smart contract*[2] called *custodian contract*. This large deposit is forfeited if the watchtower does not follow the watchtower reaction rules. To hire the watchtower, the watchtower is paid by the hiring party in exchange for a signed receipt that can be used to prove the watchtower's wrongdoing before an agreed expiry time. So, $\mathscr{C}_S$ is setting up the custodian contract and locking a large security deposit in this contract by the watchtower, payment to the watchtower and providing the corresponding signed receipt to the hiring party. $\mathscr{C}_U$ is a new payment to the watchtower and providing the corresponding new signed receipt to the hiring party. $\mathscr{C}_T$ is met at an agreed expiry time.

For Fail-safe, $\mathscr{C}_S$ is having a smart contract called the *tower contract* set up by the watchtower on the blockchain, payment to the watchtower, and providing the corresponding signed receipt to the hiring party. $\mathscr{C}_U$ is a new payment to the watchtower and providing the corresponding new signed receipt to the hiring party. The condition $\mathscr{C}_T$ is $\emptyset$, meaning that the watchtower has to monitor the channel and be responsive as long as the channel is open. Otherwise, the watchtower might lose its reward.

Tables 4.1, 4.2 and 4.3 respectively summarise the conditions $\mathscr{C}_S$, $\mathscr{C}_U$ and $\mathscr{C}_T$ for different watchtower schemes.

[2] A self-executing, programmable code that runs on a blockchain network, enabling automated and trustless execution of predefined actions and agreements between parties.

Table 4.3: Summary of the Condition $\mathscr{C}_T$ for Different Watchtower Schemes.

	$\mathscr{C}_T$
Monitor [21]	at least channel closure by the hiring party P
DCWC [32]	at least channel closure by the hiring party P
Outpost [22]	(1) channel closure by the hiring party P or (2) channel closure using a transaction with removed evidence
Cerberus [23]	(1) redeemed collateral or (2) channel closure by the hiring party P
PISA [24]	expiry time
Fail-safe [35]	$\emptyset$

4.3 Watchtower Service Properties

In this section, we formally define different new properties of a watchtower service.

4.3.1 Agility

This section defines *agility* for a watchtower. The agility of a watchtower shows how flexibly a watching process starts and terminates.

Definition 4.5 (Agility). *A payment channel γ between two parties A and B with watchtower provides agility iff*

- *two distinct contracts $C_1 = (W_1, \gamma, P, S_s, \mathscr{C})$ and $C_2 = (W_2, \gamma, P, S_s, \mathscr{C})$ may be formed, where $P \in \{A, B\}$ and $W_1 \neq W_2$, and*
- *Any watching contract can start and terminate at any arbitrary state of the channel γ.*

According to the above definition, Cerberus and Fail-safe do not achieve agility as the watchtower in both schemes should be specified in the channel create phase. Moreover, watching for both schemes must start and terminate when the channel is created and closed, respectively. [23] discusses that if the watching contract for Cerberus terminates before channel closure, the hiring party, e.g. B, might continue using the channel given that he is always online. However, this statement is correct with this assumption that after the watchtower termination phase, the watchtower will not be involved in this channel any more. Since this assumption might be violated in the real world, B would be at risk of losing some funds in the channel. In more detail, once A publishes an old channel state on the blockchain, the dispute period starts and B must publish a revocation transaction within this period to take all the channel funds. However, if

the dishonest party A has colluded with the watchtower (whose collateral has already been redeemed), they can cooperatively claim the channel funds before the revocation transaction is published by B. So, even staying always online would not guarantee the safety of the honest party's funds in the channel. In other words, once the watchtower redeems its collateral, the channel for the hiring party transforms to a μ-Secure channel with $\mu = 0$. So, the hiring party must close the channel before the watchtower's collateral redemption (or equivalently before the time when the watchtower contract terminates). The watching contract for other schemes can start and terminate at any time with any arbitrary watchtower. So, Monitor, DCWC, DCWC*, Outpost and PISA achieve agility.

4.3.2 Privacy

One of the properties of payment channels is that off-chain transactions are not broadcast on the blockchain and hence only the channel parties know how the channel funds are redistributed between them. Thus, in the following, to evaluate how the usage of watchtowers can affect this benefit, we define a *privacy game* and then based on this privacy game we define *Weak Privacy Against Watchtower*.

Challenge. Let there exist two payment channels where the first one is between honest channel parties A and B and the second one is between honest channel parties A' and B' and both channels have the same number of channel updates n and the same channel setup, i.e. $x_{A,0} = x_{A',0}$, $x_{B,0} = x_{B',0}$, $x_{A,n} = x_{A',n}$ and $x_{B,n} = x_{B',n}$. Let $\mathbf{x}_{P,[i,j]}$ show the sequence of balance values of party P between i^{th} to j^{th} states of the payment channel that P is involved in. Assume that $\mathscr{A}$ is any passive PPT adversarial watchtower excluding A, B, A' and B' which watches these two channels. To challenge $\mathscr{A}$, the challenger selects a random bit b and gives the sequence $(\mathbf{x}_{P,[1,n-1]}, \mathbf{x}_{\bar{P},[1,n-1]})$ to $\mathscr{A}$ where $P = A$ and $\bar{P} = B$ if $b = 0$ and $P = A'$ and $\bar{P} = B'$ otherwise.

Output. The adversary $\mathscr{A}$ outputs a bit b' to guess that the received sequence belongs to the first or the second channel. The adversary wins the game if and only if $b = b'$.

Definition 4.6 (Weak Privacy Against Watchtower)**.** *A payment channel with a watchtower provides weak privacy against the watchtower if according to the privacy game* $| \Pr[b = b'] - 1/2 |$ *is negligible in* κ.

Remark 4.2. *For any multihop payment routed via the channel between A and B or the channel between A′ and B′, we assume that the passive adversary is not involved as a channel party in routing such payments.*

The privacy guarantee would be stronger, if in the defined privacy game, (1) the sequence $(\mathbf{x}_{P,[0,n]}, \mathbf{x}_{\bar{P},[0,n]})$ is given to the passive PPT adversarial watchtower and (2) the watchtower determines if the channel setup for two channels are the same or not. This strong privacy implies weak privacy because the channel setup in the weak privacy game is a special case of that in the strong privacy game. Moreover, the watchtower in the strong privacy game receives more input data in the challenge phase than the watchtower in the weak privacy game. Thus, the probability of her winning the strong privacy game would be at least equal to the corresponding probability in the weak privacy game.

Monitor, DCWC, DCWC* and Outpost achieve strong privacy against the watchtower because the watchtower receives at most the encrypted version of channel transactions and if the channel parties behave honestly, these encrypted transactions are not decrypted at all. PISA and Fail-safe achieve weak privacy against the watchtower because for these schemes, states are invisible to the watchtower and only state hash values are given to the watchtower and since a large random value is also used in the computation of the hash value, finding the pre-image by exhaustive search is also infeasible. However, The first and the last states of the channel are revealed. Cerberus does not achieve privacy against the watchtower as the watchtower learns the balances of both channel parties in the channel.

Another important privacy-related subject regarding a payment channel and its corresponding watchtower is the knowledge of a third party (i.e. any party other than the hiring party and the watchtower) regarding the hiring status of the watchtower for a given channel. For example in the channel γ with two parties A and B, such knowledge can be of importance to a channel party, let's say party A, given that he is malicious and is looking for an appropriate time (i.e. when party B is not using the watchtower with a significant probability) to broadcast an old state or to a third party (other than party A) who has some other channels with party B. Such information can help him to conduct a behavioural analysis on party B to find the best time to attack him on his other channels. This type of privacy is also defined below.

Definition 4.7 (Weak Watchtower Privacy against Third-Party)**.** *Let W be a watchtower. A challenger A establishes a payment channel γ with B, samples $b \in \{0,1\}$ randomly and hires (does not hire) W if $b = 1$ ($b = 0$, respectively). The watchtower W achieves weak privacy against third-party if for all PPT adversaries B, we have that*

$$\left|\Pr[1 \leftarrow B(\cdot)|b = 1] - \frac{1}{2}\right|$$

is negligible in κ.

Definition 4.8 (Strong Watchtower Privacy against Third-Party)**.** *Let W be a watchtower. A challenger A establishes a payment channel γ with B, samples $b \in \{0,1\}$ randomly and*

hires (does not hire) W if $b = 1$ ($b = 0$, respectively). The watchtower W achieves strong privacy against third-party if for all PPT adversaries $\mathscr{A}$ including B, we have that

$$\left|\Pr[1 \leftarrow \mathscr{A}^{BAW\text{-}view}(\cdot)|b = 1] - \frac{1}{2}\right|$$

is negligible in κ, where BAW-view is an oracle in which $\mathscr{A}$ has access to what B sees on γ and sees all public keys of A and W and all transactions related to them on $\mathbb{B}$.

The strong watchtower privacy against third-party implies the weak watchtower privacy against third-party. The reason is that the PPT adversary in the former also includes B. Moreover, she accesses more information than the adversary in the weak privacy definition.

According to Definition 4.8, Monitor, DCWC and DCWC* provide strong privacy against third-party because there is not any transaction between the hiring party and the watchtower. Thus, even knowledge of all the public keys and transactions of the hiring party and the watchtower cannot help the third party to guess whether there is a watchtower contract between the hiring party and the watchtower.

However, PISA, Outpost and Cerberus do not provide strong privacy against third-party as for all these schemes payment to the watchtower is done through a one-way payment channel per update and hence $\mathscr{A}^{\text{BAW-view}}(\cdot)$ for these watchtowers can output the correct result with non-negligible probability. For instance, no transaction including public keys of A and W implies no one-way channel between A and W and hence no watching contract between them.

PISA, Outpost and Cerberus potentially achieve weak privacy against third-party given that any relationship between transactions in γ and watchtower-related transactions are invisible to B (as defined in Definition 4.7). To achieve that, public keys used in watchtower-related transactions must be independent of the ones used in γ. This condition can be simply met. Moreover, watchtower-related transactions must not change the distribution of different transaction types on the underlying blockchain. For example, if transactions with multi-signature outputs (and in particular 2-of-2 multi-signature class) are rarely exchanged between the users, usage of such transactions for watchtower purposes might be distinguishable from other transactions. In such a case, a straightforward option is aggregating public keys of the watchtower and the hiring party in watchtower-related transactions. Also, there must be some random differences between the time when the channel is created and when the watchtower-related transactions are published on the blockchain.

Fail-safe does not provide weak privacy against third-party as B knows the hiring status of W upon establishment of the channel γ.

4.3.3 Fairness and Coverage

In this section, we formalise the concepts of fairness and coverage in a watchtower scheme. Fairness with respect to the hiring party (watchtower) is actually a factor to evaluate the level of guarantee that the watchtower (hiring party) provides to the hiring party (watchtower) on its service (payment).

Definition 4.9 (Channel party α-Fairness)**.** *A payment channel with watchtower is α-party-fair, if the following holds for an honest channel party P:*

- *P can close the channel at any time and*
- *α is the largest real number such that regardless of the reward that P pays to the watchtower, P loses at most $(1-\alpha)\cdot x_P$ coins in the channel where x_P denotes balance of P in the latest channel state.*

Note that $0 \leq \alpha \leq 1$, where $\alpha = 1$ implies that the honest party P will not lose any funds in the channel and $\alpha = 0$ means that P might lose all of his funds. Since for Monitor, DCWC, Outpost and Fail-safe, the hiring party P might lose all his funds in the channel, these schemes are unfair w.r.t. the channel party (i.e. they are α-party-fair with $\alpha = 0$). The value of the watchtower's collateral for Cerberus is around the channel capacity. So the hiring party would not lose any funds in the channel. In other words, $\alpha = 1$ for Cerberus. Although for PISA α is adjustable, the hiring party must not logically accept a PISA watchtower with $\alpha < 1$. Otherwise, he might lose some funds in the channel. Thus, the value of α for PISA must be 1 in practice or equivalently the watchtower must lock some collateral per channel where the value of the collateral must be equal to the channel capacity. This collateral will be taken by the hiring party given that the watchtower is unresponsive upon fraud.

Definition 4.10 (Watchtower Fairness)**.** *A payment channel with a watchtower is watchtower-fair if the following holds for an honest watchtower $\mathscr{W}$:*

- *$\mathscr{W}$ is rewarded with some non-zero amounts of coins and*
- *given that $\mathscr{W}$ has locked some collateral as part of the watching service, it is of negligible probability that the honest watchtower cannot redeem all the collateral once watching terminates according to the watching agreement.*

Monitor [21] and DCWC [32] are called unfair w.r.t. the watchtower because, for these schemes, it is possible that the watchtower is not rewarded. In more detail, the watchtower in these schemes is rewarded if and only if (1) the counterparty of the hiring party P publishes an old channel state on the blockchain and (2) the watchtower succeeds in

broadcasting the corresponding evidence and wins the race against other watchtowers, which are simultaneously monitoring the same channel for P. On the other hand, due to the punishment mechanism of the Lightning Network, fraudulent channel closures and hence payments to the watchtower would be rare and this can dis-incentivise entities to run such services. PISA, outpost, Cerberus and Fail-safe are watchtower-fair as the watchtower in these schemes is rewarded upon each channel update and the watchtower in Cerberus can redeem its collateral at any time.

Next, we define β-coverage, which basically measures the capability of a watchtower (on a scale between 0 to 1) in watching all the existing payment channels on a fixed Blockchain.

Definition 4.11 (Coverage)**.** *For a blockchain $\mathbb{B}$ with N payment channels, a watchtower $\mathscr{W}$ provides β-coverage with $\beta := \frac{\mathscr{X}}{\mathscr{C}+\mathscr{X}}$, where $\mathscr{C}$ is the total collateral required by $\mathscr{W}$ to watch all payment channels for both channel parties and $\mathscr{X}$ is the total capacity of all channels.*

The parameter β can take any value in the interval $[0, 1]$. For Cerberus and PISA (with $\alpha = 1$), β equals $\frac{1}{3}$ because, for these schemes, collateral of the watchtower must be twice the channel capacity if the watchtower is going to be hired by both channel parties. Although PISA allows lower values of collateral, such values cannot provide channel party α-fairness with $\alpha = 1$ and hence cannot guarantee that the honest party does not lose any funds. The watchtower in other schemes does not need to lock any collateral per channel. So they provide β-coverage with $\beta := 1$.

The following Lemma shows a trade-off between watchtower fairness and coverage.

Lemma 4.1. *For an (α, R)-fair watchtower W with β-coverage, we have:*

1. *$\mathscr{X} \leq \beta \cdot MC$, where MC denotes the market cap of the used cryptocurrency and $\mathscr{X}$ is defined in Definition 4.11,*
2. *$\mathscr{C} \geq \alpha \cdot \mathscr{X}$, where $\mathscr{C}$ is defined in Definition 4.11 and*
3. *$\beta \leq \frac{1}{1+\alpha}$.*

Proof. We prove each item in the following:

1. Since $\mathscr{X} + \mathscr{C} \leq MC$ and W has β-coverage we have:

$$\mathscr{X} = \beta \cdot (\mathscr{C} + \mathscr{X}) \leq \beta \cdot MC.$$

2. For the channel γ_i with channel parties A_i and B_i, as defined in Definition 4.11, we have two watching contracts $C_i = (W, \gamma_i, A_i, S_i, \mathscr{C}_i)$ and $C_i' = (W, \gamma_i, B_i, S_{i'}, \mathscr{C}_i')$. Assume that γ_i is closed by a party, let's say party B, at state S_n by committing to state S_j with $j < n$ before the condition $\mathscr{C}_i$ being met. For the case $x_{A_i,n} = X_{A_i,B_i}$ and $x_{A_i,j} = 0$, honest party A is cheated out of its total funds which equals X_{A_i,B_i}. However, since W is (α, R)-fair, party A does not lose more than $(1-\alpha) \cdot x_{A_i,n} = (1-\alpha) \cdot X_{A_i,B_i}$ if he is honest. Thus, capital of W for this channel cannot be less than

$$X_{A_i,B_i} - (1-\alpha) \cdot X_{A_i,B_i} = \alpha \cdot X_{A_i,B_i}$$

Otherwise, party A loses more than $(1-\alpha) \cdot x_{A_i,n}$, which contradicts with definition of fairness. Therefore, the total capital of W must satisfy

$$\mathscr{C} \geq \sum_{i=1}^{N} \alpha \cdot X_{A_i,B_i} = \alpha \cdot \mathscr{X}. \tag{4.1}$$

3. Based on the definition of coverage and (4.1), we have:

$$\mathscr{X} = \beta \cdot (\mathscr{C} + \mathscr{X}) \geq \beta \cdot (\alpha \cdot \mathscr{X} + \mathscr{X})$$

Since $\mathscr{X} \neq 0$ and $\alpha \neq -1$, we have:

$$\beta \leq \frac{1}{1+\alpha}.$$

□

The above Lemma shows that although an increase in α for a watchtower promotes its fairness with regard to the hiring party, it raises the required capital of the watchtower and negatively affects its coverage. In other words, if all payment channel parties seek to achieve full guarantee by their watchtowers, in the best case at most half of the market cap of a given cryptocurrency can be used for transactions between channel parties and another half must be locked as collateral by their watchtowers.

The Table 4.4 summarises the comparison results for the existing watchtower schemes. The table illustrates that the current watchtower schemes face challenges in being deployed on Bitcoin, ensuring privacy, or providing channel-party 1-fairness.

4.4 Conclusion

In this chapter, we formalised payment channels and watchtowers. We also formally defined different properties of a watchtower scheme including agility, privacy, fairness

Table 4.4: Comparison of Different Watchtower Schemes.

	Bitcoin Support	Agility	Priv. ag. Watch.	Priv. ag. 3rd Party	Watch. Fairness	α	β
Monitor [21]	Yes	Yes	Strong	Strong	No	0	1
Outpost [22]	Yes	Yes	Strong	Weak	No	0	1
DCWC [32]	No	Yes	Strong	Strong	No	0	1
DCWC* [32]	Yes	Yes	Strong	Strong	No	0	1
Cerberus [23]	Yes	No	-	Weak	Yes	≈ 1	$\approx \frac{1}{3}$
PISA [24]	No	Yes	Weak	Weak	Yes	Adj.	$\frac{1}{1+2\alpha}$
Fail-safe [35]	No	No	Weak	-	Yes	0	1

and coverage and compared the existing watchtower schemes against these properties. Furthermore, we proved a trade-off between fairness and coverage of a watchtower. This trade-off shows although an offline channel party with a watchtower satisfying $\alpha < 1$ would be at the risk of losing some funds, the fully fair schemes (Cerberus and PISA with $\alpha = 1$) cannot achieve acceptable coverage. Moreover, upon examining existing watchtower schemes on Bitcoin, we observed that none of them can simultaneously achieve channel party 1-fairness and privacy against watchtowers.

Chapter 5

FPPW: a fair and privacy preserving Bitcoin watchtower

In this chapter, our focus is on addressing our second research question, RQ2. Our objective is to develop a watchtower scheme specifically tailored for Bitcoin, aiming to overcome the limitations of existing schemes. We concentrate on enhancing the fairness and privacy properties of the watchtower in this investigation. We construct this chapter based on the full version of our published paper, "FPPW: A Fair and Privacy Preserving Watchtower For Bitcoin" [54] (The full version paper is available at https://eprint.iacr.org/2021/117.pdf).

5.1 Introduction

Monitor [21] is the first watchtower scheme for Lightning Network, which mainly focuses on channel privacy against watchtower. However, Monitor has two main issues, both of which are related to fairness. Firstly, honest watchtowers might be rewarded for fraud (i.e. broadcast of an old state on the blockchain), which is unfair with respect to (w.r.t.) the watchtower. Secondly, honest parties cannot penalise the unresponsive watchtower, which is unfair towards an honest hiring party.

DCWC* [32] proposes the usage of a network of watchtowers that must cooperate to maximise their interest. This reduces the probability that the channel gets finalised with an old state. However, watchtowers might still crash or get unresponsive without being penalised by the hiring party. Also, the reward mechanism is still unfair w.r.t. the watchtower. Outpost [22] solves the issue of fairness towards the watchtower by paying her per channel update.

Table 5.1: Different Properties of the FPPW Scheme.

	Bitcoin Support	Agility	Priv. ag. Watch.	Priv. ag. 3rd Party	Watch. Fairness	α	β
Cerberus [23]	Yes	No	-	Weak	Yes	≈ 1	$\approx \frac{1}{3}$
PISA [24]	No	Yes	Weak	Weak	Yes	Adj.	$\frac{1}{1+2\alpha}$
FPPW	Yes	No	Weak	Weak	Yes	≈ 1	$\approx \frac{1}{2}$

Cerberus [23] and PISA [24] elegantly provide fairness w.r.t. the hiring party. However, PISA fails to be deployed in cryptocurrencies with limited script languages such as Bitcoin and Cerberus sacrificing the channel privacy against watchtower. In particular, the Cerberus watchtower learns the distribution of funds in the channel. Thus, the main motivation of this chapter is designing a watchtower scheme for Bitcoin that achieves both: (1) fairness w.r.t. both the hired watchtower and her hiring party and (2) channel privacy against the watchtower.

The contributions of this chapter are as follows:

- In Section 5.3, we present a new privacy-preserving payment channel with a watchtower scheme for Bitcoin called FPPW, which is fair w.r.t. all channel participants and allows the channel parties to go offline for a long period of time. To be more precise, FPPW is an extension of a new variant of the Generalized channel, called NVG, which is introduced in Section 5.3.2.1. Furthermore, in Section 5.5, we prove that our design achieves fairness w.r.t. all channel participants and unlike Cerberus, it provides weak privacy against the watchtower. We also show that the coverage of FPPW is $\frac{1}{2}$ which is better than that of Cerberus and PISA. Table 5.1 summarises the mentioned properties for FPPW and compares it with PISA and Cerberus.

- In Section 5.6, we propose a fee handling mechanism that allows the channel participants to determine the fee for different transactions at the time when fraud occurs. Furthermore, a proof-of-concept implementation of FPPW channels on Bitcoin is provided in Section 5.9.

5.3 FPPW Overview

5.3.1 System Model

Cryptographic primitives that have been used in FPPW (i.e. digital signature, hard relation and adaptor signature as defined in Section 3.2) are cryptographically secure. There is an authenticated and secure end-to-end communication channel between channel parties. The watchtower and channel parties are rational and might deviate from the protocol if it increases their profit. Also, each pair of participants might collude with each other if it raises the total profit of colluding participants. The watchtower is an always online service provider, but channel parties can go offline for a long period (approximately T rounds). Furthermore, the underlying blockchain contains a distributed ledger that achieves security [56]. When a valid transaction is propagated in the blockchain network, it is definitely included in the blockchain ledger immediately (i.e. the confirmation delay Δ is 1).

Remark 5.1. *FPPW channels can work with any confirmation delay. However, we assume that the confirmation delay is 1 to simplify the protocol and its analysis.*

5.3.2 Overview

A payment channel (as defined in Definition 4.1) contains a sequence of state updates between two parties where only its first and last states are recorded on the blockchain. The two channel parties process all the intermediate state updates off-chain. This eliminates the need to confirm every state update, i.e. every transaction, on the blockchain. However, as one may submit an intermediate state (which is already revoked by a later state) to the blockchain, the channel parties will need to get online frequently to monitor and punish such misbehaviours. Such a requirement may be impractical for some users. Thus, the watchtower is introduced as a third party to act on behalf of the channel parties.

FPPW is a fair and privacy-preserving watchtower service for generalised channels [14]. The watchtower in an FPPW channel obtains no data on intermediate state updates. To provide fairness towards the watchtower, the FPPW service rewards the watchtower for channel creation and per channel update. Furthermore, to achieve fairness w.r.t. channel parties, the watchtower must lock some collateral, which can be redeemed by the watchtower if the watchtower is responsive upon fraudulent channel closures. If the watchtower is dishonest and the channel is closed at an old state, protocol guarantees that the cheated party can penalise the watchtower by taking its collateral. Watchtower can terminate its employment at any time. Then, the channel parties can update the

channel on-chain and hire a new watchtower or continue using the channel. In the latter case, channel parties must get online frequently.

In the rest of this section, we will provide an overview of FPPW. To do that, we will present a simple payment channel called NVG and then, we add a watchtower service to NVG which is fair with respect to the hiring party. Finally, we make our solution fair to the watchtower by allowing the watchtower to terminate its service.

5.3.2.1 NVG: A New Variant of the Generalized Channel

Fig. 5.1 depicts a New Variant of the Generalized channel [14], called NVG channel. An NVG channel is created once channel parties publish a funding transaction on the blockchain and hence fund a 2-of-2 multi-signature output[1] on the ledger. The i^{th} channel state includes a commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ as well as a split transaction $\mathsf{TX}_{\mathsf{SP},i}$. The commit transaction sends the channel funds to a new joint account which is shared between the channel parties. The output of the commit transaction has two sub-conditions. The first sub-condition which is not time-locked, as we will explain later, is used for revocation purposes. The second sub-condition is relatively time-locked by t rounds with $t > \Delta$ and is met by the corresponding split transaction. Split transaction distributes the channel funds between the channel parties and hence represents the channel state.

The transaction $\mathsf{TX}_{\mathsf{CM},i}$ requires signatures of both parties A and B to be published. To generate $\sigma^{B}_{\mathsf{TX}_{\mathsf{CM},i}}$, party A generates a statement/witness pair $(Y_{A,i}, y_{A,i})$ and sends the statement $Y_{A,i}$ to B. Then, party B uses the pre-signing algorithm pSign of the adaptor signature and A's statement $Y_{A,i}$ to generate a pre-signature $\tilde{\sigma}^{B}_{\mathsf{TX}_{\mathsf{CM},i}}$ on $[\mathsf{TX}_{\mathsf{CM},i}]$ and sends the pre-signature to A. Thus, whenever it is necessary, A is able to use the adaptation algorithm adapt of the adaptor signature to transform the pre-signature to the signature $\sigma^{B}_{\mathsf{TX}_{\mathsf{CM},i}}$ and publish $\mathsf{TX}_{\mathsf{CM},i}$ on-chain. This also enables B to apply the extraction algorithm Ext on the published signature and its corresponding pre-signature to extract the witness value $y_{A,i}$. The witness value, as will be seen, allows the honest party to punish the dishonest channel party by claiming all the channel funds, given that the published commit transaction is already revoked.

As one may submit an intermediate state (which is already replaced by a later state) to the blockchain, the channel parties will need to punish such misbehaviours. Thus, upon channel update from state i to $i+1$, a revocation transaction $\mathsf{TX}_{\mathsf{RV},i}$ is created by parties. Unlike the split transaction, the revocation transaction can immediately spend the output of the corresponding commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ using its first sub-condition which

[1] An output that utilizes a script containing two public keys, requiring their corresponding signatures for spending the output.

does not contain any time-lock. Thus, if the revoked commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ is published by a channel party, let's say A, party B can immediately publish the revocation transaction $\mathsf{TX}_{\mathsf{RV},i}$. Moreover, since commit transactions are signed using the adaptor signature, once $\mathsf{TX}_{\mathsf{CM},i}$ is published by A, the witness $y_{A,i}$ is revealed to B. Thus, only B who knows both $y_{A,i}$ and $y_{B,i}$ can meet the condition $Y_{A,i} \wedge Y_{B,i}$ in the output of the revocation transaction and hence B will actually be the owner of all the channel funds. Broadcast of the latest commit transaction does not pose any risk to its broadcaster because the parties have not signed its corresponding revocation transaction yet.

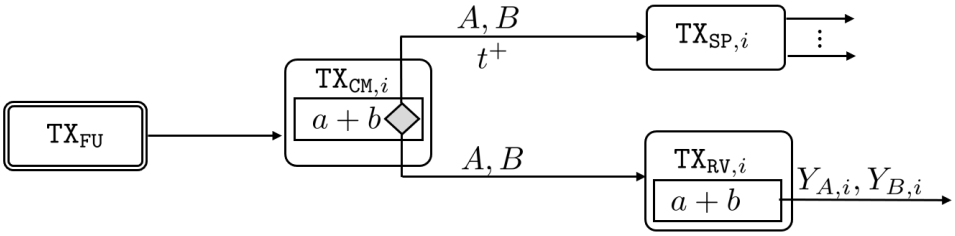

Figure 5.1: NVG Channel Transactions Flow

5.3.2.2 Adding a Watchtower Service with Fairness w.r.t. the Hiring Party to NVG

The watchtower in the introduced scheme is given the revocation transactions for the revoked states and is supposed to publish the one whose corresponding commit transaction is published on the blockchain. However, such a watchtower service would be unfair with respect to the hiring party as the watchtower might become unresponsive upon fraud. To resolve this issue, the watchtower publishes a transaction called *collateral* transaction $\mathsf{TX}_{\mathsf{CL}}$ and locks its collateral in a 3-of-3 multi-signature output shared between channel parties and the watchtower. The value of the collateral c equals the channel capacity $a + b$. If the watchtower does not appropriately act upon fraud, the cheated party has this chance to publish a transaction called *penalty* transaction 1 $\mathsf{TX}_{\mathsf{PN}_1,i}$ and take the watchtower's collateral. In other words, if the revoked commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ is published on the blockchain, either its corresponding revocation transaction $\mathsf{TX}_{\mathsf{RV},i}$ is published by the watchtower or split transaction $\mathsf{TX}_{\mathsf{SP},i}$ and then penalty transaction 1 $\mathsf{TX}_{\mathsf{PN}_1,i}$ are published by the cheating party and honest party, respectively.

As Fig. 5.2 shows, to deploy this, we add an auxiliary output with the least value supported by Bitcoin (denoted by ϵ) to the commit transaction. If the watchtower is responsive, once the revoked commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ is published, the watchtower instantly publishes the revocation transaction $\mathsf{TX}_{\mathsf{RV},i}$ and invalidates both the split $\mathsf{TX}_{\mathsf{SP},i}$ and penalty transaction 1 $\mathsf{TX}_{\mathsf{PN}_1,i}$. Otherwise, the dishonest party, let's say A, also publishes the split transaction and invalidates the revocation transaction. But B, who is

capable to extract A's witness (i.e. $y_{A,i}$), adds the required signatures to the penalty transaction 1 $\mathsf{TX}_{\mathsf{PN}_1,i}$ and publishes it on the blockchain. Penalty transaction 1 also spends the collateral transaction's output. The only output of the penalty transaction 1 $\mathsf{TX}_{\mathsf{PN}_1,i}$ can be claimed by B who knows both $y_{A,i}$ and $y_{B,i}$. Therefore, since the watchtower's collateral equals the channel capacity, B will not lose any funds in the channel.

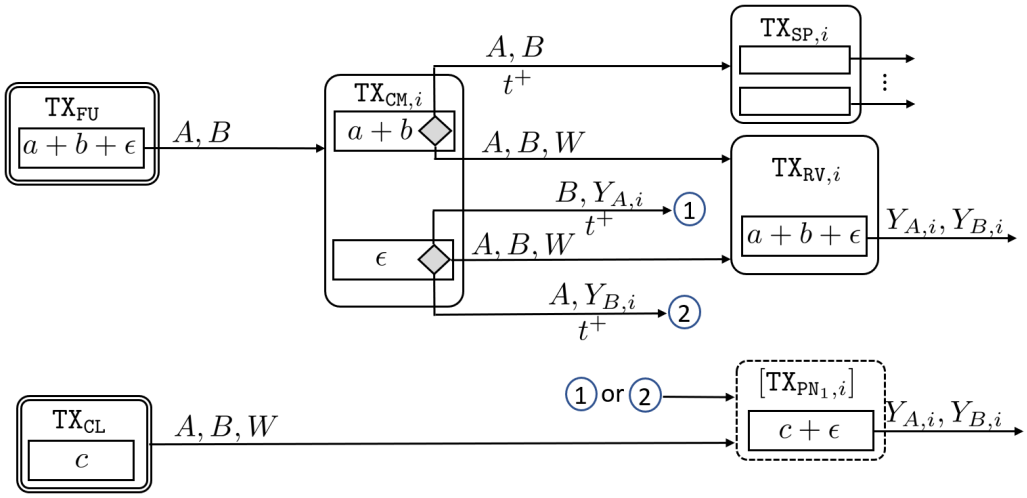

Figure 5.2: Adding a Fair Watchtower to NVG

5.3.2.3 Allowing Watchtower to Terminate its Service

Channel parties in the introduced scheme can go offline for any arbitrary period as the watchtower cannot redeem its collateral without authorisation from both channel parties. However, this would be unfair with respect to the watchtower because the watchtower's collateral might be locked forever. Therefore, there must be a way for the watchtower to terminate its service. So, we add a transaction called *reclaim* transaction $\mathsf{TX}_{\mathsf{RC},i}$ to our solution. Once the watchtower decides to terminate its service, it publishes the reclaim transaction on-chain. Then, the watchtower can claim the reclaim transaction's output after T rounds which is significantly larger than t. We also add a penalty transaction 2 $\mathsf{TX}_{\mathsf{PN}_2,i}$ to each revoked state. The only difference between this transaction and penalty transaction 1 is that penalty transaction 2 spends the output of the reclaim transaction (rather than the collateral transaction's output) as well as the auxiliary output of the commit transaction. It means that while the output of the reclaim transaction is unspent, the watchtower might be penalised by channel parties. So channel parties can practically go offline for a long period (approximately T rounds). Fig. 5.3 depicts the transaction flows for the FPPW channel.

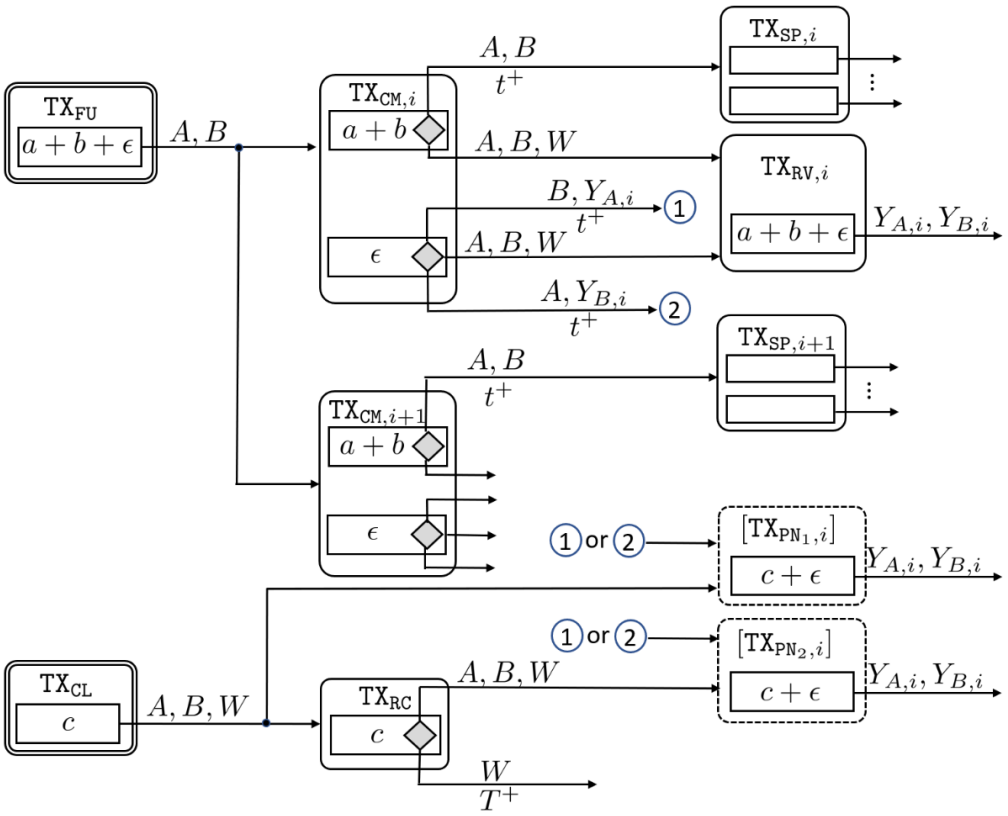

Figure 5.3: FPPW Channel Transaction flow

5.4 FPPW Protocol Description

The lifetime of an FPPW channel can be divided into 5 phases including *create*, *update*, *close*, *react* and *terminate*. We explain these phases in the following sections. The first three phases (create, update and close) correspond to different phases of a payment channel as defined in Definition 4.1. A watchtower scheme is also integrated into the FPPW design. The phases create, update and react in an FPPW channel respectively correspond to the phases start, update and react of a watchtower as defined in Definition 4.4. Also, both phases close and terminate in an FPPW channel lead to the watchtower termination as defined in Definition 4.4. Table 5.2 summarises the mentioned correspondences.

The cryptographic primitives, used in these phases, are as follows: A digital signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$; a hard relation $\mathscr{R}$ with generating algorithm $\mathsf{GenR} = \mathsf{Gen}$; an adaptor signature scheme $\Xi_{\Pi,\mathscr{R}} = (\mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Ext})$. The more detailed definitions of these cryptographic primitives can be found in Section 3.2. We assume that the watchtower is hired by both channel parties. However, FPPW can be simply extended to situations where only one party hires the watchtower. FPPW for such scenarios will be presented in Section 5.10.

Table 5.2: Different Phases of an FPPW Channel

FPPW Phases	Payment Channel Phases	Watchtower Phases
Create	Create	Start
Update	Update	Update
Close	Close	Terminate
React	-	React
Terminate	-	Terminate

5.4.1 Create

FPPW channel creation phase includes a funding transaction, a commit transaction and a split transaction. The funding transaction locks funds of the channel parties in a 2-of-2 multi-signature output and can be claimed only if both parties agree and cooperate with each other. The commit transaction is held by both channel parties and sends all the channel funds to a joint account that can be spent by the corresponding split transaction after t rounds. The split transaction actually represents the channel state and distributes the channel funds between the channel parties. The quantity t, which represents the dispute period, exists to ensure that there is enough time for punishing the dishonest channel party in the case of fraud (i.e. if the published commit transaction corresponds with a revoked state). Parties finally publish the funding transaction on the blockchain to create the channel. However, since its output can be spent if both parties cooperate, one party might lock the funds by being unresponsive. To avoid such situations, before signing and publishing the funding transaction, both channel parties must sign commit and split transactions.

Additionally, two other transactions are created in this phase: (1) the collateral transaction and (2) the reclaim transaction. These two are used for watchtower services. Using the collateral transaction, the watchtower locks its collateral in a 3-of-3 multi-signature output shared between channel parties and the watchtower. Collateral is awarded to the cheated channel party if the watchtower does not appropriately act upon fraud. The value of the collateral equals the channel capacity. Using the reclaim transaction, the watchtower can start the process of reclaiming its collateral. The watchtower can finally redeem its collateral by claiming the output of the reclaim transaction after a large relative time-lock of T rounds with $T \gg t$ which is called the penalty period. If channel parties get online at least once every $T - 1$ rounds, they will always have enough time to take the dishonest watchtower's collateral as compensation and prevent an unresponsive watchtower from redeeming its collateral. However, if the honest watchtower has published the reclaim transaction to withdraw its service, channel parties will have two options. They can either update the channel on-chain with a new watchtower or remain

almost always online to prevent fraudulent channel closures. The collateral transaction is finally recorded on-chain. However, to avoid any hostage situation, before publishing the collateral transaction, the watchtower must receive the channel parties' signatures on the reclaim transaction.

All the above-mentioned transactions are further explained hereinafter.

- **Funding transaction**: Using this transaction, channel parties A and B open an FPPW channel. If A (B, respectively) uses the x^{th} (y^{th}, respectively) output of a transaction with transaction identifier of $txid_A$ ($txid_B$, respectively) to fund the channel with a (b, respectively) coins, the funding transaction is as follows[2]:

$$\begin{aligned} \mathsf{TX}_{\mathsf{FU}}.\mathsf{Input} &:= (txid_A \| x, txid_B \| y), \\ \mathsf{TX}_{\mathsf{FU}}.\mathsf{Output} &:= \{(a+b+\epsilon, pk_A \wedge pk_B)\}, \\ \mathsf{TX}_{\mathsf{FU}}.\mathsf{Witness} &:= ((1, \sigma^{A,1}_{\mathsf{TX}_{\mathsf{FU}}}), (1, \sigma^{B,2}_{\mathsf{TX}_{\mathsf{FU}}})). \end{aligned} \tag{5.1}$$

  where ϵ is the minimum value supported by the Bitcoin blockchain and a and b are the initial balance of A and B in the channel (regardless of the negligible value $\epsilon/2$). In other words, A and B fund the channel with $a + \epsilon/2$ and $b + \epsilon/2$, respectively. Output of $\mathsf{TX}_{\mathsf{FU}}$ is a 2-of-2 multi-signature output shared between A and B. The public keys pk_A and pk_B of A and B are generated using the key generation algorithm of the underlying digital signature Gen.

- **Commit transaction**: There exists one commit transaction per state but only the first one ($\mathsf{TX}_{\mathsf{CM},i}$ with $i = 0$) is created at the channel create phase. $\mathsf{TX}_{\mathsf{CM},i}$ is as follows:

$$\begin{aligned} \mathsf{TX}_{\mathsf{CM},i}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid} \| 1, \\ \mathsf{TX}_{\mathsf{CM},i}.\mathsf{Output} &:= ((a+b, \varphi_1 \vee \varphi_2), \\ &\qquad (\epsilon, \varphi'_1 \vee \varphi'_2 \vee \varphi'_3)) \\ \mathsf{TX}_{\mathsf{CM},i}.\mathsf{Witness} &:= \{(1, \{\sigma^{A}_{\mathsf{TX}_{\mathsf{CM},i}}, \sigma^{B}_{\mathsf{TX}_{\mathsf{CM},i}}\})\} \end{aligned} \tag{5.2}$$

  with $\varphi_1 := pk_A \wedge pk_B \wedge t^+$, $\varphi_2 := pk_A \wedge pk_B \wedge pk_W$, $\varphi'_1 := pk_B \wedge Y_{A,i} \wedge t^+$, $\varphi'_2 := pk_A \wedge pk_B \wedge pk_W$ and $\varphi'_3 := pk_A \wedge Y_{B,i} \wedge t^+$ where $Y_{A,i}$ and $Y_{B,i}$ are statements of a hard relation $\mathscr{R}$ generated by A and B for the i^{th} state using the generating algorithm GenR and t^+ shows relative time-lock of t rounds. The first output with a value of $a + b$ is the main output. Normally, if parties act honestly and $\mathsf{TX}_{\mathsf{CM},i}$ is published

[2]We assume that funding sources of $\mathsf{TX}_{\mathsf{FU}}$ are two typical UTXOs owned by A and B.

on-chain, the first sub-condition of its main output ($pk_A \wedge pk_B \wedge t^+$) is met by $\mathsf{TX}_{\mathsf{SP},i}$ after t rounds. The second output of $\mathsf{TX}_{\mathsf{CM},i}$ with a value of ϵ is the auxiliary output, which as will be explained in Section 5.4.2, is only used for watchtower purposes.

The transaction $\mathsf{TX}_{\mathsf{CM},i}$ requires signatures of both parties A and B to be published. To sign the commit transaction $\mathsf{TX}_{\mathsf{CM},i}$, each party uses their counterparty's statement to generate a pre-signature on the commit transaction for their counterparty. Then, each party can use the corresponding witness to adapt the pre-signature, received from their counterparty to a valid digital signature on the commit transaction. So, if the commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ is published by A, the witness $y_{A,i}$ is revealed to B and vice versa. The witness value, as will be seen in Section 5.4.2, might be used to penalise the dishonest channel party or the unresponsive watchtower.

Remark 5.2. *A has two public keys in the first output of $\mathsf{TX}_{\mathsf{CM},i}$, which for simplicity, we denote them both by pk_A. However, in practice, such public keys are selected dis-jointly. This is also extended to other participants and other outputs.*

- **Split transaction**: $\mathsf{TX}_{\mathsf{SP},i}$ actually represents the i^{th} channel state where only the first one ($\mathsf{TX}_{\mathsf{SP},i}$ with $i = 0$) is created in the channel create phase. This transaction is as follows:

$$\begin{aligned} \mathsf{TX}_{\mathsf{SP},i}.\mathsf{Input} &:= (\mathsf{TX}_{\mathsf{CM},i}.\mathsf{txid} \| 1), \\ \mathsf{TX}_{\mathsf{SP},i}.\mathsf{Output} &:= (\theta_1, \theta_2, \cdots), \\ \mathsf{TX}_{\mathsf{SP},i}.\mathsf{Witness} &:= ((1, \{\sigma^A_{\mathsf{TX}_{\mathsf{SP},i}}, \sigma^B_{\mathsf{TX}_{\mathsf{SP},i}}\}) \end{aligned} \tag{5.3}$$

  The $\mathsf{TX}_{\mathsf{SP},i}$ spends the main output of $\mathsf{TX}_{\mathsf{CM},i}$ by meeting the sub-condition $pk_A \wedge pk_B \wedge t^+$.

- **Collateral transaction**: $\mathsf{TX}_{\mathsf{CL}}$ locks the collateral of the watchtower on-chain and its output can be spent if A, B and W cooperate. The collateral value c equals $a+b$. If W uses the z^{th} output of a transaction with transaction identifier of $txid_W$ to lock c coins, the collateral transaction $\mathsf{TX}_{\mathsf{CL}}$ would be defined as follows:

$$\begin{aligned} \mathsf{TX}_{\mathsf{CL}}.\mathsf{Input} &:= (txid_W \| z), \\ \mathsf{TX}_{\mathsf{CL}}.\mathsf{Output} &:= \{(c, pk_A \wedge pk_B \wedge pk_W)\}, \\ \mathsf{TX}_{\mathsf{CL}}.\mathsf{Witness} &:= \{(1, \sigma^{W,1}_{\mathsf{TX}_{\mathsf{CL}}})\}. \end{aligned} \tag{5.4}$$

- **Reclaim transaction**: This transaction spends the output of $\mathsf{TX}_{\mathsf{CL}}$ and its output can be spent by A, B and W if they cooperate or by W after a long relative time-lock period. The $\mathsf{TX}_{\mathsf{RC}}$ is defined as follows:

$$\begin{aligned} \mathsf{TX}_{\mathsf{RC}}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{CL}}.\mathsf{txid} \| 1), \\ \mathsf{TX}_{\mathsf{RC}}.\mathsf{Output} &:= \{(c, (pk_A \wedge pk_B \wedge pk_W) \vee (pk_W \wedge T^+))\}, \\ \mathsf{TX}_{\mathsf{RC}}.\mathsf{Witness} &:= \{(1, (\sigma^{A,1}_{\mathsf{TX}_{\mathsf{RC}}}, \sigma^{B,1}_{\mathsf{TX}_{\mathsf{RC}}}, \sigma^{W,1}_{\mathsf{TX}_{\mathsf{RC}}}))\}. \end{aligned} \tag{5.5}$$

  The second sub-condition in the output is used by the watchtower to redeem its collateral after T rounds and terminate its service. However, as will be mentioned in the following sections, the first sub-condition is used to penalise the unresponsive watchtower.

Fig. 5.4 summarises the channel create phase. Section 5.7 provides details of the corresponding protocol.

5.4.2 Update

Assume that an FPPW channel is in state i with $i \geq 0$ and channel parties decide to update it from state i to $i+1$. This is performed in two sub-phases. In the first sub-phase, channel parties create a new commit transaction and a new split transaction for the new state. However, to avoid any hostage situation, they sign the split transaction before signing the commit transaction. In the second sub-phase, channel parties revoke the previous state by signing one revocation and two penalty transactions. At most one out of these three transactions might be published on-chain upon fraud (i.e. upon broadcast of the revoked commit transaction). While the revocation transaction might be used to penalise the cheating channel party, penalty transactions might be utilised for punishing the dishonest watchtower.

The revocation transaction is the only transaction that spends both outputs of the revoked commit transaction using their non-time-locked sub-conditions $pk_A \wedge pk_B \wedge pk_W$. Thus, once a dishonest channel party publishes the revoked commit transaction, the watchtower or the counterparty can immediately publish the revocation transaction. It invalidates both penalty transactions because they also spend the auxiliary output of the revoked commit transaction. The single output of the revocation transaction is spendable by someone who knows the witness value y of both channel parties (i.e. party A can claim it if party B has published the revoked commit transaction and vice versa).

Now assume that a dishonest channel party publishes the revoked commit transaction but the watchtower does not react in time. Then the dishonest channel party might also

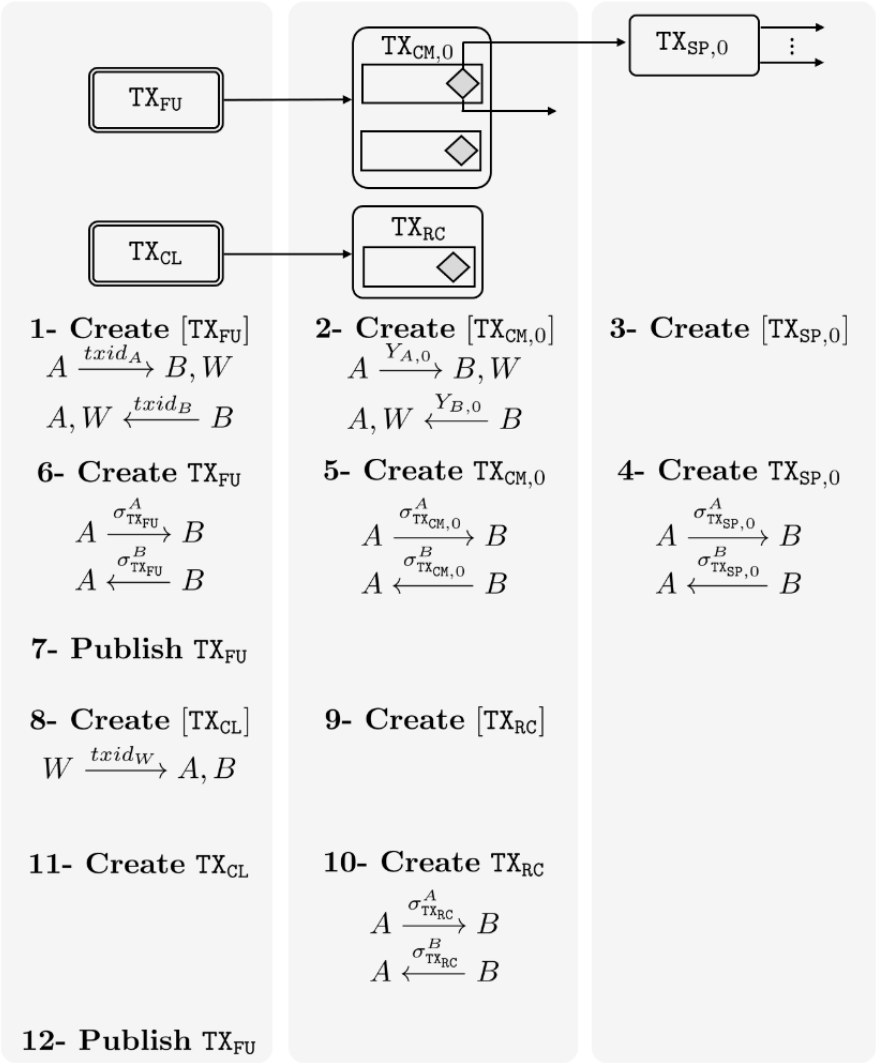

Figure 5.4: A summary of FPPW Channel Create.

publish the corresponding split transaction after t rounds. This spends the main output of the revoked commit transaction and invalidates the revocation transaction. However, since the honest channel party go offline for at most $T - 1$ rounds, it gets online when the watchtower has not completed reclaiming its collateral yet (i.e. the watchtower has not broadcast the reclaim transaction or has not spent its output yet). Thus, the honest party can publish one of two penalty transactions. Both penalty transactions spend the auxiliary output of the revoked commit transaction as well as the output of collateral and reclaim transactions, respectively. Similar to the revocation transaction, only the honest cheated party can claim the output of the published penalty transaction.

The introduced transactions will be explained further below:

- **Revocation transaction**: When parties A and B want to revoke $\mathsf{TX}_{\mathsf{CM},i}$, each channel participant (A, B and W) generates all the required signatures for the revocation transaction $\mathsf{TX}_{\mathsf{RV},i}$ and sends the signatures to other two participants. $\mathsf{TX}_{\mathsf{RV},i}$ is as follows:

$$\begin{aligned}
\mathsf{TX}_{\mathsf{RV},i}.\mathsf{Input} &:= (\mathsf{TX}_{\mathsf{CM},i}.\mathsf{txid}\|1, \mathsf{TX}_{\mathsf{CM},i}.\mathsf{txid}\|2),\\
\mathsf{TX}_{\mathsf{RV},i}.\mathsf{Output} &:= (a+b+\epsilon, Y_{A,i} \wedge Y_{B,i}),\\
\mathsf{TX}_{\mathsf{RV},i}.\mathsf{Witness} &:= ((2,(\sigma^{A,1}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{B,1}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{W,1}_{\mathsf{TX}_{\mathsf{RV},i}})),(2,(\sigma^{A,2}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{B,2}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{RV},i}})))
\end{aligned} \tag{5.6}$$

  The $\mathsf{TX}_{\mathsf{RV},i}$ spends both outputs of $\mathsf{TX}_{\mathsf{CM},i}$ using the non-time-locked sub-condition $pk_A \wedge pk_B \wedge pk_W$ and sends all the channel funds to output with condition $Y_{A,i} \wedge Y_{B,i}$. When a dishonest party, let's say A, publishes the revoked $\mathsf{TX}_{\mathsf{CM},i}$, A must wait for t rounds before being able to publish $\mathsf{TX}_{\mathsf{SP},i}$. However, W or B can immediately publish $\mathsf{TX}_{\mathsf{RV},i}$. Since $\mathsf{TX}_{\mathsf{CM},i}$ has been published by A, party B can obtain $y_{A,i}$. Thus, only party B who knows both $y_{A,i}$ and $y_{B,i}$ will own all the channel funds.

- **Penalty transaction 1**: The penalty transaction 1 $\mathsf{TX}_{\mathsf{PN}_1,i}$ is used to penalise W, given that a dishonest party publishes $\mathsf{TX}_{\mathsf{CM},i}$ and spends its main output using $\mathsf{TX}_{\mathsf{SP},i}$. The $\mathsf{TX}_{\mathsf{PN}_1,i}$ is defined as follows:

$$\begin{aligned}
\mathsf{TX}_{\mathsf{PN}_1,i}.\mathsf{Input} &:= (\mathsf{TX}_{\mathsf{CM}}.\mathsf{txid}, \mathsf{TX}_{\mathsf{CL}}.\mathsf{txid}\|1),\\
\mathsf{TX}_{\mathsf{PN}_1,i}.\mathsf{Output} &:= (c+\epsilon, Y_{A,i} \wedge Y_{B,i}),\\
\mathsf{TX}_{\mathsf{PN}_1,i}.\mathsf{Witness} &:= (W_1,(1,(\sigma^{A,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, \sigma^{B,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}})))
\end{aligned} \tag{5.7}$$

  where $W_1 := (1,(\sigma^{B,1}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, y_{A,i}))$ given that broadcaster of $\mathsf{TX}_{\mathsf{CM},i}$ is A or $W_1 := (3,(\sigma^{A,1}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, y_{B,i}))$ otherwise. When parties want to revoke $\mathsf{TX}_{\mathsf{CM},i}$, A and W (B and W) compute the required signatures for the second input of $\mathsf{TX}_{\mathsf{PN}_1,i}$ and send the signatures to B (A). Now assume that one party, let's say A, publishes the revoked $\mathsf{TX}_{\mathsf{CM},i}$ and spends its main output after t rounds. Then, B obtains $y_{A,i}$ and hence can add the witness W_1 to $\mathsf{TX}_{\mathsf{PN}_1,i}$ and publish it, given that $\mathsf{TX}_{\mathsf{CL}}$.Output is still unspent. Then, the transaction $\mathsf{TX}_{\mathsf{PN}_1,i}$ spends the second output of $\mathsf{TX}_{\mathsf{CM},i}$ using the time-locked sub-condition $pk_B \wedge Y_{A,i} \wedge t^+$ as well as the output of the collateral transaction. Only B can claim output of $\mathsf{TX}_{\mathsf{PN}_1,i}$. A similar scenario occurs if B is the broadcaster of $\mathsf{TX}_{\mathsf{CM},i}$.

- **Penalty transaction 2**: There exists one penalty transaction 2 $\mathsf{TX}_{\mathsf{PN}_2,i}$ per state. It is exactly the same as $\mathsf{TX}_{\mathsf{PN}_1,i}$, with the only difference that it spends $\mathsf{TX}_{\mathsf{RC}}$.Output

(rather that $\mathsf{TX}_{\mathsf{CL}}$.Output) using the sub-condition $pk_A \wedge pk_B \wedge pk_W$. Thus, it is useful for cases where the watchtower does not react upon fraud but by publishing $\mathsf{TX}_{\mathsf{RC}}$ tries to reclaim its collateral. However, since the honest party goes offline for at most $T-1$ rounds, it gets online when $\mathsf{TX}_{\mathsf{RC}}$.Output is still unspent. Thus, the honest party can add the required witness to $[\mathsf{TX}_{\mathsf{PN}_2,i}]$ and publish it. The $\mathsf{TX}_{\mathsf{PN}_2,i}$ is defined as follows:

$$\begin{aligned}
\mathsf{TX}_{\mathsf{PN}_2,i}.\mathsf{Input} &:= (\mathsf{TX}_{\mathsf{CM},i}.\mathsf{txid}, \mathsf{TX}_{\mathsf{RC}}.\mathsf{txid} \| 1), \\
\mathsf{TX}_{\mathsf{PN}_2,i}.\mathsf{Output} &:= (c+\epsilon, Y_{A,i} \wedge Y_{B,i}), \\
\mathsf{TX}_{\mathsf{PN}_2,i}.\mathsf{Witness} &:= (W_1, (1, (\sigma^{A,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}, \sigma^{B,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}})))
\end{aligned} \tag{5.8}$$

where $W_1 := (1, (\sigma^{B,1}_{\mathsf{TX}_{\mathsf{PN}_2,i}}, y_{A,i}))$ given that broadcaster of $\mathsf{TX}_{\mathsf{CM},i}$ is A or $W_1 := (3, (\sigma^{A,1}_{\mathsf{TX}_{\mathsf{PN}_2,i}}, y_{B,i}))$ otherwise..

Fig. 5.5 summarises the channel update phase. Section 5.7 provides details of the corresponding protocol.

Remark 5.3. *Watchtower is actively involved in steps 6 and 7 of the channel update phase (See Fig. 5.5). Therefore, this phase fails to complete if the watchtower is unavailable. Section 5.8 introduces an update protocol for such scenarios.*

5.4.3 Close

Assume that the channel parties A and B have updated their channel n times and then A and/or B decide to close it. They can close the channel cooperatively. To do so, A and B create a new transaction, called modified split transaction $\mathsf{TX}_{\overline{\mathsf{SP}}}$, and publish it on-chain. The $\mathsf{TX}_{\overline{\mathsf{SP}}}$ is defined as follows:

$$\begin{aligned}
\mathsf{TX}_{\overline{\mathsf{SP}}}.\mathsf{Input} &:= (\mathsf{TX}_{\mathsf{FU},i}.\mathsf{txid} \| 1), \\
\mathsf{TX}_{\overline{\mathsf{SP}}}.\mathsf{Output} &:= (\theta_1, \theta_2, \cdots), \\
\mathsf{TX}_{\overline{\mathsf{SP}}}.\mathsf{Witness} &:= ((1, (\sigma^{A}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}, \sigma^{B}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}))
\end{aligned} \tag{5.9}$$

Outputs of this transaction might be similar to those for $\mathsf{TX}_{\mathsf{SP},n}$. Note that the value of auxiliary output of $\mathsf{TX}_{\mathsf{CM},n}$ (ϵ) can also be given to A and B ($\epsilon/2$ each) through outputs of $\mathsf{TX}_{\overline{\mathsf{SP}}}$. If one of the channel parties gets unresponsive, its counterparty can still close the channel non-collaboratively by publishing $\mathsf{TX}_{\mathsf{CM},n}$ and then $\mathsf{TX}_{\mathsf{SP},n}$ on-chain. The collaborative and non-collaborative channel closure protocols can be found in Section 5.7.

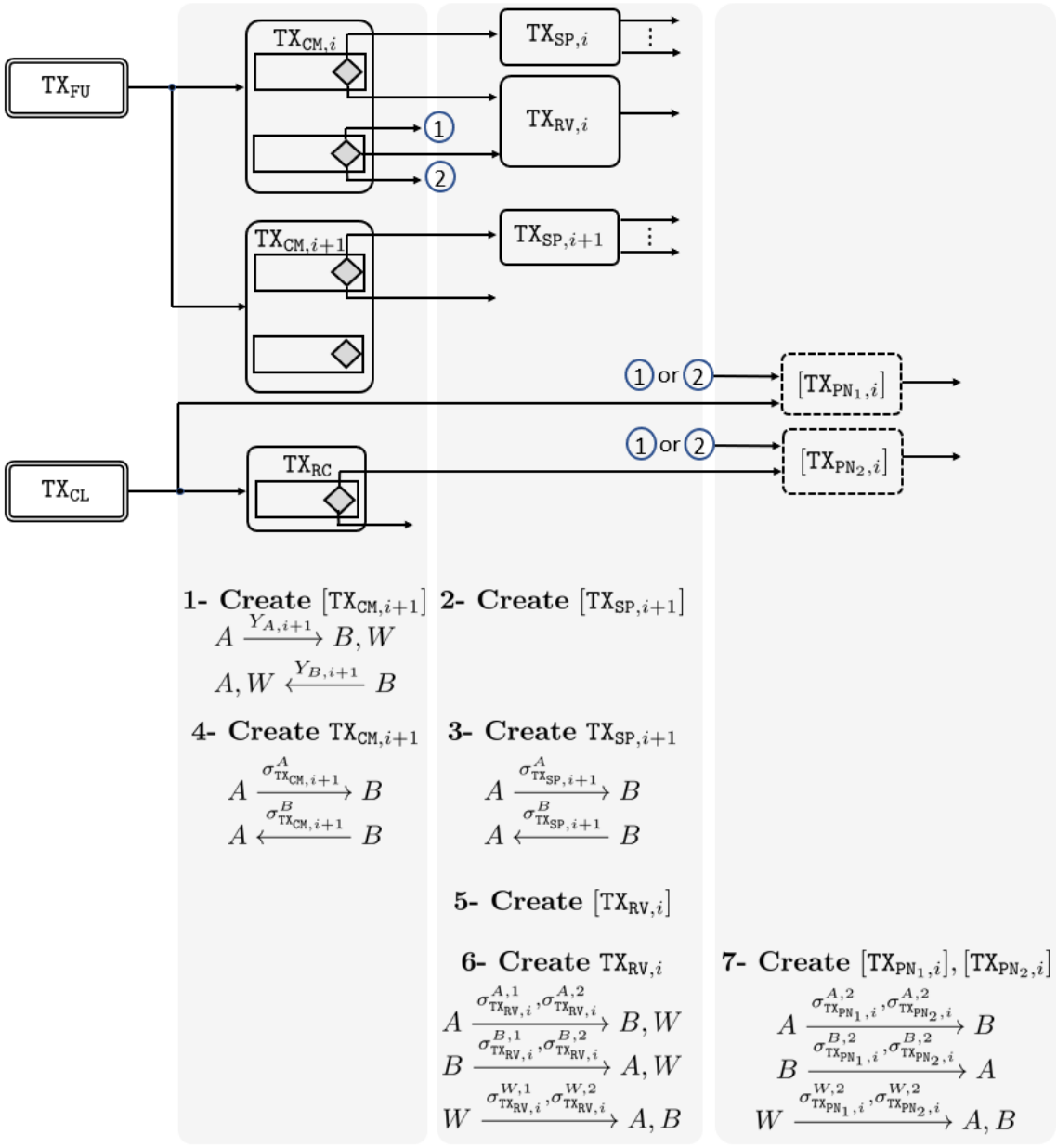

Figure 5.5: FPPW Channel Update.

5.4.4 React

It is always possible that a channel party publishes a revoked commit transaction $\mathtt{TX}_{\mathtt{CM},i}$ on-chain. Then, the watchtower or the counterparty publishes the corresponding revocation transaction within $t-1$ rounds. Only the honest counterparty can claim the output of the revocation transaction. If the watchtower is unresponsive and the honest party is offline, a malicious party can publish a revoked commit transaction $\mathtt{TX}_{\mathtt{CM},i}$ with $i < n$ and its corresponding split transaction $\mathtt{TX}_{\mathtt{SP},i}$ on-chain. Then the honest party, who gets online once every $T-1$ rounds, can penalise the unresponsive watchtower by

publishing either $\mathsf{TX}_{\mathsf{PN}_1,i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$. Protocols for these scenarios can be found in Section 5.7.

5.4.5 Watchtower Terminate

In this phase, W decides to terminate its employment. To do that, W publishes $\mathsf{TX}_{\mathsf{RC}}$ and spends its output after T rounds. Since A and B do not go offline for more than $T-1$ rounds, they get online during this T-round interval and observe that $\mathsf{TX}_{\mathsf{RC}}$ is on the chain. Then parties can close the channel and open a new one with a new watchtower. Parties can also continue using this channel without any watchtower. To do so, channel parties must check the blockchain at least once every $t-1$ rounds to prevent fraudulent channel closures. New channel updates can be performed according to the Generalized channels [14] or its new variant introduced in Section 5.3.2.1.

5.5 Security Analysis

In this section, we analyse privacy and coverage of FPPW protocol through Theorems 5.1 and 5.2, respectively. Then, we analyse the fairness of FPPW through Theorem 5.3.

Theorem 5.1. *FPPW provides weak privacy against watchtower as defined in Definition 4.6.*

Proof. Assume that the conditions mentioned in the two-stage privacy game (see Section 4.3.2) are satisfied. By observing different steps and transactions of the protocol, one can see that only split transactions contain information on $x_{A,i}$ and $x_{B,i}$ with $i \in [1, n-1]$. However, these transactions are never published on-chain or sent to the watchtower or any external entity. Other transactions in the protocol contain no information regarding $x_{A,i}$ or $x_{B,i}$ with $i \in [1, n-1]$. Note that the monetary value of outputs of $\mathsf{TX}_{\mathsf{CM},i}$, $\mathsf{TX}_{\mathsf{RV},i}$, $\mathsf{TX}_{\mathsf{PN}_1,i}$, $\mathsf{TX}_{\mathsf{PN}_2,i}$, $\mathsf{TX}_{\mathsf{CL}}$, and $\mathsf{TX}_{\mathsf{RC}}$ of the first payment channels are the same as those for the second one. Furthermore, $\mathsf{TX}_{\mathsf{FU}}$, $\mathsf{TX}_{\mathsf{SP},n}$ or $\mathsf{TX}_{\overline{\mathsf{SP}}}$ contain no information regarding the i^{th} channel state with $i \in [1, n-1]$. Thus, the view of any adversary $\mathscr{A}$ on $(\mathbf{x}_{A,[1,n-1]}, \mathbf{x}_{B,[1,n-1]})$ is indistinguishable from its view on $(\mathbf{x}_{A',[1,n-1]}, \mathbf{x}_{B',[1,n-1]})$. □

Theorem 5.2. *FPPW provides β-coverage with $\beta = 1/2$ based on Definition 4.11.*

Proof. Assume that we have N payment channels, with channel capacities $X_i = a_i + b_i$, $i \in [1, N]$. Thus, the total capacity of the channels is $\mathscr{X} = \sum_{i=1}^{N} X_i$. Since the i^{th} channel collateral c_i equals $a_i + b_i$, the total watchtower collateral is $\mathscr{C} = \sum_{i=1}^{N} c_i = \sum_{i=1}^{N} X_i = \mathscr{X}$. Thus, we have $\beta = \frac{\mathscr{X}}{\mathscr{X}+\mathscr{C}} = 1/2$. □

As mentioned earlier, Theorem 5.3 analyses the fairness of FPPW. Lemmas 5.1 and 5.2 are utilised to prove this theorem. They show how FPPW guarantees that funds of the honest channel party and the honest watchtower are safe in the channel. We first prove Lemmas 5.1 and 5.2 and then, considering these two lemmas, we prove Theorem 5.3.

Lemma 5.1. *Let* Π *be a* $\mathsf{EUF-CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. For an FPPW channel, assume that the honest channel party* $P \in \{A, B\}$ *checks the blockchain at the end of the channel creation phase and then gets online periodically with a period of at most* $T - 1$ *rounds. The probability that* P *loses any funds in the channel is negligible.*

Proof. Without loss of generality let $P = A$. Based on Lemma 5.4, A does not lose any funds with non-negligible probability unless a revoked commit transaction is published on-chain. However, at the end of the channel creation phase, no revoked commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ exists to be published on-chain. Furthermore, the channel creation completes once $\mathsf{TX}_{\mathsf{CL}}$ is recorded on-chain and since $\mathsf{TX}_{\mathsf{RC}}$ spends the output of $\mathsf{TX}_{\mathsf{CL}}$, right after the end of the channel create phase, neither a revoked commit transaction nor the reclaim transaction is on-chain. Now according to assumptions, A checks the chain at the end of the channel create phase and goes offline for at most $T-1$ rounds. The next time that A gets online, there will be 4 possibilities regarding the broadcast of a revoked $\mathsf{TX}_{\mathsf{CM},i}$ or $\mathsf{TX}_{\mathsf{RC}}$ during the time interval when A has been offline:

1. Only $\mathsf{TX}_{\mathsf{RC}}$ has been published on-chain,
2. Both a revoked $\mathsf{TX}_{\mathsf{CM},i}$ and $\mathsf{TX}_{\mathsf{RC}}$ have been published,
3. Only a revoked $\mathsf{TX}_{\mathsf{CM},i}$ has been published, or
4. neither a revoked $\mathsf{TX}_{\mathsf{CM},i}$ nor $\mathsf{TX}_{\mathsf{RC}}$ has been published.

For the first 3 possibilities, based on Lemmas 5.5, 5.6, and 5.7, the probability that A loses any funds is negligible. For the 4^{th} possibility, the condition set is the same as the end of the channel create phase when neither a revoked commit transaction nor the reclaim transaction was on the blockchain. Therefore, since A will again go offline for at most $T-1$ rounds, the whole process repeats. □

Lemma 5.2. *Let* Π *be a* $\mathsf{EUF-CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. For an FPPW channel, assume that the honest watchtower* W *checks the blockchain at the end of the channel creation phase and then remains online. The probability that* W *loses any funds in the channel is negligible.*

Proof. Based on Lemma 5.8, W does not lose any funds with non-negligible probability unless a revoked commit transaction and then at least t rounds later its corresponding $\mathsf{TX}_{\mathsf{PN}_1,i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$ are published. However, at the end of the channel create phase, no revoked commit transaction exists to be published yet. Also, following our assumptions, W remains online after the channel create phase. Now assume that a revoked $\mathsf{TX}_{\mathsf{CM},i}$ is published through the block $\mathscr{B}_j$. The time-locked sub-conditions of outputs of $\mathsf{TX}_{\mathsf{CM},i}$ cannot be met within $t-1$ rounds. Also, meeting the non-time-locked sub-conditions of the first and second output of $\mathsf{TX}_{\mathsf{CM},i}$ requires W's signature and W does not grant such authorisations on any transaction other than $\mathsf{TX}_{\mathsf{RV},i}$. Thus, once $\mathsf{TX}_{\mathsf{CM},i}$ is published on the chain, due to our assumption regarding the security of the underlying digital signature, it is of negligible probability that any adversary $\mathscr{A}$ spends the first or second output of $\mathsf{TX}_{\mathsf{CM},i}$ within $t-1$ rounds using any transaction other than $\mathsf{TX}_{\mathsf{RV},i}$. Furthermore, W has received $\mathsf{TX}_{\mathsf{RV},i}$ (through step 6 of the channel update phase from state i to $i+1$ in Fig. 5.5), before giving authorisation on the second input of $\mathsf{TX}_{\mathsf{PN}_1,i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$ (through step 7 of the channel update phase from state i to $i+1$ in Fig. 5.5). Thus, once $\mathsf{TX}_{\mathsf{CM},i}$ is published, W can publish $\mathsf{TX}_{\mathsf{RV},i}$ through one of the blocks $\mathscr{B}_{j+1} \cdots \mathscr{B}_{j+t-1}$, meaning that the honest watchtower has at least $t-1$ rounds time to publish $\mathsf{TX}_{\mathsf{RV},i}$, which is enough based on our blockchain assumptions regarding the value of the confirmation delay. Following Lemma 5.9, it is of negligible probability that broadcast of $\mathsf{TX}_{\mathsf{RV},i}$ causes the honest watchtower W to lose any funds in the channel. □

Theorem 5.3. *Let* Π *be a* $\mathsf{EUF-CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. FPPW provides channel party* α*-fairness with* $\alpha = 1$ *and watchtower fairness as defined in Definition 4.9 and 4.10, respectively.*

Proof. The honest channel party always have at least one non-revoked commit transaction and its corresponding split transaction by broadcasting which she can close the channel. This proves that FPPW meets the first requirement of Definition 4.9. Furthermore, we know that based on FPPW protocol, the honest channel party checks the chain at least once every $T-1$ round and according to Lemma 5.1, the probability that the honest channel party loses any funds in the channel is negligible. This proves that FPPW provides channel party α-fairness with $\alpha = 1$.

The watchtower in FPPW is paid for channel creation and each channel update and hence her reward amount is non-zero. Also, we know that based on FPPW protocol, the honest watchtower always remains online and according to Lemma 5.2, the probability that such an honest watchtower loses any funds in the channel is negligible. Additionally, the watchtower can publish $\mathsf{TX}_{\mathsf{RC}}$ at any time and redeem her collateral after T rounds. Thus, FPPW meets both requirements of Definition 4.10. □

Lemma 5.3. *Let* Π *be a* EUF − CMA *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. For an FPPW channel, it is of negligible probability that broadcast of* $\mathsf{TX}_{\mathsf{RV},i}$, $\mathsf{TX}_{\mathsf{PN}_1,i}$ *or* $\mathsf{TX}_{\mathsf{PN}_2,i}$ *causes the honest channel party* $P \in \{A, B\}$ *to lose any funds in the channel.*

Proof. Without loss of generality let $P = A$. The transaction $\mathsf{TX}_{\mathsf{RV},i}$ spends the main output of the revoked $\mathsf{TX}_{\mathsf{CM},i}$ and hence cannot be published unless $\mathsf{TX}_{\mathsf{CM},i}$ is on-chain. Based on the protocol, the honest party A never broadcasts the revoked $\mathsf{TX}_{\mathsf{CM},i}$ on-chain. The party A only creates the pre-signature $\tilde{\sigma}_{\mathsf{CM},i}$ on the transaction $\mathsf{TX}_{\mathsf{CM},i}$. Thus, if $\mathsf{TX}_{\mathsf{CM},i}$ is published, the probability that A fails to obtain $y_{B,i}$ is negligible. Otherwise, aEUF − CMA security or witness extractability of the used adaptor signature is violated. Furthermore, $\mathsf{TX}_{\mathsf{RV},i}$ has only one output with the condition of $Y_{A,i} \wedge Y_{B,i}$ and the value of $a+b+\epsilon$. Since A privately preserves its witness value $y_{A,i}$, the probability that any PPT adversary claims $\mathsf{TX}_{\mathsf{RV},i}$.Output is negligible. Otherwise, the utilised hard relation would break. Therefore, it is of negligible probability that A (who knows both $y_{A,i}$ and $y_{B,i}$) fails to claim $\mathsf{TX}_{\mathsf{RV},i}$.Output.

Also transactions $\mathsf{TX}_{\mathsf{PN}_1,i}$ and $\mathsf{TX}_{\mathsf{PN}_2,i}$ spend the auxiliary output of $\mathsf{TX}_{\mathsf{CM},i}$ as well as output of $\mathsf{TX}_{\mathsf{CL}}$ and $\mathsf{TX}_{\mathsf{RC}}$, respectively. These transactions have one output with the condition of $Y_{A,i} \wedge Y_{B,i}$ and the value of $c + \epsilon = a + b + \epsilon$. Since output condition for $\mathsf{TX}_{\mathsf{PN}_1,i}$ and $\mathsf{TX}_{\mathsf{PN}_2,i}$ is the same as that of $\mathsf{TX}_{\mathsf{RV},i}$, the proof for $\mathsf{TX}_{\mathsf{PN}_1,i}$ and $\mathsf{TX}_{\mathsf{PN}_2,i}$ is also similar to that of $\mathsf{TX}_{\mathsf{RV},i}$. □

Remark 5.4. *Before stating the next Lemma, it must be noted that while the channel update phase from state* i *to* $i+1$ *is incomplete yet, it is possible that* $\mathsf{TX}_{\mathsf{CM},i+1}$ *is published on-chain and* t *rounds later* $\mathsf{TX}_{\mathsf{SP},i+1}$ *is also broadcast. In such situations, we assume that the channel party does not lose any funds in the channel even if her counterparty's balance in state* $i+1$ *is larger than that of state* i*. In other words, it is assumed that during steps 6 and 7 of the channel update phase (see Fig. 5.5), there are two valid channel states where channel closure with each one does not cause the honest party to lose any funds in the channel. A similar assumption is also made for other payment channels of type replace by revocation [15, 14].*

Lemma 5.4. *Let* Π *be a* EUF − CMA *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. For an FPPW channel with* n *channel updates, it is of negligible probability that the honest party* $P \in \{A, B\}$ *loses any funds using any scenario other than the broadcast of* $\mathsf{TX}_{\mathsf{CM},i}$ *with* $i < n$*.*

Proof. Without loss of generality let $P = A$. Funds of A are locked in $\mathsf{TX}_{\mathsf{FU}}$.Output. It is of negligible probability that any PPT adversary $\mathscr{A}$ spends the output of $\mathsf{TX}_{\mathsf{FU}}$ without the honest party A's authorisation. Otherwise, the underlying digital signature would be forgeable. Furthermore, $\mathsf{TX}_{\overline{\mathsf{SP}}}$, $\mathsf{TX}_{\mathsf{CM},i}$ with $i = [0, n-1]$, $\mathsf{TX}_{\mathsf{CM},n}$ and possibly $\mathsf{TX}_{\mathsf{CM},n+1}$ (given

that channel update phase from state n to $n+1$ has started) are the only transactions in the protocol that spend the output of $\mathsf{TX}_{\mathsf{FU}}$ and A grants authorisation for. Thus, these transactions will be discussed further to see how each one of them can cause the honest party A to be cheated out of its funds.

Since $\mathsf{TX}_{\overline{\mathsf{SP}}}$ represents the final agreed state of the channel, its broadcast cannot cause A to lose any funds in the channel. Additionally, since both sub-conditions in the main output of $\mathsf{TX}_{\mathsf{CM},n}$ include pk_A, due to our assumption regarding the security of the underlying digital signature, it is of negligible probability that the main output of $\mathsf{TX}_{\mathsf{CM},n}$ is spent without A's authorisation. Since $\mathsf{TX}_{\mathsf{SP},n}$ is the only transaction in the protocol that spends the main output of $\mathsf{TX}_{\mathsf{CM},n}$ and A grants authorisation for, the probability of spending the main output of $\mathsf{TX}_{\mathsf{CM},n}$ using any transaction other than $\mathsf{TX}_{\mathsf{SP},n}$ is negligible. However, since $\mathsf{TX}_{\mathsf{SP},n}$ also represents the final state of the channel, its broadcast cannot cause A to lose any funds. According to Remark 5.4, similar statements can be stated for $\mathsf{TX}_{\mathsf{CM},n+1}$ and $\mathsf{TX}_{\mathsf{SP},n+1}$. Thus, cheating the honest party A using any scenario other than broadcast of $\mathsf{TX}_{\mathsf{CM},i}$ with $i < n$ is of negligible probability.

□

Case 1. Let there exist an FPPW channel with n channel updates where $n \geq 0$. Assume that the honest channel party $P \in \{A, B\}$ gets online when the last published block on the blockchain is $\mathscr{B}_j$. Party P observes that $\mathsf{TX}_{\mathsf{CM},i}$ is unpublished but $\mathsf{TX}_{\mathsf{RC}}$ has been published through the block $\mathscr{B}_k$ with $k \leq j$.

Lemma 5.5. *Let* Π *be a* $\mathsf{EUF-CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. If conditions of Case 1 are satisfied, the probability that* P *loses any funds in the channel is negligible.*

Proof. Without loss of generality let $P = A$. If party A checks the blockchain and observes that $\mathsf{TX}_{\mathsf{CM},i}$ is unpublished but $\mathsf{TX}_{\mathsf{RC}}$ has been published, A can get online periodically with a period of at most $t-1$ rounds. Based on Lemma 5.4, if A is going to lose some funds in the channel with non-negligible probability, the PPT adversary $\mathscr{A}$ must publish $\mathsf{TX}_{\mathsf{CM},i}$ with $i < n$ through the block $\mathscr{B}_k$ with $k > j$. It must be noted that since the next time that A gets online again, $\mathscr{B}_{j+t-1}$ is the latest block on the blockchain, if we have $k > j+t-1$, all the conditions mentioned for Case 1 repeat with j being replaced with $j+t-1$. Thus, we assume that we have $j < k \leq j+t-1$.

time-locked sub-conditions of outputs of $\mathsf{TX}_{\mathsf{CM},i}$ cannot be met within $t-1$ rounds. Also, their non-time-locked sub-conditions include pk_A. Therefore, it is of negligible probability that any adversary spends the first or the second output of $\mathsf{TX}_{\mathsf{CM},i}$ through one of the blocks $\mathscr{B}_{k+1}, \cdots, \mathscr{B}_{k+t-1}$ without the honest party A's authorisation. Otherwise, the underlying digital signature would break. However, A never grants such authorisations

on a transaction other than $\mathsf{TX}_{\mathsf{RV},i}$. Furthermore, the honest party A has created $\mathsf{TX}_{\mathsf{RV},i}$ through step 6 of the channel update phase from state i to $i+1$ (See Fig. 7). When A gets online the last block on the blockchain is $\mathscr{B}_{j+t-1}$. Thus, A is able to publish $\mathsf{TX}_{\mathsf{RV},i}$ through one of the blocks $\mathscr{B}_{j+t}, \cdots, \mathscr{B}_{k+t-1}$. Since we have $k-j \geq 1$, A always have at least 1 block time to publish $\mathsf{TX}_{\mathsf{RV},i}$, which is enough based on our blockchain assumption regarding the value of the confirmation delay. Also, according to Lemma 5.3, it is of negligible probability that broadcast of $\mathsf{TX}_{\mathsf{RV},i}$ causes the honest party A to lose any funds in the channel. □

Case 2. Let there exist an FPPW channel with n channel updates where $n \geq 0$. Assume that the honest channel party $P \in \{A, B\}$ gets online when the last published block on the blockchain is $\mathscr{B}_j$. Party P observes that $\mathsf{TX}_{\mathsf{CM},i}$ with $i < n$ and $\mathsf{TX}_{\mathsf{RC}}$ have been published on-chain through the blocks $\mathscr{B}_k$ and $\mathscr{B}_l$ respectively with $k, l \leq j$ and $j+1 < l+T$.

Lemma 5.6. *Let* Π *be a* $\mathsf{EUF-CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. If conditions of Case 2 are satisfied, the probability that* P *loses any funds in the channel is negligible.*

Proof. Without loss of generality let $P = A$. All sub-conditions of auxiliary output of $\mathsf{TX}_{\mathsf{CM},i}$ include either pk_A or $Y_{A,i}$. Thus, it is of negligible probability that any PPT adversary $\mathscr{A}$ spends this output of $\mathsf{TX}_{\mathsf{CM},i}$ without A's authorisation. Otherwise, the underlying digital signature or the hard relation would break. The channel party A grants such an authorisation only on $\mathsf{TX}_{\mathsf{RV},i}$. However, based on Lemma 5.3, if $\mathsf{TX}_{\mathsf{RV},i}$ is published, the probability that A loses any funds is negligible. Thus, we assume that when A gets online, $\mathsf{TX}_{\mathsf{RV},i}$ is unpublished and hence the auxiliary output of $\mathsf{TX}_{\mathsf{CM},i}$ is unspent yet. According to values of j and k two categories of cases are possible. We firstly consider cases with $j+1 \geq k+t$ and prove that in such cases A can publish $\mathsf{TX}_{\mathsf{PN}_2,i}$. Then, we show that if $j+1 < k+t$, A can still publish $\mathsf{TX}_{\mathsf{RV},i}$. Thus, according to Lemma 5.3, for all cases, the probability that A loses any funds is negligible.

Consider cases with $j+1 \geq k+t$. The non-time-locked sub-condition of $\mathsf{TX}_{\mathsf{RC}}$.Output includes pk_A and hence due to our assumption regarding the security of the underlying digital signature, it is of negligible probability that the output of $\mathsf{TX}_{\mathsf{RC}}$ is spent without A's authorisation through the block $\mathscr{B}_m$ with $m < l+T$. Also A grants this authorisation only on $\mathsf{TX}_{\mathsf{PN}_2,i}$. Actually, all channel participants grant such authorisation on $\mathsf{TX}_{\mathsf{PN}_2,i}$. Additionally, since $\mathsf{TX}_{\mathsf{CM},i}$ is on-chain, the probability that A fails to obtain $y_{B,i}$ and hence fails to generate the required signatures for the first input of $\mathsf{TX}_{\mathsf{PN}_2,i}$ is negligible. Otherwise, $\mathsf{aEUF-CMA}$ security or extractability of the used adaptor signature is violated. Thus, party A might publish $\mathsf{TX}_{\mathsf{PN}_2,i}$ on-chain through one of the blocks $\mathscr{B}_{j+1}, \cdots, \mathscr{B}_{l+T-1}$ and since based on assumptions of Case 2 we have $j+1 \leq l+T-1$, the honest party A

will have at least $l + T - 1 - j \geq 1$ rounds time to publish $\mathsf{TX}_{\mathsf{PN}_2,i}$ which is enough based on our assumptions regarding the value of confirmation delay.

Now let $j + 1 < k + t$. We know that without having the honest party A's authorisation, it is of negligible probability that any PPT adversary is able to spend the first or the second output of $\mathsf{TX}_{\mathsf{CM},i}$ through one of the blocks $\mathscr{B}_{k+1}, \cdots, \mathscr{B}_{k+t-1}$. All the channel participants grant such authorisations only on $\mathsf{TX}_{\mathsf{RV},i}$ and A has created this transaction in step 6 of the channel update phase from state i to $i + 1$ (See Fig. 5.5). Thus, A can publish it through one of the blocks $\mathscr{B}_{j+1}, \cdots, \mathscr{B}_{k+t-1}$ and since $j+1 < k+t$, A will have at least $k + t - 1 - j \geq 1$ rounds time to publish $\mathsf{TX}_{\mathsf{RV},i}$.

Therefore, for all cases, A can publish either $\mathsf{TX}_{\mathsf{RV},i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$ and hence according to Lemma 5.3, the probability that A loses any funds is negligible.

□

Case 3. Let there exist an FPPW channel with n channel updates where $n \geq 0$. Assume that the honest channel party $P \in \{A, B\}$ gets online when the last published block on the blockchain is $\mathscr{B}_j$. Party P observes that $\mathsf{TX}_{\mathsf{CM},i}$ has been published on-chain through the block $\mathscr{B}_k$ with $k \leq j$ but $\mathsf{TX}_{\mathsf{RC}}$ is unpublished.

Lemma 5.7. *Let* Π *be a* $\mathsf{EUF} - \mathsf{CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. If conditions of Case 3 are satisfied, the probability that* P *loses any funds in the channel is negligible.*

Proof. Without loss of generality let $P = A$. Similar to the proof of Lemma 5.6, we assume that $\mathsf{TX}_{\mathsf{RV},i}$ is unpublished because otherwise, the probability that A loses some funds is negligible. According to proof of Lemma 5.6, if we have $j + 1 < k + t$, A will have at least $k + t - 1 - j \geq 1$ rounds time to publish $\mathsf{TX}_{\mathsf{RV},i}$.

Now let $j+1 \geq k+t$. We know that due to the security of the underlying digital signature, it is of negligible probability that someone spends $\mathsf{TX}_{\mathsf{CL}}$.Output without A's authorisation and A grants this authorisation only on $\mathsf{TX}_{\mathsf{RC}}$ and $\mathsf{TX}_{\mathsf{PN}_1,i}$. Also, since $\mathsf{TX}_{\mathsf{CM},i}$ is on-chain, the probability that A fails to obtain $y_{B,i}$ is negligible. Thus, it is of negligible probability that party A cannot generate the required signatures for the first input of $\mathsf{TX}_{\mathsf{PN}_1,i}$. Then, party A can publish $\mathsf{TX}_{\mathsf{PN}_1,i}$ on-chain through one of the blocks $\mathscr{B}_{j+1}, \mathscr{B}_{j+2}, \cdots$. Based on Lemma 5.3, it is of negligible probability that broadcast of $\mathsf{TX}_{\mathsf{PN}_1,i}$ can cause A to lose any funds.

If before the broadcast of $\mathsf{TX}_{\mathsf{PN}_1,i}$, $\mathsf{TX}_{\mathsf{RC}}$ is published by the watchtower, conditions of Case 2 get satisfied and due to Lemma 5.6, we know that it is of negligible probability that A loses any funds in the channel.

□

Lemma 5.8. *Let* Π *be a* $\mathsf{EUF-CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. For an FPPW channel with n channel updates, an honest watchtower* W *does not lose any funds with non-negligible probability unless first a revoked commit transaction* $\mathsf{TX}_{\mathsf{CM},i}$ *with* $i < n$ *is broadcast on the chain and at least* t *rounds later, either* $\mathsf{TX}_{\mathsf{PN}_1,i}$ *or* $\mathsf{TX}_{\mathsf{PN}_2,i}$ *is also published on the chain.*

Proof. Assume that the channel update phase has completed n times with $n \geq 0$. We discuss different scenarios using which the honest watchtower can be cheated. Funds of the watchtower are locked in $\mathsf{TX}_{\mathsf{CL}}$.Output and condition of $\mathsf{TX}_{\mathsf{CL}}$.Output includes pk_W. Thus, due to security of the underlying digital signature, it is of negligible probability that any PPT adversary $\mathscr{A}$ spends $\mathsf{TX}_{\mathsf{CL}}$.Output without W's authorisation. $\mathsf{TX}_{\mathsf{RC}}$ and $\mathsf{TX}_{\mathsf{PN}_1,i}$ with $i = [0, n-1]$ are the only transactions in the protocol that spend $\mathsf{TX}_{\mathsf{CL}}$.Output and W grants authorisation for. Both sub-conditions of $\mathsf{TX}_{\mathsf{RC}}$.Output include pk_W. Thus, due to the security of the underlying digital signature, it is of negligible probability that any PPT adversary $\mathscr{A}$ spends $\mathsf{TX}_{\mathsf{RC}}$.Output without W's authorisation and $\mathsf{TX}_{\mathsf{PN}_2,i}$ with $i = [0, n-1]$ are only transactions in the protocol that spend $\mathsf{TX}_{\mathsf{RC}}$.Output and W grants authorisation for. Therefore, all scenarios with non-negligible probability lead to the broadcast of $\mathsf{TX}_{\mathsf{PN}_1,i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$. Since both $\mathsf{TX}_{\mathsf{PN}_1,i}$ and $\mathsf{TX}_{\mathsf{PN}_2,i}$ take the auxiliary output of $\mathsf{TX}_{\mathsf{CM},i}$ as their first inputs, those transactions can only be published if $\mathsf{TX}_{\mathsf{CM},i}$ is on-chain. Meeting the non-time-locked sub-condition of the auxiliary output of $\mathsf{TX}_{\mathsf{CM},i}$ requires W's authorisation and W does not grant such an authorisation on $\mathsf{TX}_{\mathsf{PN}_1,i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$. Therefore, it is of negligible probability that any PPT adversary publishes $\mathsf{TX}_{\mathsf{PN}_1,i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$ before t rounds being elapsed since the broadcast of $\mathsf{TX}_{\mathsf{CM},i}$. □

Lemma 5.9. *Let* Π *be a* $\mathsf{EUF-CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. For an FPPW channel with n channel updates, if* $\mathsf{TX}_{\mathsf{CM},i}$ *with* $i \in [0, n-1]$ *is published on-chain, it is of negligible probability that broadcast of* $\mathsf{TX}_{\mathsf{RV},i}$ *causes the honest watchtower* W *to lose any funds in the channel.*

Proof. Based on Lemma 5.8, the probability of cheating the watchtower without publishing $\mathsf{TX}_{\mathsf{PN}_1,i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$ is negligible. However, since $\mathsf{TX}_{\mathsf{RV},i}$, $\mathsf{TX}_{\mathsf{PN}_1,i}$ and $\mathsf{TX}_{\mathsf{PN}_2,i}$ spend auxiliary output of $\mathsf{TX}_{\mathsf{CM},i}$, if $\mathsf{TX}_{\mathsf{RV},i}$ is published on-chain, $\mathsf{TX}_{\mathsf{PN}_1,i}$ and $\mathsf{TX}_{\mathsf{PN}_2,i}$ cannot be recorded on-chain anymore. □

5.6 Fee Handling

Once a revoked commit transaction is recorded on the blockchain, the watchtower must record its corresponding revocation transaction within $t - 1$ rounds. Otherwise, the watchtower might be penalised. However, the time it takes for a transaction to be

recorded on the blockchain depends on its fee value and the network congestion. The body of a revocation transaction is created during the channel update phase but it might be broadcast in the blockchain network later upon fraud. Thus, the fee amount must be large enough to ensure the watchtower that the revocation transaction will be accepted by miners within the dispute period. In other words, when channel participants are creating a revocation transaction, they must assume that the blockchain network will be highly congested at the time when fraud will occur.

An alternative approach involves manipulating the SIGHASH type parameter to accommodate an increase in the fee amount during congestion periods. SIGHASH is a component of a transaction that specifies which parts of the transaction data is signed by the participants. By specifying a specific SIGHASH type, participants can determine which parts of the transaction are included in the signature, and which parts can be modified without invalidating the signature. There are different SIGHASH types, each denoted by a specific number, and they serve various purposes [57]. For our use case, we propose the usage of SIGHASH of type 0x81 (SIGHASH_ALL | SIGHASH_ANYONECANPAY) for channel parties' signatures for both inputs of revocation transactions. Thus, a signature for each input applies to that input and the output. Therefore, when due to network congestion the considered fee for the revocation transaction is low, the watchtower can add some inputs to the revocation transaction to increase the fee amount, sign all inputs using SIGHASH of type 0x01 (SIGHASH_ALL) and submit it to the network. If there exists enough time, the watchtower might even repeat this process several times and raise this extra fee each time until one of the revocation transactions is accepted by the miners. This method can be used if revocation transactions are only held by the watchtower (i.e. if channel parties do not receive signatures of the watchtower on revocation transactions during the channel update phase).

A similar approach can also be used for penalty transactions. Channel parties and the watchtower can use SIGHASH of type 0x02 (SIGHASH_NONE) for the second input of penalty transactions. Then, signatures apply only to all inputs of penalty transactions. In this way, the watchtower can be certain that a penalty transaction cannot be published unless its corresponding commit transaction is on-chain. However, if a revoked $\mathsf{TX}_{\mathsf{CM},i}$ is published by a channel party, let's say A, and its main output is spent by $\mathsf{TX}_{\mathsf{SP},i}$, party B has the opportunity to set the output value of the penalty transaction according to the network congestion and sign the corresponding penalty transaction (to meet the sub-condition $pk_B \wedge Y_{A,i} \wedge t^+$) using SIGHASH of type 0x01 (SIGHASH_ALL). In this way, B can reduce the output value if the current fee is low and this difference value is used as the extra fee amount. If there exists enough time, B can even repeat this process multiple times, each time with a higher fee until one penalty transaction is recorded on-chain.

5.7 FPPW Protocol

In this section, protocols for different phases of FPPW will be presented. In different steps of the protocol, channel participants generate (or verify) some signatures or pre-signatures on protocol transactions. When a signature or pre-signature is going to be generated (or verified) for j^{th} input of the transaction TX_i, the input message to the signing (or verification) algorithm is denoted by $f([\mathsf{TX}_i], j)$ [58].

Remark 5.5. *The FPPW protocol which is presented in this Section is slightly different from what was explained in Section 5.4. Those differences are as follows. In the channel create phase,* $[\mathsf{TX}_{\mathsf{RV},0}]$ *is also created and W's signatures on this transaction are given to both A and B. This transaction lacks the channel parties' signatures. Those signatures are computed in the channel update phase when* $\mathsf{TX}_{\mathsf{CM},0}$ *is going to be revoked. However, A and B require W's signature in this phase to be able to continue using the channel given that W will be non-cooperative or temporarily unavailable in the channel update phase from state 0 to 1. Otherwise, A and B will have to update the channel on-chain because* $\mathsf{TX}_{\mathsf{CM},0}$ *will be irrevocable. Similarly, in the update channel phase from state* i *to* $i+1$*, W also generates its signatures for both inputs of* $\mathsf{TX}_{\mathsf{RV},i+1}$ *and sends them to A and B. This will allow A and B to update the channel from state* $i+1$ *to* $i+2$ *and revoke* $\mathsf{TX}_{\mathsf{CM},i+1}$ *even if W will be temporarily unavailable or uncooperative. Otherwise,* $\mathsf{TX}_{\mathsf{CM},i+1}$ *will be irrevocable.*

FPPW channel create protocol is as follows:

Preconditions: A, B and W own $a+\epsilon/2$, $b+\epsilon/2$ and $c = a+b$ coins on-chain in output of transactions with transaction identifiers $txid_A$, $txid_B$ and $txid_W$ respectively. A, B and W know each other's public keys, pk_A, pk_B and pk_W and values of ϵ, a, b and c that will be used in the channel.

1. Create $[\mathsf{TX}_{\mathsf{FU}}]$:
   (a) $P \in \{A, B\} \xleftrightarrow{txid_P} \bar{P}, W$
   (b) If P receives $txid_{\bar{P}}$, it creates $[\mathsf{TX}_{\mathsf{FU}}]$ according to 5.1. Else it stops.
   (c) If W receives $txid_P$ with $P = \{A, B\}$, it creates $[\mathsf{TX}_{\mathsf{FU}}]$ according to 5.1. Else it stops.
2. Create $[\mathsf{TX}_{\mathsf{CM},0}]$:
   (a) $P \in \{A, B\}$ generates $(Y_{P,0}, y_{P,0}) \leftarrow \mathsf{GenR}$.
   (b) $P \xleftrightarrow{Y_{P,0}} \bar{P}, W$
   (c) If P receives $Y_{\bar{P},0}$, it creates $[\mathsf{TX}_{\mathsf{CM},0}]$ according to 5.2. Else it stops.

(d) If W receives $Y_{P,0}$ with $P = \{A, B\}$, it creates $[\mathsf{TX}_{\mathsf{CM},0}]$ according to 5.2. Else it stops.

3. Create $[\mathsf{TX}_{\mathsf{SP},0}]$: Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{SP},0}]$ according to 5.3.

4. Create $[\mathsf{TX}_{\mathsf{RV},0}]$:

   (a) W creates $[\mathsf{TX}_{\mathsf{RV},0}]$ according to 5.6.

   (b) W computes $\sigma^{W,j}_{\mathsf{TX}_{\mathsf{RV},0}} = \mathsf{Sign}_{sk_W}(f([\mathsf{TX}_{\mathsf{RV},0}], j))$ with $j = \{1, 2\}$.

   (c) $W \xhookrightarrow{\sigma^{W,1}_{\mathsf{TX}_{\mathsf{RV},0}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{RV},0}}} A, B$.

   (d) If party $P \in \{A, B\}$ receives $\sigma^{W,1}_{\mathsf{TX}_{\mathsf{RV},0}}$ and $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{RV},0}}$ from W s.t. $\mathsf{Vrfy}_{pk_W}(f([\mathsf{TX}_{\mathsf{RV},0}], j); \sigma^{W,j}_{\mathsf{TX}_{\mathsf{RV},0}}) = 1$ with $j = \{1, 2\}$, it continues. Else it stops.

5. Create $\mathsf{TX}_{\mathsf{SP},0}$:

   (a) Party $P \in \{A, B\}$ computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{SP},0}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{SP},0}], 1))$.

   (b) $P \xhookrightarrow{\sigma^{P}_{\mathsf{TX}_{\mathsf{SP},0}}} \bar{P}$

   (c) If party P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{SP},0}}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{SP},0}], 1); \sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{SP},0}}) = 1$, it continues. Else it stops.

   (d) Party P creates $\mathsf{TX}_{\mathsf{SP},0}$.

6. Create $\mathsf{TX}_{\mathsf{CM},0}$:

   (a) Party $P \in \{A, B\}$ computes $\tilde{\sigma}^{P}_{\mathsf{TX}_{\mathsf{CM},0}} = \mathsf{pSign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},0}], 1), Y_{\bar{P},0})$.

   (b) $P \xhookrightarrow{\tilde{\sigma}^{P}_{\mathsf{TX}_{\mathsf{CM},0}}} \bar{P}$

   (c) If party P receives $\tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},0}}$ s.t. $\mathsf{pVrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{CM},0}], 1), Y_{P,0}; \tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},0}}) = 1$, it computes $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},0}} = \mathsf{Adapt}(\tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},0}}, y_{P,0})$, computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{CM},0}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},0}], 1))$, creates $\mathsf{TX}_{\mathsf{CM},0}$ and continues. Else it stops.

7. Create $\mathsf{TX}_{\mathsf{FU}}$:

   (a) Party $P \in \{A, B\}$ computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{FU}}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{FU}}], j))$ where $j := 1$ if $P = A$ or $j := 2$ otherwise.

   (b) $P \xhookrightarrow{\sigma^{P}_{\mathsf{TX}_{\mathsf{FU}}}} \bar{P}$

   (c) If party P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{FU}}}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{FU}}], j); \sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{FU}}}) = 1$ with $j := 1$ if $P = B$ or $j := 2$ otherwise, it continues. Else it stops.

(d) Party P creates $\mathsf{TX}_{\mathsf{FU}}$.

8. Publish $\mathsf{TX}_{\mathsf{FU}}$: Party P publishes $\mathsf{TX}_{\mathsf{FU}}$ on-chain.
9. Create $[\mathsf{TX}_{\mathsf{CL}}]$:
   (a) W creates $[\mathsf{TX}_{\mathsf{CL}}]$ according to 5.4.
   (b) $W \xhookrightarrow{txid_W} A, B$.
   (c) If party P receives $txid_W$, it creates $[\mathsf{TX}_{\mathsf{CL}}]$. Else it stops.
10. Create $[\mathsf{TX}_{\mathsf{RC}}]$: A, B and W creates $[\mathsf{TX}_{\mathsf{RC}}]$ according to 5.5.
11. Create $\mathsf{TX}_{\mathsf{RC}}$:
   (a) Party $P \in \{A, B\}$ computes $\sigma^P_{\mathsf{TX}_{\mathsf{RC}}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RC}}], 1))$.
   (b) $P \xhookrightarrow{\sigma^P_{\mathsf{TX}_{\mathsf{RC}}}} W$
   (c) If W receives $\sigma^P_{\mathsf{TX}_{\mathsf{RC}}}$ with $P = \{A, B\}$ s.t. $\mathsf{Vrfy}_{pk_P}(f([\mathsf{TX}_{\mathsf{RC}}], 1); \sigma^P_{\mathsf{TX}_{\mathsf{RC}}}) = 1$, it continues. Else it stops.
   (d) W computes $\sigma^W_{\mathsf{TX}_{\mathsf{RC}}} = \mathsf{Sign}_{sk_W}(f([\mathsf{TX}_{\mathsf{RC}}], 1))$ and creates $\mathsf{TX}_{\mathsf{RC}}$.
12. Create $\mathsf{TX}_{\mathsf{CL}}$:
   (a) W computes $\sigma^W_{\mathsf{TX}_{\mathsf{RV}}} = \mathsf{Sign}_{sk_W}(f([\mathsf{TX}_{\mathsf{CL}}], 1))$.
   (b) W creates $\mathsf{TX}_{\mathsf{CL}}$.
13. Publish $\mathsf{TX}_{\mathsf{CL}}$: W publishes $\mathsf{TX}_{\mathsf{CL}}$ on-chain.

FPPW channel update protocol is as following:

Preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ and $\mathsf{TX}_{\mathsf{CL}}$ are on-chain. The channel update phase has completed i times and hence the channel is at state i.

1. Create $[\mathsf{TX}_{\mathsf{CM},i+1}]$:
   (a) Party $P \in \{A, B\}$ generates $(Y_{P,i+1}, y_{P,i+1}) \leftarrow \mathsf{GenR}$.
   (b) $P \xhookrightarrow{Y_{P,i+1}} \bar{P}, W$
   (c) If Party P receives $Y_{\bar{P},i+1}$, it creates $[\mathsf{TX}_{\mathsf{CM},i+1}]$ according to 5.2. Else it stops.

(d) If W receives $Y_{P,i+1}$ with $P = \{A, B\}$, it creates $[\mathsf{TX}_{\mathsf{CM},i+1}]$ according to 5.2. Else it stops.

2. Create $[\mathsf{TX}_{\mathsf{SP},i+1}]$: Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{SP},i+1}]$ according to 5.3.

3. Create $[\mathsf{TX}_{\mathsf{RV},i+1}]$:

   (a) W creates $[\mathsf{TX}_{\mathsf{RV},i+1}]$ according to 5.6.

   (b) W computes $\sigma^{W,j}_{\mathsf{TX}_{\mathsf{RV},i+1}} = \mathsf{Sign}_{sk_W}([f(\mathsf{TX}_{\mathsf{RV},i+1}], j))$ with $j = \{1, 2\}$.

   (c) $W \xhookrightarrow{\sigma^{W,1}_{\mathsf{TX}_{\mathsf{RV},i+1}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{RV},i+1}}} A, B.$

   (d) If party $P \in \{A, B\}$ receives $\sigma^{W,1}_{\mathsf{TX}_{\mathsf{RV},i+1}}$ and $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{RV},i+1}}$ from W s.t. $\mathsf{Vrfy}_{pk_W}(f([\mathsf{TX}_{\mathsf{RV},i+1}], j); \sigma^{W,j}_{\mathsf{TX}_{\mathsf{RV},i+1}}) = 1$ with $j = \{1, 2\}$, it continues. Else it stops.

4. Create $\mathsf{TX}_{\mathsf{SP},i+1}$:

   (a) Party $P \in \{A, B\}$ computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{SP},i+1}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{SP},i+1}], 1))$.

   (b) $P \xhookrightarrow{\sigma^{P}_{\mathsf{TX}_{\mathsf{SP},i+1}}} \bar{P}$

   (c) If party P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{SP},i+1}}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{SP},i+1}], 1); \sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{SP},i+1}}) = 1$, it continues. Else it stops.

   (d) Party P creates $\mathsf{TX}_{\mathsf{SP},i+1}$.

5. Create $\mathsf{TX}_{\mathsf{CM},i+1}$:

   (a) Party $P \in \{A, B\}$ computes $\tilde{\sigma}^{P}_{\mathsf{TX}_{\mathsf{CM},i+1}} = \mathsf{pSign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},i+1}], 1), Y_{\bar{P},i+1})$.

   (b) $P \xhookrightarrow{\tilde{\sigma}^{P}_{\mathsf{TX}_{\mathsf{CM},i+1}}} \bar{P}$

   (c) If party P receives $\tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},i+1}}$ s.t. $\mathsf{pVrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{CM},i+1}], 1), Y_{P,i+1}; \tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},i+1}}) = 1$, it computes $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},i+1}} = \mathsf{Adapt}(\tilde{\sigma}^{\bar{P}}_{\mathsf{CM},i+1}, y_{P,i+1})$, computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{CM},i+1}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},i+1}], 1))$, creates $\mathsf{TX}_{\mathsf{CM},i+1}$ and continues. Else it execute the non-collaborative closure phase (from P's point of view the channel is still at state i).

6. Create $\mathsf{TX}_{\mathsf{RV},i}$:

   (a) Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{RV},i}]$ according to 5.6.

   (b) Party P computes $\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 1))$ and $\sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 2))$.

(c) $P \xleftrightarrow{\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}}} \bar{P}$

(d) If party P receives $\sigma^{\bar{P},1}_{\mathsf{TX}_{\mathsf{RV},i}}$ and $\sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{RV},i}}$ from $\bar{P}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{RV},i}], 1); \sigma^{\bar{P},1}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$ and $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{RV},i}], 2); \sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$, it continues. Else it executes the non-collaborative closure phase (The channel now is at state $i + 1$).

(e) $P \xleftrightarrow{\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}}} W$

(f) If W receives $\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}}$ and $\sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}}$ from $P = \{A, B\}$ s.t. $\mathsf{Vrfy}_{pk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 1); \sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$ and $\mathsf{Vrfy}_{pk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 2); \sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$, it continues. Else it stops.

(g) A, B and W create $\mathsf{TX}_{\mathsf{RV},i}$ according to 5.6.

7. Create $[\mathsf{TX}_{\mathsf{PN}_1,i}]$ and $[\mathsf{TX}_{\mathsf{PN}_2,i}]$

(a) Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{PN}_1,i}]$ and $[\mathsf{TX}_{\mathsf{PN}_2,i}]$ according to 5.7 and 5.8.

(b) Party P computes $\sigma^{P,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 2))$ and $\sigma^{P,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 2))$.

(c) $P \xleftrightarrow{\sigma^{P,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, \sigma^{P,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}} \bar{P}$

(d) If P receives signatures $\sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}$ and $\sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 2); \sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}) = 1$ and $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 2); \sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}) = 1$, it continues. Else it executes the non-collaborative closure phase (The channel now is at state $i + 1$) or gets online at least once every $t - 1$ blocks.

(e) W creates $[\mathsf{TX}_{\mathsf{PN}_1,i}]$ and $[\mathsf{TX}_{\mathsf{PN}_2,i}]$ according to 5.7 and 5.8

(f) W computes $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}} = \mathsf{Sign}_{sk_W}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 2)$ and $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}} = \mathsf{Sign}_{sk_W}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 2)$.

(g) $W \xleftrightarrow{\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}} P$ with $P = \{A, B\}$.

(h) If $P \in \{A, B\}$ receives signatures $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}$ and $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}$ from W s.t. $\mathsf{Vrfy}_{pk_W}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 2); \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}) = 1$ and $\mathsf{Vrfy}_{pk_W}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 2); \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}) = 1$ it continues. Else it executes the non-collaborative closure phase (The channel now is at state $i + 1$) or gets online at least once every $t - 1$ blocks.

FPPW channel collaborative closure protocol is as following:

Preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ and $\mathsf{TX}_{\mathsf{CL}}$ are on-chain. The channel is at state n.

1. Create $\mathsf{TX}_{\overline{\mathsf{SP}}}$:
   (a) Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\overline{\mathsf{SP}}}]$ according to 5.9.
   (b) P computes $\sigma^{P}_{\mathsf{TX}_{\overline{\mathsf{SP}}}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\overline{\mathsf{SP}}}], 1))$.
   (c) $P \xhookrightarrow{\sigma^{P}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}} \bar{P}$
   (d) If P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}$ from $\bar{P}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\overline{\mathsf{SP}}}], 1); \sigma^{\bar{P}}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}) = 1$, it continues. Else it executes the non-collaborative closure phase (from P's point of view the channel is still at state n).
2. Publish $\mathsf{TX}_{\overline{\mathsf{SP}}}$: Party $P \in \{A, B\}$ publishes $\mathsf{TX}_{\overline{\mathsf{SP}}}$ on-chain.

FPPW channel non-collaborative closure protocol is as following:

Preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ and $\mathsf{TX}_{\mathsf{CL}}$ are on-chain. The channel is at state n.

1. Party $P \in \{A, B\}$ publishes $\mathsf{TX}_{\mathsf{CM},n}$ on-chain.
2. Once $\mathsf{TX}_{\mathsf{CM},n}$ is recorded on-chain, P waits for t rounds and then publishes $\mathsf{TX}_{\mathsf{SP},n}$ on-chain.

The protocol for penalising the cheating party is as follows:

preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ and $\mathsf{TX}_{\mathsf{CL}}$ are on-chain. The channel is at state n. $\mathsf{TX}_{\mathsf{CM},i}$ with $i < n$ is recorded on-chain by a channel party. The watchtower is always online.

1. W observes that $\mathsf{TX}_{\mathsf{CM},i}$ is on-chain.
2. W publishes $\mathsf{TX}_{\mathsf{RV},i}$ on-chain before t rounds being elapsed since broadcast of $\mathsf{TX}_{\mathsf{CM},i}$.

The protocol for penalising the unresponsive watchtower is as follows:

preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ and $\mathsf{TX}_{\mathsf{CL}}$ are on-chain. The channel update phase has successfully completed n times. $\mathsf{TX}_{\mathsf{CM},i}$ with

$i < n$ is recorded on-chain. Parties check the blockchain at least once every $T - 1$ round.

1. Party P observes that $\mathscr{B}_j$ is the latest block on the blockchain and $\mathsf{TX}_{\mathsf{CM},i}$ has been published through the block $\mathscr{B}_k$ with $k \leq j$ but its first output has not been spent by $\mathsf{TX}_{\mathsf{RV},i}$.
2. if $j + 1 - k < t$:
   (a) P publishes $\mathsf{TX}_{\mathsf{RV},i}$ on the blockchain.

   Otherwise

   (a) P extract $\mathsf{TX}_{\mathsf{CM},i}.\mathsf{Witness}[1]$ as $(1, (\sigma^{A}_{\mathsf{TX}_{\mathsf{CM},i}}, \sigma^{B}_{\mathsf{TX}_{\mathsf{CM},i}}))$.
   (b) P computes $y_{\bar{P},i} = \mathsf{Ext}(\sigma^{P}_{\mathsf{TX}_{\mathsf{CM},i}}, \tilde{\sigma}^{P}_{\mathsf{CM},i}, Y_{\bar{P},i})$.
   (c) If $\mathsf{TX}_{\mathsf{RC}}$ is unpublished:
   - P computes $\sigma^{P,1}_{\mathsf{TX}_{\mathsf{PN}_1,i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 1))$ and $\sigma'^{\bar{P},1}_{\mathsf{TX}_{\mathsf{PN}_1,i}} = \mathsf{Sign}_{y_{\bar{P},i}}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 1))$.
   - P creates $\mathsf{TX}_{\mathsf{PN}_1,i}$ using $[\mathsf{TX}_{\mathsf{PN}_1,i}]$, $\mathsf{TX}_{\mathsf{PN}_1,i}.\mathsf{Witness}[1] = (1, (\sigma^{P,1}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, \sigma'^{\bar{P},1}_{\mathsf{TX}_{\mathsf{PN}_1,i}}))$ with $j = 1$ if $P = B$ or $j = 3$ otherwise and $\mathsf{TX}_{\mathsf{PN}_1,i}.\mathsf{Witness}[2] = (1, (\sigma^{A,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, \sigma^{B,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}))$.
   - P publishes $\mathsf{TX}_{\mathsf{PN}_1,i}$ on-chain.

   Else:
   - P computes $\sigma^{P,1}_{\mathsf{TX}_{\mathsf{PN}_2,i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 1))$ and $\sigma'^{\bar{P},1}_{\mathsf{TX}_{\mathsf{PN}_2,i}} = \mathsf{Sign}_{y_{\bar{P},i}}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 1))$.
   - P creates $\mathsf{TX}_{\mathsf{PN}_2,i}$ using $[\mathsf{TX}_{\mathsf{PN}_2,i}]$, $\mathsf{TX}_{\mathsf{PN}_2,i}.\mathsf{Witness}[1] = (j, (\sigma^{P,1}_{\mathsf{TX}_{\mathsf{PN}_2,i}}, \sigma'^{\bar{P}_Y,1}_{\mathsf{TX}_{\mathsf{PN}_2,i}}))$ with $j = 1$ if $P = B$ or $j = 3$ otherwise and $\mathsf{TX}_{\mathsf{PN}_2,i}.\mathsf{Witness}[2] = (1, (\sigma^{A,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}, \sigma^{B,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}))$.
   - P publishes $\mathsf{TX}_{\mathsf{PN}_2,i}$ on-chain.

5.8 Temporarily unavailable watchtower

FPPW can adapt to situations where channel parties want to update the channel state but the watchtower is temporarily unavailable. A way to deal with such occasions is that channel parties wait for the watchtower to get responsive. However, it disturbs the main functionality of the payment channel. Another solution is updating the channel by skipping those steps that require the watchtower to sign the new revocation transaction

(see step 3 of the channel update protocol in Section 5.7) and new penalty transactions (see step 7 of the channel update protocol in Section 5.7). This solution also has two problems. Firstly, without the watchtower cooperation, the new commit transaction gets irrevocable. Thus, channel parties would not be able to update the channel multiple times. Secondly, if the watchtower gets uncooperative, channel parties will be forced to update the channel on-chain even if they both agree to remain always online and continue using the channel.

To resolve this issue, channel parties can take steps in the channel update phase using the following commit transactions:

$$\begin{aligned} \mathsf{TX}_{\mathsf{CM},i+1}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid}\|1, \\ \mathsf{TX}_{\mathsf{CM},i+1}.\mathsf{Output} &:= ((a+b, \varphi_1 \vee \varphi_2 \vee \varphi_3), (\epsilon, \varphi_1' \vee \varphi_2' \vee \varphi_3')) \\ \mathsf{TX}_{\mathsf{CM},i+1}.\mathsf{Witness} &:= \{(1, \{\sigma^{A}_{\mathsf{TX}_{\mathsf{CM},i}}, \sigma^{B}_{\mathsf{TX}_{\mathsf{CM},i}}\})\} \end{aligned} \tag{5.10}$$

with $\varphi_1 := pk_A \wedge pk_B \wedge 2t^+$, $\varphi_2 := pk_A \wedge pk_B \wedge pk_W$, $\varphi_3 := pk_A \wedge pk_B \wedge t^+$, $\varphi_1' := pk_B \wedge Y_{A,i+1} \wedge t^+$, $\varphi_2' := pk_A \wedge pk_B \wedge pk_W$ and $\varphi_3' := pk_A \wedge Y_{B,i+1} \wedge t^+$ where $Y_{A,i+1}$ and $Y_{B,i+1}$ are statements of a hard relation $\mathscr{R}$ generated by A and B for the $i+1^{\text{th}}$ state using the generating algorithm GenR.

Also, a new type of revocation transaction is introduced as follows:

$$\begin{aligned} \mathsf{TX}_{\mathsf{RV}',i+1}.\mathsf{Input} &:= (\mathsf{TX}_{\mathsf{CM},i+1}.\mathsf{txid}\|1), \\ \mathsf{TX}_{\mathsf{RV}',i+1}.\mathsf{Output} &:= (a+b, Y_{A,i+1} \wedge Y_{B,i+1}), \\ \mathsf{TX}_{\mathsf{RV}',i+1}.\mathsf{Witness} &:= \{(3, (\sigma^{A,1}_{\mathsf{TX}_{\mathsf{RV}',i+1}}, \sigma^{B,1}_{\mathsf{TX}_{\mathsf{RV}',i+1}}))\} \end{aligned} \tag{5.11}$$

Then, the update protocol is as follows:

Preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ and $\mathsf{TX}_{\mathsf{CL}}$ are on-chain. The channel update phase has completed i times and hence the channel is at state i. The watchtower is unavailable.

1. Create $[\mathsf{TX}_{\mathsf{CM},i+1}]$:
   (a) Party P generates $(Y_{P,i+1}, y_{P,i+1}) \leftarrow \mathsf{GenR}$.
   (b) $P \xrightarrow{Y_{P,i+1}} \bar{P}$

(c) If Party P receives $Y_{\bar{P},i+1}$, it creates $[\mathsf{TX}_{\mathsf{CM},i+1}]$ according to 5.10. Else it stops.

2. Create $[\mathsf{TX}_{\mathsf{SP},i+1}]$: Party P creates $[\mathsf{TX}_{\mathsf{SP},i+1}]$ according to 5.3.

3. Create $[\mathsf{TX}_{\mathsf{RV}',i+1}]$: Party P creates $[\mathsf{TX}_{\mathsf{RV}',i+1}]$ according to 5.11.

4. Create $\mathsf{TX}_{\mathsf{SP},i+1}$:

   (a) Party P computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{SP},i+1}} = \mathsf{Sign}_{sk_P}([f(\mathsf{TX}_{\mathsf{SP},i+1}], 1))$.

   (b) $P \xhookrightarrow{\sigma^{P}_{\mathsf{TX}_{\mathsf{SP},i+1}}} \bar{P}$

   (c) If party P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{SP},i+1}}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{SP},i+1}], 1); \sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{SP},i+1}}) = 1$, it continues. Else it stops.

   (d) Party P creates $\mathsf{TX}_{\mathsf{SP},i+1}$.

5. Create $\mathsf{TX}_{\mathsf{CM},i+1}$:

   (a) Party P computes $\tilde{\sigma}^{P}_{\mathsf{CM},i+1} = \mathsf{pSign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},i+1}], 1), Y_{\bar{P},i+1})$.

   (b) $P \xhookrightarrow{\tilde{\sigma}^{P}_{\mathsf{CM},i+1}} \bar{P}$

   (c) If party P receives $\tilde{\sigma}^{\bar{P}}_{\mathsf{CM},i+1}$ s.t. $\mathsf{pVrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{CM},i+1}], 1), Y_{P,i+1}; \tilde{\sigma}^{\bar{P}}_{\mathsf{CM},i+1}) = 1$, it computes $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},i+1}} = \mathsf{Adapt}(\tilde{\sigma}^{\bar{P}}_{\mathsf{CM},i+1}, y_{P,i+1})$, computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{CM},i+1}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},i+1}], 1))$, creates $\mathsf{TX}_{\mathsf{CM},i+1}$ and continues. Else it execute the non-collaborative closure phase (from P's point of view the channel is still at state i).

6. Create $\mathsf{TX}_{\mathsf{RV},i}$ or $\mathsf{TX}_{\mathsf{RV}',i}$:

   (a) If $\mathsf{TX}_{\mathsf{CM},i}$ is according to 5.2,

      i. Party P computes $\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 1))$ and $\sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 2))$.

      ii. $P \xhookrightarrow{\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}}} \bar{P}$

      iii. If party P receives $\sigma^{\bar{P},1}_{\mathsf{TX}_{\mathsf{RV},i}}$ and $\sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{RV},i}}$ from $\bar{P}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{RV},i}], 1); \sigma^{\bar{P},1}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$ and $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{RV},i}], 2); \sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$, it continues. Else it executes the non-collaborative closure phase (The channel now is at state $i + 1$).

      iv. Party P creates $\mathsf{TX}_{\mathsf{RV},i}$.

   (b) Otherwise:

i. Party P computes $\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV}',i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RV}',i}), 1])$.

ii. $P \xhookrightarrow{\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV}',i}}} \bar{P}$.

iii. If party P receives $\sigma^{\bar{P},1}_{\mathsf{TX}_{\mathsf{RV}',i}}$ from $\bar{P}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{RV}',i}], 1); \sigma^{\bar{P},1}_{\mathsf{TX}_{\mathsf{RV}',i}}) = 1$, it continues. Else it executes the non-collaborative closure phase (The channel now is at state $i+1$).

iv. Party P creates $\mathsf{TX}_{\mathsf{RV}',i}$.

Remark 5.6. *The first (second) condition in step 6 corresponds with the case when watchtower was available (unavailable) during the channel update from state $i-1$ to i.*

Now assume that one of the revoked $\mathsf{TX}_{\mathsf{CM},i}$, for which $\mathsf{TX}_{\mathsf{RV}',i}$ has been created, is published by a channel party. The cheating party must wait $2t$ rounds to be able to publish the split transaction $\mathsf{TX}_{\mathsf{SP},i}$. However, its online counterparty can wait for t rounds and then publish the $\mathsf{TX}_{\mathsf{RV}',i}$ and take all the funds of the channel. The steps of this procedure are as follows:

Preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ and $\mathsf{TX}_{\mathsf{CL}}$ are on-chain. The channel update phase has successfully completed n times. $\mathsf{TX}_{\mathsf{CM},i}$ with $i < n$ is recorded on-chain. Parties have created $\mathsf{TX}_{\mathsf{RV}',i}$. Channel parties are always online.

1. P observes that $\mathsf{TX}_{\mathsf{CM},i}$ is on-chain. P waits for t blocks.
2. P publishes $\mathsf{TX}_{\mathsf{RV}',i}$ on-chain.

When the watchtower becomes available, $\mathsf{TX}_{\mathsf{RV},i}$, $\mathsf{TX}_{\mathsf{PN}_1,i}$ and $\mathsf{TX}_{\mathsf{PN}_2,i}$ (respectively according to 5.6, 5.7 and 5.8) for the new agreed states can be created by A, B and W. The steps of this procedure are as follows:

Preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ and $\mathsf{TX}_{\mathsf{CL}}$ are on-chain. The channel update phase has successfully completed m times with $m \geq 1$. The watchtower W was unavailable during the latest k channel updates with $0 < k \leq m$. The watchtower W is now available.

1. A, B and W repeats the following steps for $i = m-k+1$ to $i = m-1$:
   (a) Create $[\mathsf{TX}_{\mathsf{RV},i}]$:
      i. W creates $[\mathsf{TX}_{\mathsf{RV},i}]$ according to 5.6.

ii. W computes $\sigma^{W,j}_{\mathsf{TX}_{\mathsf{RV},i}} = \mathsf{Sign}_{sk_W}([f(\mathsf{TX}_{\mathsf{RV},i}], j))$ with $j = \{1, 2\}$.

iii. $W \xleftrightarrow{\sigma^{W,1}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{RV},i}}} A, B.$

iv. If party $P \in \{A, B\}$ receives $\sigma^{W,1}_{\mathsf{TX}_{\mathsf{RV},i}}$ and $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{RV},i}}$ from W s.t. $\mathsf{Vrfy}_{pk_W}(f([\mathsf{TX}_{\mathsf{RV},i}], j); \sigma^{W,j}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$ with $j = \{1, 2\}$, it continues. Else it stops.

(b) Create $\mathsf{TX}_{\mathsf{RV},i}$:

i. Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{RV},i}]$ according to 5.6.

ii. Party P computes $\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 1))$ and $\sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 2))$.

iii. $P \xleftrightarrow{\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}}} \bar{P}$

iv. If party P receives $\sigma^{\bar{P},1}_{\mathsf{TX}_{\mathsf{RV},i}}$ and $\sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{RV},i}}$ from $\bar{P}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{RV},i}], 1); \sigma^{\bar{P},1}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$ and $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{RV},i}], 2); \sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$, it continues. Else it executes the non-collaborative closure phase (The channel now is at state m).

v. $P \xleftrightarrow{\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}}} W.$

vi. If W receives $\sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}}$ and $\sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}}$ from $P = \{A, B\}$ s.t. $\mathsf{Vrfy}_{pk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 1); \sigma^{P,1}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$ and $\mathsf{Vrfy}_{pk_P}(f([\mathsf{TX}_{\mathsf{RV},i}], 2); \sigma^{P,2}_{\mathsf{TX}_{\mathsf{RV},i}}) = 1$, it continues. Else it stops.

vii. A, B and W create $\mathsf{TX}_{\mathsf{RV},i}$ using $[\mathsf{TX}_{\mathsf{RV},i}]$ and $\mathsf{TX}_{\mathsf{RV},i}.\mathsf{Witness}[j] = (2, (\sigma^{A,j}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{B,j}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{W,j}_{\mathsf{TX}_{\mathsf{RV},i}}))$ with $j = \{1, 2\}$.

(c) Create $[\mathsf{TX}_{\mathsf{PN}_1,i}]$ and $[\mathsf{TX}_{\mathsf{PN}_2,i}]$

i. Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{PN}_1,i}]$ and $[\mathsf{TX}_{\mathsf{PN}_2,i}]$ according to 5.7 and 5.8.

ii. Party P computes $\sigma^{P,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 2))$ and $\sigma^{P,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 2))$, respectively.

iii. $P \xleftrightarrow{\sigma^{P,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, \sigma^{P,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}} \bar{P}$

iv. If P receives signatures $\sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}$ and $\sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 2); \sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}) = 1$ and $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 2); \sigma^{\bar{P},2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}) = 1$, it continues. Else it executes the non-collaborative closure phase (The channel now is at state m) or gets online at least once every $t - 1$ blocks.

v. W creates $[\mathsf{TX}_{\mathsf{PN}_1,i}]$ and $[\mathsf{TX}_{\mathsf{PN}_2,i}]$ according to 5.7 and 5.8

vi. W computes $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}} = \mathsf{Sign}_{sk_W}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 2)$ and $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}} = \mathsf{Sign}_{sk_W}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 2)$.

vii. $W \xhookrightarrow{\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}} P$ with $P = \{A, B\}$.

viii. If $P \in \{A, B\}$ receives signatures $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}$ and $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}$ from W s.t. $\mathsf{Vrfy}_{pk_W}(f([\mathsf{TX}_{\mathsf{PN}_1,i}], 2); \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_1,i}}) = 1$ and $\mathsf{Vrfy}_{pk_W}(f([\mathsf{TX}_{\mathsf{PN}_2,i}], 2); \sigma^{W,2}_{\mathsf{TX}_{\mathsf{PN}_2,i}}) = 1$ it continues. Else it executes the non-collaborative closure phase (The channel now is at state m) or gets online at least once every $t - 1$ blocks.

2. Create $[\mathsf{TX}_{\mathsf{RV},m}]$:

   (a) W creates $[\mathsf{TX}_{\mathsf{RV},m}]$ according to 5.6.

   (b) W computes $\sigma^{W,j}_{\mathsf{TX}_{\mathsf{RV},m}} = \mathsf{Sign}_{sk_W}([f(\mathsf{TX}_{\mathsf{RV},m}], j))$ with $j = \{1, 2\}$.

   (c) $W \xhookrightarrow{\sigma^{W,1}_{\mathsf{TX}_{\mathsf{RV},m}}, \sigma^{W,2}_{\mathsf{TX}_{\mathsf{RV},m}}} A, B$.

   (d) If party $P \in \{A, B\}$ receives $\sigma^{W,1}_{\mathsf{TX}_{\mathsf{RV},m}}$ and $\sigma^{W,2}_{\mathsf{TX}_{\mathsf{RV},m}}$ from W s.t. $\mathsf{Vrfy}_{pk_W}(f([\mathsf{TX}_{\mathsf{RV},m}], j); \sigma^{W,j}_{\mathsf{TX}_{\mathsf{RV},m}}) = 1$ with $j = \{1, 2\}$, it continues. Else it stops.

The commit and revocation transactions introduced in 5.10 and 5.11 have an interesting property that allows the honest channel party to penalise both the cheating party and the unresponsive watchtower. Assume that there is a revoked commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ according to 5.10 for which the watchtower has also signed penalty transactions. If this transaction is published by a cheating party, let's say A, the watchtower can immediately publish $\mathsf{TX}_{\mathsf{RV},i}$ to invalidate the penalty transactions. However, if the watchtower is unresponsive, after t rounds, B can firstly publish $\mathsf{TX}_{\mathsf{RV}',i}$ to penalise the cheating party B and then publish $\mathsf{TX}_{\mathsf{PN}_1,i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$ to penalise the unresponsive watchtower. Split transaction $\mathsf{TX}_{\mathsf{SP},i}$ cannot be published within $2t - 1$ rounds since broadcast of $\mathsf{TX}_{\mathsf{CM},i}$.

5.9 FPPW Transactions Scripts

Bitcoin scripting is a fundamental component of the Bitcoin protocol that allows users to create and enforce conditions for spending bitcoins. It involves using a simple and stack-based scripting language to define the conditions that must be met to unlock the funds in a Bitcoin transaction output (UTXO). When bitcoins are sent to an address, the recipient specifies conditions in the form of a ScriptPubKey or simply the script. This script defines the spending conditions required to unlock and spend the funds. To spend

the bitcoins locked in a UTXO, the sender must provide a witness in the spending transaction. The witness is combined with the ScriptPubKey of the UTXO, and the resulting script is executed. The script execution is done by pushing data onto a stack and applying various script operations. If the final result of the script evaluation is true, the UTXO can be spent. The script operations include basic arithmetic, cryptographic operations, conditional statements (e.g. OP_IF and OP_ELSE), and signature checks (e.g. OP_CHECKMULTISIG) [59].

Funding transaction has one output with the following script where pubkeyA, pubkeyB are public keys of A and B, respectively:

2 ⟨pubkeyA⟩ ⟨pubkeyB⟩ 2 OP_CHECKMULTISIG

Commit transaction has one input that takes the output of the funding transaction with witness script 0 ⟨pubkeyA_sig⟩ ⟨pubkeyB_sig⟩. It also has two outputs where the script of its first output (main output) is as follows:

OP_IF

Revocation

3 ⟨Rev_pubkeyA⟩ ⟨Rev_pubkeyB⟩ ⟨Rev_pubkeyW⟩ 3 OP_CHECKMULTISIG

OP_ELSE

Split

⟨delay t⟩ OP_CHECKSEQUENCEVERIFY OP_DROP

2 ⟨Spl_pubkeyA⟩ ⟨Spl_pubkeyB⟩ 2

OP_ENDIF

where ⟨Rev_pubkeyA⟩ and ⟨Spl_pubkeyA⟩ are public keys of A, ⟨Rev_pubkeyB⟩ and ⟨Spl_pubkeyB⟩ are public keys of B and ⟨Rev_pubkeyW⟩ is public key of W.

The script for the second output (auxiliary output) of the commit transaction is as follows:

OP_IF

Revocation

3 ⟨Rev_pubkeyA⟩ ⟨Rev_pubkeyB⟩ ⟨Rev_pubkeyW⟩ 3 OP_CHECKMULTISIG

OP_ELSE

⟨delay t⟩ OP_CHECKSEQUENCEVERIFY OP_DROP

OP_IF

Penalty1 or Penalty2 by party B

2 ⟨Pen_pubkeyB⟩ ⟨YA⟩ 2 OP_CHECKMULTISIG

OP_ELSE

Penalty1 or Penalty2 by party A

2 ⟨Pen_pubkeyA⟩ ⟨YB⟩ 2 OP_CHECKMULTISIG

OP_ENDIF

OP_ENDIF

where ⟨Rev_pubkeyA⟩ and ⟨Pen_pubkeyA⟩ are public keys of A, ⟨Rev_pubkeyB⟩ and ⟨Pen_pubkeyB⟩ are public keys of B and ⟨Rev_pubkeyW⟩ is public key of W. Also, YA and YB are statement of A and B, respectively. The witness script for input of split transaction is 0 ⟨Spl_pubkeyA_Sig⟩ ⟨Spl_pubkeyB_Sig⟩ 0

The revocation transaction has two inputs where its first and second inputs take the first and second outputs of the corresponding commit transaction, respectively. The witness script for both inputs is 0 ⟨Rev_pubkeyA_sig⟩ ⟨Rev_pubkeyB_sig⟩ ⟨Rev_pubkeyW_sig⟩ 1. It also has one output with the following script:

2 ⟨YA⟩ ⟨YB⟩ 2 OP_CHECKMULTISIG

Collateral transaction has one output with script 3 ⟨pubkeyA⟩ ⟨pubkeyB⟩ ⟨pubkeyW⟩ 3 OP_CHECKMULTISIG, where pubkeyA, pubkeyB and pubkeyW are the public keys of A, B and W, respectively.

The reclaim transaction has one input taking the collateral transaction output with witness script:

0 ⟨pubkeyA_sig⟩ ⟨pubkeyB_sig⟩ ⟨pubkeyW_sig⟩

It also has a single output, with the following script:

OP_IF

Penalty2

3 ⟨Pen_pubkeyA⟩ ⟨Pen_pubkeyB⟩ ⟨Pen_pubkeyW⟩ 3 OP_CHECKMULTISIG

OP_ELSE

normal

⟨delay T⟩ OP_CHECKSEQUENCEVERIFY OP_DROP

⟨pubkeyW⟩ OP_CHECKSIG

OP_ENDIF

where Pen_pubkeyA, Pen_pubkeyB and Pen_pubkeyW are public keys of A, B and W, respectively. pubkeyW is also the public key of W.

Penalty transaction 1 has two inputs. The first one takes the second output of the corresponding commit transaction with the following witness script if B is broadcasting the penalty transaction:

0 ⟨Pen_pubkeyB_Sig⟩ ⟨YA_Sig⟩ 1 0

or with the following witness script, if A is broadcasting it:

0 ⟨Pen_pubkeyA_Sig⟩ ⟨YB_Sig⟩ 0 0

The second input takes the output of the collateral transaction with the witness script 0 ⟨pubkeyA_Sig⟩ ⟨pubkeyB_Sig⟩ ⟨pubkeyW_Sig⟩. Also, the script for its output is similar to that of the revocation transaction.

Penalty transaction 2 is similar to penalty transaction 1. The only difference is that its second input spends the output of the reclaim transaction. The witness for the second input is 0 ⟨Pen_pubkeyA_Sig⟩ ⟨Pen_pubkeyB_Sig⟩ ⟨Pen_pubkeyW_Sig⟩

5.10 FPPW with One Hiring Party

If only one of the channel parties is willing to hire the watchtower, some changes must be applied to FPPW. The transaction flows for FPPW in such scenarios are depicted in Fig. 5.6. In this scenario, without loss of generality, we assume that party A is the hiring party and hence only A funds the extra ϵ. Also, the auxiliary output of the commit transaction has only two sub-conditions where the first one is used by A for penalty purposes and the second one can be used by both parties for revocation purposes. Output condition for $\mathsf{TX}_{\mathsf{CL}}$ as well as the first sub-condition for output of $\mathsf{TX}_{\mathsf{RC}}$ is $pk_A \wedge pk_W$ and public key of party B is not involved in these transactions anymore. Furthermore, output of $\mathsf{TX}_{\mathsf{PN}_1,i}$ and $\mathsf{TX}_{\mathsf{PN}_2,i}$ can be only claimed by party A. Therefore, If party B publishes a revoked commit transaction, W can publish its corresponding revocation transaction and then only A can claim its output. Otherwise, A can penalise the watchtower by publishing either $\mathsf{TX}_{\mathsf{PN}_1,i}$ or $\mathsf{TX}_{\mathsf{PN}_2,i}$. Similarly, if A publishes a revoked commit transaction, B can broadcast its corresponding revocation transaction and claim all the channel funds. The watchtower is only paid by A.

5.11 Conclusion

In this chapter, we presented a new payment channel with a watchtower for Bitcoin, called FPPW, which achieves fairness with respect to both the channel party and the watchtower as well as weak privacy against the watchtower. FPPW is the same as Cerberus with respect to the watchtower privacy against third parties meaning that it can

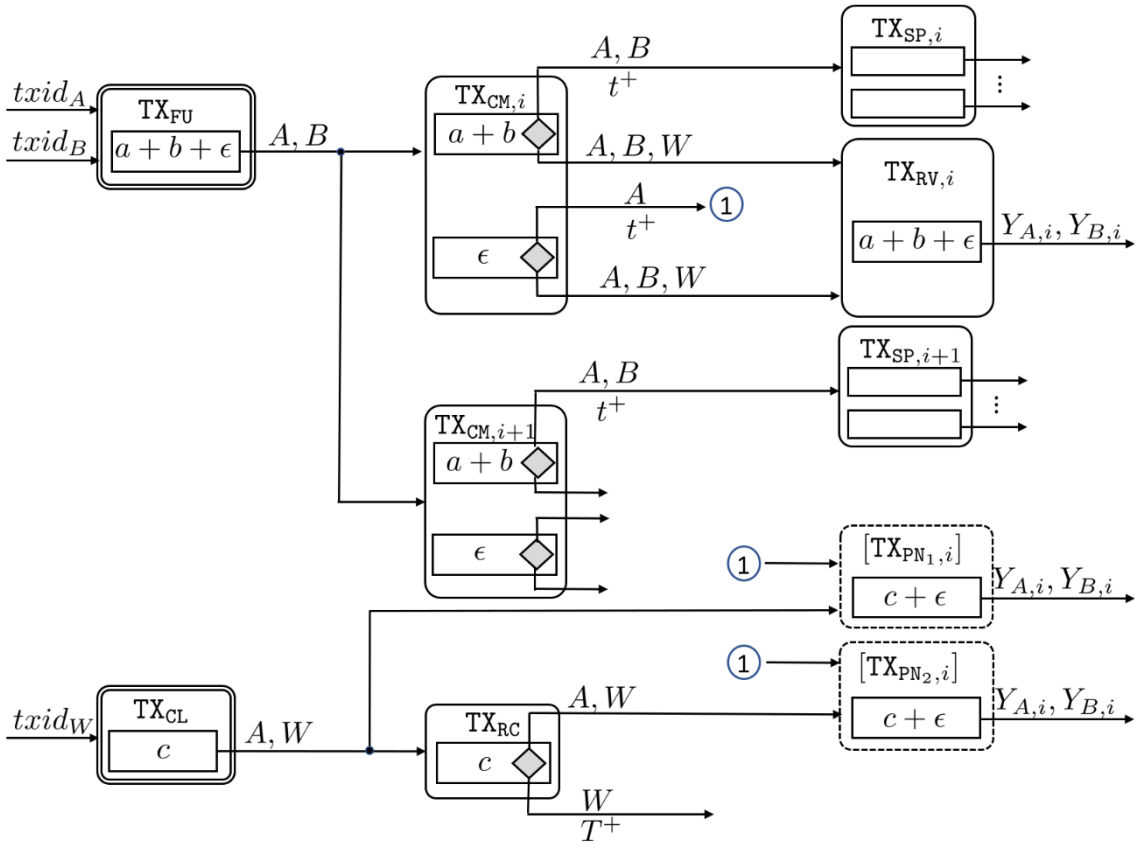

Figure 5.6: An FPPW Bitcoin Channel with only A Being the Hiring Party.

potentially achieve weak privacy against third parties (See Section 4.3.2 for more details). FPPW provides β-coverage with $\beta = \frac{1}{2}$, which is higher than Cerberus and PISA with the same channel party fairness. Although unlike Cerberus, the watchtower contract for FPPW can start and terminate at any time, the watchtower identity should be specified at the channel creation phase and hence the channel party cannot change the watchtower later. So, according to the Definition 4.5, FPPW cannot achieve agility.

Chapter 6

Garrison: a storage efficient Bitcoin watchtower

We construct this chapter based on our published paper, "Garrison: a novel watchtower scheme for bitcoin" [60].

6.1 Introduction

In the previous chapter, we mostly focused on the privacy and fairness of a watchtower scheme and presented a fair and privacy-preserving watchtower scheme, called FPPW. However, in this chapter, our attention shifts towards evaluating the storage costs incurred by both the watchtower and channel parties. These costs play a crucial role in determining the operational expenses associated with running services such as hubs in PCHs or watchtower services. The storage size of the watchtower in Monitor [15], DCWC* [32] as well as Generalized [14], Cerberus [23] and FPPW [54] channels increases linearly with each channel update and hence the watchtower's storage costs would be $\mathcal{O}(n)$ with n being the number of channel updates.

Outpost [22] is a novel payment channel with a watchtower scheme that reduces the watchtower's storage requirements per channel from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$. This consequently reduces the operational costs of maintaining watchtowers. Although elegantly designed, Outpost suffers from the following shortcomings,

- The storage cost of each channel party is still $\mathcal{O}(n)$.
- Each party has his own version of the channel state where this state duplication causes the number of transactions to exponentially increase with the number of applications on top of each other [14]. In other words, to add an application (e.g.

Table 6.1: Comparison of different dispute period-based payment channels with n channel updates, m HTLC outputs on average per state and k channel splits on top of each other.

Scheme	Party's St. Cost	Watch. St. Cost	on-chain TX.[a]	off-chain TX.[b]
Lightning [21]	$\mathcal{O}(\log n)$	$\mathcal{O}(N)$	$\mathcal{O}(m)$	$\mathcal{O}(2^k)$
Generalized [32]	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Outpost [22]	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(m)$	$\mathcal{O}(2^k)$
FPPW [54]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Cerberus [23]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(m)$	$\mathcal{O}(2^k)$
Garrison	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

[a]Number of on-chain transactions upon dispute.
[b]Number of off-chain transactions per state.

Virtual channel [27]) on top of the channel, parties must split their channel into sub-channels. If parties recursively split their channel k times, then to update their last layer sub-channel, they must create $\mathcal{O}(2^k)$ different versions of the channel state.

- Outpost works based on "punish-per-output" pattern, meaning that if there are m outputs in the published old state, the cheated party must claim each output separately [14]. Then, the required on-chain transactions upon dispute would be $\mathcal{O}(m)$ with m being the number of outputs in the published old state.

Therefore, the main motivation of this chapter is designing a Bitcoin payment channel with a watchtower scheme which is storage-efficient for channel parties and the watchtower and also avoids state duplication and punish-per-output pattern.

The contribution of this chapter is to present a new payment channel with a watchtower for Bitcoin, called Garrison, for which the storage cost of channel parties and the watchtower would be logarithmic in the maximum number of channel updates. Furthermore, both channel parties store the same version of transactions. Additionally, regardless of the number of outputs in each channel state, there exists a single revocation transaction per state. We also prove the security of the Garrison channel based on the security of its underlying cryptographic primitives. Table 6.1 presents a comparison between Garrison and other Bitcoin payment channels that work based on dispute period.

6.2 Notations

In this section, we add a new notation to the existing notations introduced in Section 3.3. We already know that a Bitcoin output θ is a tuple of two attributes, $\theta = (\text{cash}, \varphi)$, where

θ.cash denotes the number of coins held in this output. However, the OP_RETURN output is a special output that does not hold any coins and is used to add some arbitrary data to the blockchain. Such an output is denoted by $\theta = (0, data)$, where *data* is its arbitrary data. OP_RETURN outputs are illustrated by blocked lines (instead of directional arrows) in charts. As an example, the second output in the transaction TX in Fig. 6.1 is an OP_RETURN output.

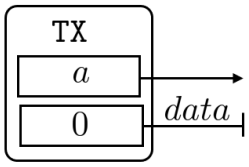

Figure 6.1: A sample transaction flow.

6.3 Garrison Overview

6.3.1 System Model

Channel parties exchange data using an authenticated and secure communication channel. Channel participants might deviate from the protocol if it increases their profit. Furthermore, the underlying blockchain contains a distributed ledger that achieves security [56]. If a valid transaction is propagated in the blockchain network, it is included in the blockchain ledger within Δ rounds (i.e. the confirmation delay is Δ).

6.3.2 Garrison Overview

This section overviews the Garrison channel between A (Alice) and B (Bob). To introduce Garrison, we use the payment channel NVG, introduced in Section 5.3.2.1, as the starting point and modify it step by step to mitigate its limitations.

6.3.2.1 Reducing the Storage Requirements of the Watchtower

One of the limitations of NVG channel is that all revocation transactions must be stored by channel parties or their watchtowers to be published upon fraud. To reduce the storage requirements of the watchtower, similar to Outpost [22], our main idea is to storing the revocation transaction $\mathsf{TX}_{\mathsf{RV},i}$ inside the commit transaction $\mathsf{TX}_{\mathsf{CM},i}$. Then, once $\mathsf{TX}_{\mathsf{CM},i}$ is published, the watchtower extracts $\mathsf{TX}_{\mathsf{RV},i}$ and records it on the blockchain. However, we have $\mathsf{TX}_{\mathsf{RV},i}.\mathsf{Input} = \mathsf{TX}_{\mathsf{CM},i}.\mathsf{txid} \| 1$. Thus, if $\mathsf{TX}_{\mathsf{RV},i}$ is created, signed and finally stored

inside $\mathsf{TX}_{\mathsf{CM},i}$, then $[\mathsf{TX}_{\mathsf{CM},i}]$ and hence $\mathsf{TX}_{\mathsf{CM},i}$.txid and $\mathsf{TX}_{\mathsf{RV},i}$ change. Thus, there is a self-loop situation [22]. To solve this issue, we add an auxiliary output with the value of ϵ to commit transactions where ϵ is the minimum value supported by the Bitcoin blockchain. We also add an auxiliary transaction between each commit transaction and its corresponding split transaction. This new transaction $\mathsf{TX}_{\mathsf{AU},i}$ spends the auxiliary output of the commit transaction. The signatures of party A and party B on $[\mathsf{TX}_{\mathsf{RV},i}]$ are stored in an OP_RETURN output of the auxiliary transaction $\mathsf{TX}_{\mathsf{AU},i}$. The split transaction $\mathsf{TX}_{\mathsf{SP},i}$ spends the main output of $\mathsf{TX}_{\mathsf{CM},i}$ as well as the main output of the auxiliary transaction $\mathsf{TX}_{\mathsf{AU},i}$. Based on this design, parties can be sure that once the revoked commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ is published on the blockchain, its split transaction $\mathsf{TX}_{\mathsf{SP},i}$ cannot be published unless $\mathsf{TX}_{\mathsf{AU},i}$ is also on the blockchain. Furthermore, due to the time-lock in the main output of $\mathsf{TX}_{\mathsf{AU},i}$, once this transaction is published on-chain, $\mathsf{TX}_{\mathsf{SP},i}$ cannot be published within $t-1$ rounds. However, the honest party or the watchtower can extract the signatures on $[\mathsf{TX}_{\mathsf{RV},i}]$ from $\mathsf{TX}_{\mathsf{AU},i}$ and publish $\mathsf{TX}_{\mathsf{RV},i}$ immediately. Fig. 6.2 depicts the transaction flows.

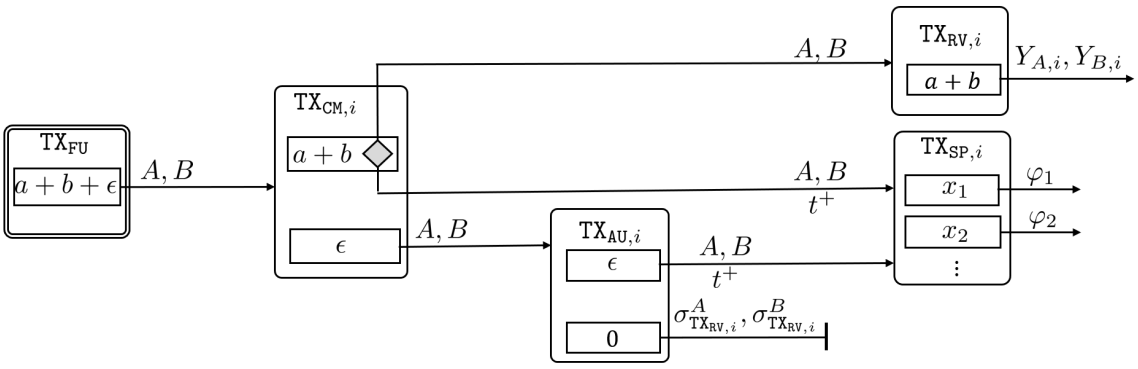

Figure 6.2: Reducing the Storage Requirements of the Watchtower

However, this scheme has the following issues:

- To create and publish the revocation transaction, the watchtower must also know the value of $Y_{A,i}$ and $Y_{B,i}$.
- Typically, revocation transaction of state i must be created once parties update the channel state from state i to $i+1$. However, in the proposed scheme signatures for $\mathsf{TX}_{\mathsf{RV},i}$ is stored in $\mathsf{TX}_{\mathsf{AU},i}$ and hence $\mathsf{TX}_{\mathsf{RV},i}$ must actually be created once parties update the channel state from state $i-1$ to i. It means if an honest party records the latest commit and auxiliary transactions on the blockchain, the counterparty might publish the revocation transaction and take all the channel funds.

To solve the first mentioned issue, $Y_{A,i}$ and $Y_{B,i}$ are stored in an OP_RETURN output that is added to the commit transaction $\mathsf{TX}_{\mathsf{CM},i}$. To solve the second mentioned issue, we add

two statements from the hard relation $\mathscr{R}$, $R_{A,i}$ and $R_{B,i}$, to the first sub-condition of the main output of $\mathsf{TX}_{\mathsf{CM},i}$, where $R_{A,i}$ ($R_{B,i}$) is generated by A (B) for the state i. Then, once the latest commit and auxiliary transactions are published by A, party B cannot record the revocation transaction as he does not know his counterparty's witness $r_{A,i}$. The witnesses $r_{A,i}$ and $r_{B,i}$ are exchanged between the parties and are given to the watchtower once parties have created $\mathsf{TX}_{\mathsf{CM},i+1}$, $\mathsf{TX}_{\mathsf{AU},i+1}$ and $\mathsf{TX}_{\mathsf{SP},i+1}$. Thus, $\mathsf{TX}_{\mathsf{FU}}$.txid, public keys pk_A and pk_B as well as r values of both parties are all data needed by the watchtower to watch the channel for both parties. Fig. 6.3 depicts the mentioned modifications.

The security requirement for r values is that B (or the watchtower) must not be able to compute $r_{A,j}$ given that he knows $r_{A,i}$ with $i < j$. Otherwise, when A submits the latest commit transaction $\mathsf{TX}_{\mathsf{CM},j}$, party B uses $r_{A,i}$ to compute $r_{A,j}$. Then, B publishes the revocation transaction $\mathsf{TX}_{\mathsf{RV},j}$ and claims its output. If r values are randomly generated, the mentioned security requirement is met but the storage cost of channel parties and the watchtower would be $\mathcal{O}(n)$. To reduce the storage and meet the stated security requirement, parties generate their r values in a binary Merkle tree and use them from the deepest leaf nodes in the tree to the root [61]. In more detail, in a binary Merkle tree, each node has two child nodes where having the value of a node, the value of each of its child nodes can simply be computed using a one-way function. But deriving the value of a node from its child nodes' values is computationally infeasible. Thus, since r values are used from the deepest leaf nodes in the tree, the stated security requirement is achieved. Moreover, the storage needed by each channel party (or the watchtower) to store r values, received from her counterparty, will be $\mathcal{O}(\log n)$ because upon receipt of a node value, its child nodes' values can be removed from the storage.

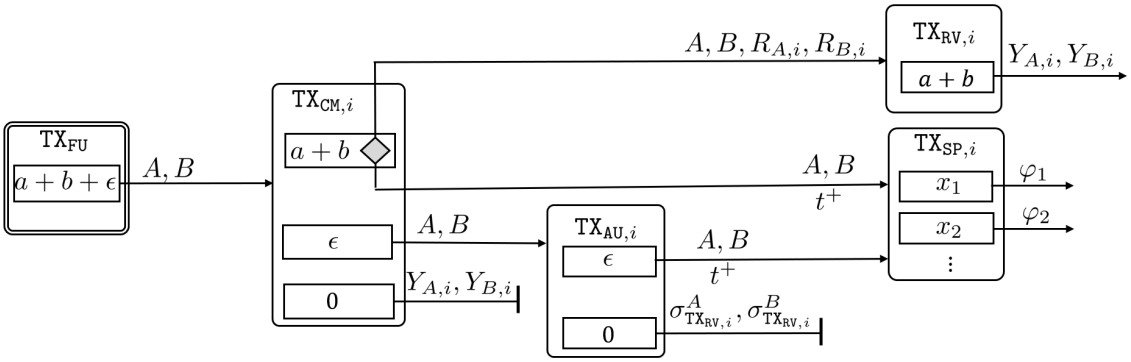

Figure 6.3: Adding Y and R Values to Commit Transactions

6.3.2.2 Reducing the Storage Requirements of channel parties

Although the storage of the watchtower is $\mathcal{O}(\log n)$, channel parties still have to store all the signatures of their counterparties on the revocation transactions. Otherwise, the dishonest channel party publishes a revoked commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ without publishing its auxiliary transaction $\mathsf{TX}_{\mathsf{AU},i}$. Then, the channel funds could be locked forever. This raises a hostage situation. The scheme Outpost suffers from this problem which is why the storage requirement of channel parties is $\mathcal{O}(n)$. To solve this problem, we add one sub-condition, $Y_{A,i} \wedge Y_{B,i} \wedge 3t^{+}$, to the main output of the commit transaction $\mathsf{TX}_{\mathsf{CM},i}$. This sub-condition allows the honest channel party to claim all the channel funds in such hostage situations. In other words, if party A publishes the revoked commit transaction $\mathsf{TX}_{\mathsf{CM},i}$, she has $3T$ rounds time to publish $\mathsf{TX}_{\mathsf{AU},i}$ and $\mathsf{TX}_{\mathsf{SP},i}$ before B can claim the channel funds by meeting the sub-condition $Y_{A,i} \wedge Y_{B,i} \wedge 3t^{+}$. If during this interval, $\mathsf{TX}_{\mathsf{AU},i}$ is published, party B instantly establishes and publishes $\mathsf{TX}_{\mathsf{RV},i}$ and claims its output. To do so, each party must have r values of both parties stored. Since these keys are generated in a Merkle tree, the storage requirements of each channel party for storing these values would be $\mathcal{O}(\log n)$ (See Fig. 6.4).

Once party A publishes $\mathsf{TX}_{\mathsf{CM},i}$, party B must be able to use Ext algorithm to extract the value of $y_{A,i}$. To do so, he must know the corresponding pre-signature $\tilde{\sigma}^{B}_{\mathsf{TX}_{\mathsf{CM},i}}$. If parties store all their own pre-signatures, their storage cost would be $\mathcal{O}(n)$. To acquire lower storage costs, parties must be able to regenerate the required pre-signature, once a commit transaction is published. To achieve this goal, random values which are required to generate pre-signatures must be generated in a Merkle tree and be used from the root to the deepest leaf node in the tree. In this way, once the commit transaction $\mathsf{TX}_{\mathsf{CM},i}$ is published by A, party B can regenerate the required random value, recompute the corresponding pre-signature $\tilde{\sigma}^{B}_{\mathsf{TX}_{\mathsf{CM},i}}$ and finally extract the value of $y_{A,i}$. Thus, the storage requirements would be still $\mathcal{O}(\log n)$.

Additionally, party B must know the value of $y_{B,i}$ to meet $Y_{A,i} \wedge Y_{B,i}$. The security requirement for y values is that A must not be able to compute $y_{B,i}$ given that he knows $y_{B,j}$ with $j > i$. Otherwise, once B submits the latest commit transaction $\mathsf{TX}_{\mathsf{CM},j}$, A computes $y_{B,j}$ and hence derives $y_{B,i}$ with $i < j$ and then try to publish $\mathsf{TX}_{\mathsf{CM},i}$ before $\mathsf{TX}_{\mathsf{CM},j}$ being published on the ledger. Then, A might be able to claim all the channel funds by meeting the third sub-condition of the main output of $\mathsf{TX}_{\mathsf{CM},i}$ or by publishing the revocation transaction $\mathsf{TX}_{\mathsf{RV},i}$ and claiming its output. If y values are randomly generated, the mentioned security requirement is met but the parties' storage cost would be $\mathcal{O}(n)$. To reduce the storage and simultaneously meet the stated security requirement, parties generate their y values in a Merkle tree and give the corresponding Y values to their counterparties from the root to the deepest leaf nodes in the tree.

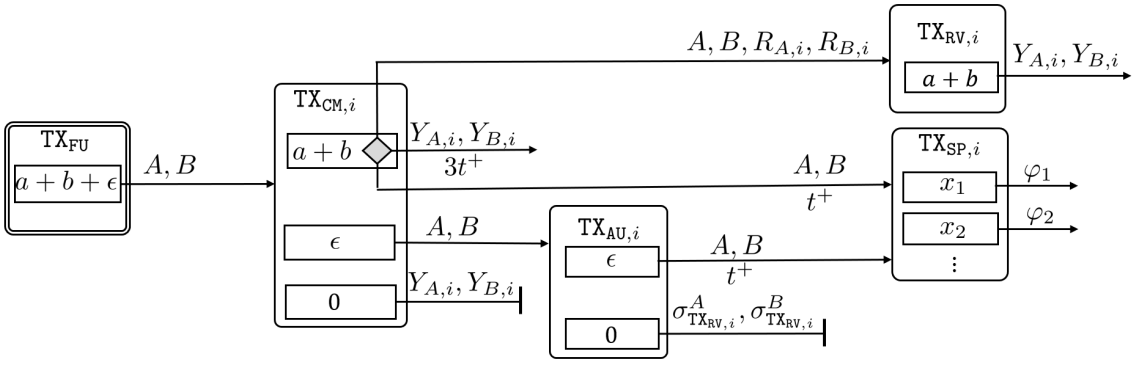

Figure 6.4: Reducing Storage Requirements of Channel Parties

6.4 Garrison Protocol Description

The lifetime of a Garrison channel can be divided into 4 phases including *create*, *update*, *close*, and *punish*. These phases will be explained in the following.

6.4.1 Create

The channel creation phase completes once the funding transaction $\mathtt{TX}_{\mathtt{FU}}$, the commit transactions $\mathtt{TX}_{\mathtt{CM},0}$, the split transaction $\mathtt{TX}_{\mathtt{SP},0}$, the auxiliary transaction $\mathtt{TX}_{\mathtt{AU},0}$ and body of the revocation transaction $[\mathtt{TX}_{\mathtt{RV},0}]$ are created, and $\mathtt{TX}_{\mathtt{FU}}$ is published on the blockchain. In this phase, parties do not have access to $\mathtt{TX}_{\mathtt{RV},0}$ as they have not exchanged $r_{A,0}$ and $r_{B,0}$ yet. At the end of the channel creation phase, the channel would be at state 0. Since the output of the funding transaction can only be spent if both parties agree, one party might become unresponsive to raise a hostage situation. To avoid this, parties must sign the commit, revocation, auxiliary and split transactions before signing and publishing the funding transaction. Fig. 6.5 summarises the channel creation phase.

The introduced transactions will be explained further below:

- **Funding transaction** Given that A (B, respectively) uses the x^{th} (y^{th}, respectively) output of a transaction with transaction identifier of $txid_A$ ($txid_B$, respectively) to fund the channel with a (b, respectively) coins, the funding transaction

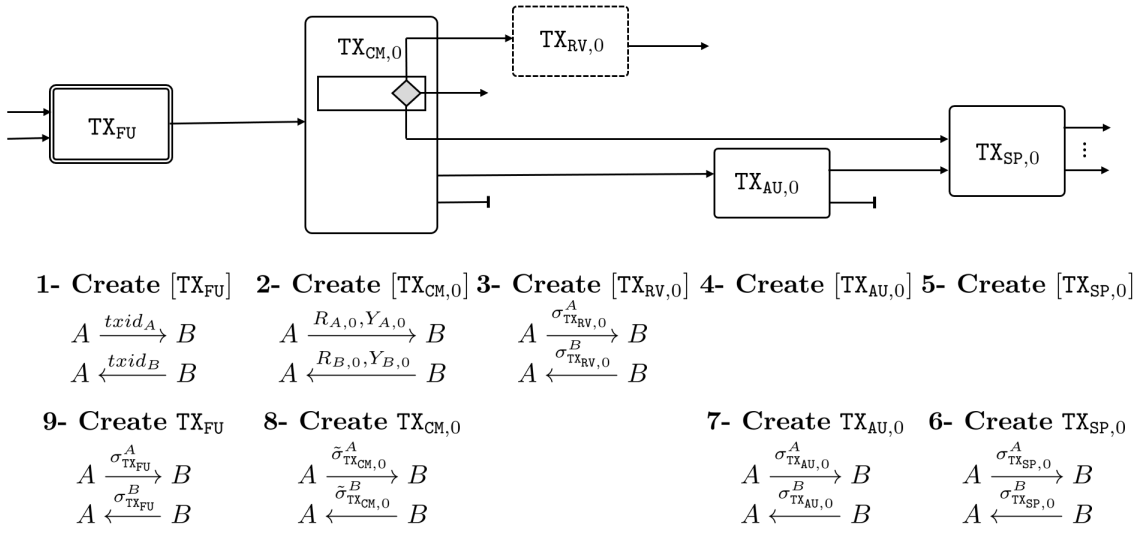

Figure 6.5: Summary of Garrison channel creation phase.

is as follows[1]:

$$\begin{aligned}
\mathsf{TX}_{\mathsf{FU}}.\mathsf{Input} &:= (txid_A \| x, txid_B \| y),\\
\mathsf{TX}_{\mathsf{FU}}.\mathsf{Output} &:= \{(a+b, pk_A \wedge pk_B)\},\\
\mathsf{TX}_{\mathsf{FU}}.\mathsf{Witness} &:= ((1, \sigma^{A,1}_{\mathsf{TX}_{\mathsf{FU}}}), (1, \sigma^{B,2}_{\mathsf{TX}_{\mathsf{FU}}})).
\end{aligned} \tag{6.1}$$

Commit transaction There exists one commit transaction per state but only the first one ($\mathsf{TX}_{\mathsf{CM},i}$ with $i = 0$) is created at the channel creation phase.

$$\begin{aligned}
\mathsf{TX}_{\mathsf{CM},i}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid} \| 1,\\
\mathsf{TX}_{\mathsf{CM},i}.\mathsf{Output} &:= ((a+b, \varphi_1 \vee \varphi_2 \vee \varphi_3),\\
&\qquad\quad (\epsilon, pk_A \wedge pk_B),\\
&\qquad\quad (0, (Y_{A,i}, Y_{B,i})))\\
\mathsf{TX}_{\mathsf{CM},i}.\mathsf{Witness} &:= \{(1, \{\sigma^A_{\mathsf{TX}_{\mathsf{CM},i}}, \sigma^B_{\mathsf{TX}_{\mathsf{CM},i}}\})\}
\end{aligned} \tag{6.2}$$

with $\varphi_1 := (pk_A \wedge pk_B \wedge R_{A,i} \wedge R_{B,i})$, $\varphi_2 := (Y_{A,i} \wedge Y_{B,i} \wedge 3t^+)$, and $\varphi_3 := (pk_A \wedge pk_B \wedge t^+)$ where $Y_{A,i}$ and $R_{A,i}$ ($Y_{B,i}$ and $R_{B,i}$) are statements of a hard relation $\mathscr{R}$ generated by A (B) for the i^{th} state and t is any number such that $t > \Delta$. The first and second outputs of the transaction are the main and auxiliary outputs. Normally, if $\mathsf{TX}_{\mathsf{CM},i}$ is the last commit transaction and is published on-chain, first its auxiliary output

[1]We assume that funding sources of $\mathsf{TX}_{\mathsf{FU}}$ are two typical UTXOs owned by A and B.

and then its main output is spent by the auxiliary and split transactions, respectively. The third output of $\mathsf{TX}_{\mathsf{CM},i}$ is an OP_RETURN output containing values of $Y_{A,i}$ and $Y_{B,i}$. Parties A and B use their counterparties' statements $Y_{B,i}$ and $Y_{A,i}$ and the underlying adaptor signature to generate a pre-signature on the commit transaction for their counterparties. Thus, once A publishes the commit transaction $\mathsf{TX}_{\mathsf{CM},i}$, she also reveals her witness $y_{B,i}$.

Remark 6.1. *Each Bitcoin transaction can have at most one OP_RETURN output with the size constraint of 80 bytes. To store $Y_{A,i}$ and $Y_{B,i}$ inside an OP_RETURN output, their compressed version, each with a 33-byte length, are stored.*

- **Revocation transaction** The revocation transaction for state i is denoted by $\mathsf{TX}_{\mathsf{RV},i}$ and is as follows, where at the channel creation phase only $\mathsf{TX}_{\mathsf{RV},i}$ with $i = 0$ is created:

$$\begin{aligned} \mathsf{TX}_{\mathsf{RV},i}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{CM},i}.\mathsf{txid} \| 1, \\ \mathsf{TX}_{\mathsf{RV},i}.\mathsf{Output} &:= \{(a+b, Y_{A,i} \wedge Y_{B,i})\}, \\ \mathsf{TX}_{\mathsf{RV},i}.\mathsf{Witness} &:= \{(1, \{\sigma^{A}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{B}_{\mathsf{TX}_{\mathsf{RV},i}}, r_{A,i}, r_{B,i}\})\} \end{aligned} \tag{6.3}$$

  The $\mathsf{TX}_{\mathsf{RV},i}$ spends the main output of $\mathsf{TX}_{\mathsf{CM},i}$ using its non-time-locked subcondition $pk_A \wedge pk_B \wedge R_{A,i} \wedge R_{B,i}$ and sends all the channel funds to an output with the condition $Y_{A,i} \wedge Y_{B,i}$. When a dishonest party, let's say A, publishes the revoked $\mathsf{TX}_{\mathsf{CM},i}$, she must publish $\mathsf{TX}_{\mathsf{AU},i}$ and then wait for t rounds before being able to publish $\mathsf{TX}_{\mathsf{SP},i}$. However, given that the state i is revoked, B knows the value of $r_{A,i}$ and hence creates the revocation transaction $\mathsf{TX}_{\mathsf{RV},i}$ and instantly publishes it on the blockchain. The output of $\mathsf{TX}_{\mathsf{RV},i}$ can only be claimed by B because no one else knows the witness $y_{B,i}$.

- **Auxiliary transaction** Auxiliary transaction for state i is as follows, where at the channel creation phase only $\mathsf{TX}_{\mathsf{AU},i}$ with $i = 0$ is created:

$$\begin{aligned} \mathsf{TX}_{\mathsf{AU},i}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{CM}}.\mathsf{txid} \| 2, \\ \mathsf{TX}_{\mathsf{AU},i}.\mathsf{Output} &:= ((\epsilon, pk_A \wedge pk_B \wedge t^{+}), \\ &\qquad (0, (\sigma^{A}_{\mathsf{TX}_{\mathsf{RV},i}}, \sigma^{B}_{\mathsf{TX}_{\mathsf{RV},i}}))) \\ \mathsf{TX}_{\mathsf{AU},i}.\mathsf{Witness} &:= \{(1, \{\sigma^{A}_{\mathsf{TX}_{\mathsf{AU},i}}, \sigma^{B}_{\mathsf{TX}_{\mathsf{AU},i}}\})\} \end{aligned} \tag{6.4}$$

  This transaction spends the auxiliary output of the commit transaction and its output is spent by the split transaction. In other words, the split transaction cannot

be published unless the auxiliary transaction is on the blockchain. The second output of $\mathsf{TX}_{\mathsf{AU},i}$ is an OP_RETURN output containing signatures of both parties on the corresponding revocation transaction.

Remark 6.2. *Each encoded Bitcoin signature can be up to 73 bytes long. Thus, due to the size constraint of the OP_RETURN output, two separate signatures do not fit into the auxiliary transaction. To solve this issue, A and B can aggregate their public keys pk_A and pk_B to form an aggregated public key pk [62] and change φ_1 in $\mathsf{TX}_{\mathsf{CM},i}$ to $(pk \wedge R_{A,i} \wedge R_{B,i})$. Then, rather than two separate signatures on the revocation transaction, they generate a multisignature (with up to 73-byte size) and store it inside the OP_RETURN output of $\mathsf{TX}_{\mathsf{AU},i}$.*

- **Split transaction**

  $\mathsf{TX}_{\mathsf{SP},i}$ actually represents the i^{th} channel state and is as follows, where at the channel creation phase only the first one, $\mathsf{TX}_{\mathsf{SP},i}$ with $i = 0$, is created:

$$\begin{aligned} \mathsf{TX}_{\mathsf{SP},i}.\mathsf{Input} &:= (\mathsf{TX}_{\mathsf{CM},i}.\mathsf{txid} \| 1, \mathsf{TX}_{\mathsf{AU},i}.\mathsf{txid} \| 1), \\ \mathsf{TX}_{\mathsf{SP},i}.\mathsf{Output} &:= (\theta_1, \theta_2, \cdots), \\ \mathsf{TX}_{\mathsf{SP},i}.\mathsf{Witness} &:= ((3, \{\sigma^A_{\mathsf{TX}_{\mathsf{SP},i}}, \sigma^B_{\mathsf{TX}_{\mathsf{SP},i}}\}), (1, \{\sigma^A_{\mathsf{TX}_{\mathsf{SP},i}}, \sigma^B_{\mathsf{TX}_{\mathsf{SP},i}}\})) \end{aligned} \tag{6.5}$$

  The split transaction spends the main output of the commit transaction and the first output of the auxiliary transaction.

6.4.2 Update

Let the channel be in state $i \geq 0$ and channel parties decide to update it to state $i+1$. The update process is performed in two sub-phases. In the first sub-phase, channel parties create $\mathsf{TX}_{\mathsf{CM},i+1}$, $\mathsf{TX}_{\mathsf{SP},i+1}$, $\mathsf{TX}_{\mathsf{AU},i+1}$, and $[\mathsf{TX}_{\mathsf{RV},i+1}]$ for the new state. In the second sub-phase, channel parties revoke the state i by exchanging $r_{A,i}$ and $r_{B,i}$ and giving these values to the watchtower. We assume that the watchtower is also paid after each channel update. Fig. 6.6 summarises the channel update phase.

6.4.3 Close

Assume that the channel parties A and B have updated their channel n times and then A and/or B decide to close it. They can close the channel cooperatively. To do so, A and

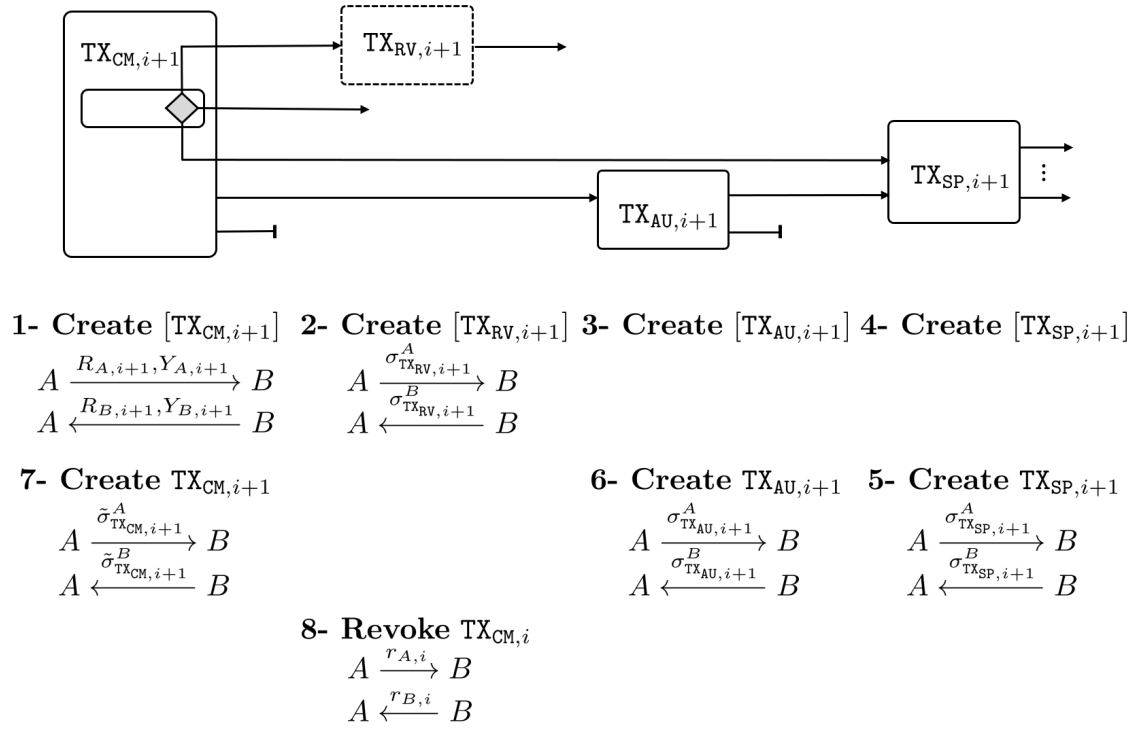

Figure 6.6: Summary of Garrison channel update phase from state i to $i+1$.

B create the below transaction, called modified split transaction $\mathsf{TX}_{\overline{\mathsf{SP}}}$, and publish it on the blockchain:

$$\begin{aligned} \mathsf{TX}_{\overline{\mathsf{SP}}}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid} \| 1, \\ \mathsf{TX}_{\overline{\mathsf{SP}}}.\mathsf{Output} &:= \mathsf{TX}_{\mathsf{SP},n}.\mathsf{Output}, \\ \mathsf{TX}_{\overline{\mathsf{SP}}}.\mathsf{Witness} &:= \{(1, \{\sigma^{A}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}, \sigma^{B}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}\})\}. \end{aligned} \tag{6.6}$$

If one of the channel parties, e.g. party B, becomes unresponsive, A can still non-collaboratively close the channel. To do so, she publishes $\mathsf{TX}_{\mathsf{CM},n}$ and $\mathsf{TX}_{\mathsf{AU},n}$ on the ledger. Then, she waits for t rounds, and finally publishes $\mathsf{TX}_{\mathsf{SP},n}$.

6.4.4 Punish

Let the channel be at state n. If a channel party, e.g. party A, publishes $\mathsf{TX}_{\mathsf{CM},i}$ and then $\mathsf{TX}_{\mathsf{AU},i}$ with $i < n$ on the blockchain, party B or his watchtower can create the transaction $\mathsf{TX}_{\mathsf{RV},i}$ and publish it within t rounds. If only $\mathsf{TX}_{\mathsf{CM},i}$ is published, party B claims its first output by meeting its second sub-condition $Y_{A,i} \wedge Y_{A,i} \wedge 3t^{+}$.

Remark 6.3. *If the watchtower is non-responsive, the channel might be closed in an old state. The paper [33] proposes a reputation system, called HashCashed, which forces watchtowers to be responsive without requiring them to lock any funds as collateral. Garrison might be used with the HashCashed system.*

6.5 Security Analysis

In this section, we prove that for the Garrison channel, it is of negligible probability that an honest party loses any funds.

Lemma 6.1. *Let* Π *be a* $\mathsf{EUF-CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. Then, for a Garrison channel with* n *channel updates, the broadcast of* $\mathsf{TX}_{\mathsf{RV},i}$ *with* $i < n$ *causes the honest channel party* $P \in \{A, B\}$ *to lose any funds in the channel with negligible probability.*

Proof. Without loss of generality let $P = A$. The transaction $\mathsf{TX}_{\mathsf{RV},i}$ with $i < n$ spends the main output of the revoked $\mathsf{TX}_{\mathsf{CM},i}$ and hence cannot be published unless $\mathsf{TX}_{\mathsf{CM},i}$ is on-chain. The transaction $\mathsf{TX}_{\mathsf{CM},i}$ spends the output of $\mathsf{TX}_{\mathsf{FU}}$. Since the condition in $\mathsf{TX}_{\mathsf{FU}}$.Output contains pk_A, this output cannot be spent without A's authorisation. Otherwise, the security of the underlying digital signature would be violated. Based on the protocol, the honest party A never broadcasts the revoked $\mathsf{TX}_{\mathsf{CM},i}$ on-chain and her pre-signature $\tilde{\sigma}_{\mathsf{TX}_{\mathsf{CM},i}}$ on the transaction $\mathsf{TX}_{\mathsf{CM},i}$ is the only authorisation he grants for spending $\mathsf{TX}_{\mathsf{FU}}$.Output using $\mathsf{TX}_{\mathsf{CM},i}$. Thus, if $\mathsf{TX}_{\mathsf{CM},i}$ is published, the probability that A fails to obtain $y_{B,i}$ is negligible. Otherwise, a$\mathsf{EUF-CMA}$ security or witness extractability of the used adaptor signature is violated. Furthermore, $\mathsf{TX}_{\mathsf{RV},i}$ has only one output with the condition of $Y_{A,i} \wedge Y_{B,i}$ and the value of $a + b$. Since A privately preserves its witness value $y_{A,i}$, the probability that any PPT adversary claims $\mathsf{TX}_{\mathsf{RV},i}$.Output is negligible. Otherwise, the utilised hard relation would break. Therefore, it is of negligible probability that A (who knows both $y_{A,i}$ and $y_{B,i}$) fails to claim $\mathsf{TX}_{\mathsf{RV},i}$.Output and obtain all the channel funds. □

Lemma 6.2. *Let* Π *be a* $\mathsf{EUF-CMA}$ *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. Then, for a Garrison channel between* A *and* B *with* $P \in \{A, B\}$ *being the honest party, if* P*'s counterparty publishes* $\mathsf{TX}_{\mathsf{CM},i}$*, it is with negligible probability that*

- P *fails to obtain the data required to meet the second sub-condition of* $\mathsf{TX}_{\mathsf{CM},i}$.Output[1].$\varphi$.
- *any PPT adversary can meet the second sub-condition of* $\mathsf{TX}_{\mathsf{CM},i}$.Output[1].$\varphi$.

Proof. Without loss of generality let $P = A$. Similar to the proof of Lemma 6.1, if B publishes $\mathsf{TX}_{\mathsf{CM},i}$, the probability that A fails to obtain $y_{B,i}$ is negligible. Otherwise, aEUF − CMA security or witness extractability of the used adaptor signature is violated. The witness $y_{A,i}$ has also been created by A and hence he has the whole data required to meet $Y_{A,i} \wedge Y_{B,i} \wedge 3t^+$. Furthermore, given that A privately preserves its witness value $y_{A,i}$, the probability that any PPT adversary meets this sub-condition is negligible. Otherwise, the utilised hard relation would break. □

Lemma 6.3. *Let* Π *be a* EUF − CMA *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. Then, for a Garrison channel with* n *channel updates, if the honest party* $P \in \{A, B\}$ *publishes* $\mathsf{TX}_{\mathsf{CM},n}$*,* P *loses funds in the channel with negligible probability.*

Proof. Without loss of generality let $P = A$. We assume that A publishes $\mathsf{TX}_{\mathsf{CM},n}$ in the block $\mathscr{B}_j$ of the blockchain and prove that it is of negligible probability that A fails to publish $\mathsf{TX}_{\mathsf{SP},n}$. Then, since $\mathsf{TX}_{\mathsf{SP},n}$ corresponds with the latest channel state, its broadcast cannot cause A to lose any funds.

The condition $\mathsf{TX}_{\mathsf{CM},n}.\mathsf{Output}[2].\varphi$ contains pk_A and hence it is of negligible probability that this output is spent without A's authorisation. Otherwise, the security of the underlying digital signature is violated. The honest party A grants such an authorisation only on the transaction $\mathsf{TX}_{\mathsf{AU},n}$ which is held by both A and B. Based on the protocol, once $\mathsf{TX}_{\mathsf{CM},n}$ is published on the blockchain by A, he also instantly submits $\mathsf{TX}_{\mathsf{AU},n}$ to the blockchain. According to our assumptions regarding the blockchain, $\mathsf{TX}_{\mathsf{AU},n}$ is published on the blockchain in the block $\mathscr{B}_{j+k}$ with $0 < k \leq \Delta < t$. Similarly, the first output of $\mathsf{TX}_{\mathsf{AU},n}$ can only be spent by $\mathsf{TX}_{\mathsf{SP},n}$. According to the protocol, A holds $\mathsf{TX}_{\mathsf{SP},n}$ and submits it to the blockchain t rounds after $\mathsf{TX}_{\mathsf{AU},n}$ is published on-chain. Thus, given that the first input of $\mathsf{TX}_{\mathsf{SP},n}$ (or equivalently the first output of $\mathsf{TX}_{\mathsf{CM},n}$) is still unspent, based on our assumptions regarding the blockchain, $\mathsf{TX}_{\mathsf{SP},n}$ is published on the blockchain in the block $\mathscr{B}_{j+k+l+t}$ with $0 < l \leq \Delta < t$. Now, we prove that, when $\mathscr{B}_{j+k+l+t}$ with $0 < l, k < t$ is added to the blockchain, the first output of $\mathsf{TX}_{\mathsf{CM},n}$, $\mathsf{TX}_{\mathsf{CM},n}.\mathsf{Output}[1]$, is still unspent.

The first and third sub-conditions of $\mathsf{TX}_{\mathsf{CM},n}.\mathsf{Output}[1]$ contains $R_{A,n}$ and pk_A, respectively and hence it is of negligible probability that these two sub-conditions are met without A's authorisation. Otherwise, the underlying hard relation or digital signature would break. Party A grants such an authorisation only on $\mathsf{TX}_{\mathsf{SP},n}$. Moreover, the second sub-condition $Y_{A,i} \wedge Y_{B,i} \wedge 3t^+$ cannot be met in block $\mathscr{B}_{j+k+l+t}$ with $0 < l, k < t$ because $j + k + l + t < j + 3t$. □

Lemma 6.4. *Let* Π *be a* EUF − CMA *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. Then, for a Garrison channel with* n *channel updates*

and with $P \in \{A, B\}$ being the honest party, if P's counterparty publishes $\mathsf{TX}_{\mathsf{CM},n}$, it is of negligible probability that P loses any funds in the channel.

Proof. Without loss of generality let $P = A$. The proof is similar to the proof of Lemma 6.3. The only difference is that following Lemma 6.2, it is of negligible probability that A fails to meet the second sub-condition of $\mathsf{TX}_{\mathsf{CM},n}$.Output[1]. Therefore, A can either publishes both $\mathsf{TX}_{\mathsf{AU},n}$ and $\mathsf{TX}_{\mathsf{SP},n}$ or claim $\mathsf{TX}_{\mathsf{CM},n}$.Output[1] by meeting its second sub-condition. None of these two cases can cause the honest party A to lose any funds in the channel. □

Lemma 6.5. *Let Π be a $\mathsf{EUF-CMA}$ secure digital signature, $\mathscr{R}$ be a hard relation and Ξ be a secure adaptor digital signature. Then, for a Garrison channel with n channel updates and with $P \in \{A, B\}$ being the honest party, if any adversary publishes $\mathsf{TX}_{\mathsf{CM},i}$ with $i < n$, it is of negligible probability that P loses any funds in the channel.*

Proof. Without loss of generality let $P = A$. The output $\mathsf{TX}_{\mathsf{CM},i}$.Output[1] includes 3 sub-conditions, one of which must be met to cheat A out of its funds. The first sub-condition contains pk_A and hence it is of negligible probability that this output is spent without A's authorisation. Otherwise, the security of the used digital signature is violated. The honest party A grants such an authorisation only on the transaction $\mathsf{TX}_{\mathsf{RV},i}$. However, according to Lemma 6.1, it is of negligible probability that broadcast of $\mathsf{TX}_{\mathsf{RV},i}$ causes A to lose any funds. Moreover, according to Lemma 6.2, it is of negligible probability that any PPT adversary can meet the second sub-condition. Now, we prove that if the third sub-condition is used to cheat A out of her funds, it leads to a contradiction.

Assume that the third sub-condition of $\mathsf{TX}_{\mathsf{CM},i}$.Output[1] is used to cheat A out of her funds. This sub-condition contains pk_A and hence it is of negligible probability that this condition is met without A's authorisation. Otherwise, the security of the underlying digital signature is violated. The honest party A grants such an authorisation only on the transaction $\mathsf{TX}_{\mathsf{SP},i}$. Assume that $\mathsf{TX}_{\mathsf{SP},i}$ is included in the block $\mathscr{B}_k$ of the blockchain. The transaction $\mathsf{TX}_{\mathsf{SP},i}$ cannot be added to the blockchain unless its inputs are some unspent outputs on the blockchain. It means that $\mathsf{TX}_{\mathsf{AU},i}$ is also on the blockchain and following the condition in $\mathsf{TX}_{\mathsf{AU},i}$.Output[1], the transaction $\mathsf{TX}_{\mathsf{AU},i}$ must have been published in the block $\mathscr{B}_j$ with $j \leq k - t$. However, based on the protocol, once A or her watchtower observes $\mathsf{TX}_{\mathsf{AU},i}$ on the blockchain, they create the corresponding revocation transaction $\mathsf{TX}_{\mathsf{RV},i}$ and submit it to the blockchain. According to our blockchain assumptions, this transaction is published on the blockchain in block $\mathscr{B}_l$ with $j < l \leq j + \Delta < j + t \leq k$. However, once $\mathsf{TX}_{\mathsf{RV},i}$ is published in the block $\mathscr{B}_l$ of the blockchain, the transaction $\mathsf{TX}_{\mathsf{SP},i}$ becomes invalid and cannot be published in block $\mathscr{B}_k$ of the blockchain which leads to a contradiction. □

Theorem 6.1. *Let* Π *be a* EUF − CMA *secure digital signature,* $\mathscr{R}$ *be a hard relation and* Ξ *be a secure adaptor digital signature. Then, for a Garrison channel, an honest party* $P \in \{A, B\}$ *loses any funds in the channel with negligible probability.*

Proof. Without loss of generality let $P = A$. Funds of A are locked in $\mathsf{TX}_{\mathsf{FU}}$.Output. It is of negligible probability that any PPT adversary $\mathscr{A}$ spends the output of $\mathsf{TX}_{\mathsf{FU}}$ without the honest party A's authorisation. Otherwise, the underlying digital signature would be forgeable. Furthermore, $\mathsf{TX}_{\overline{\mathsf{SP}}}$, $\mathsf{TX}_{\mathsf{CM},i}$ with $i = [0, n-1]$, $\mathsf{TX}_{\mathsf{CM},n}$ are the only transactions in the protocol that spend the output of $\mathsf{TX}_{\mathsf{FU}}$ and A grants authorisation for. Thus, these transactions will be discussed further. Since $\mathsf{TX}_{\overline{\mathsf{SP}}}$ represents the final agreed state of the channel, its broadcast cannot cause A to lose any funds. Moreover, according to Lemmas 6.3 and 6.4, it is of negligible probability that broadcast of $\mathsf{TX}_{\mathsf{CM},n}$ causes A to be cheated out of her funds. Also, based on the protocol, A never publishes $\mathsf{TX}_{\mathsf{CM},i}$ with $i = [0, n-1]$ and according to Lemma 6.5, if one of these transactions is published by the adversary, it causes A to lose any funds with negligible probability. This concludes the proof.

□

6.6 Garrison Transactions Scripts

The funding transaction is similar to the corresponding transaction in the FPPW channel. Commit transaction has one input that takes the output of the funding transaction with witness script of 0 ⟨pubkeyA_sig⟩ ⟨pubkeyB_sig⟩. It also has three outputs where the script of its first output (main output) is as follows:

```
OP_IF
    # Revocation
    4 ⟨Rev_pubkeyA⟩ ⟨Rev_pubkeyB⟩ ⟨R_pubkeyA⟩ ⟨R_pubkeyB⟩ 4 OP_CHECKMULTISIG
OP_ELSE
    OP_IF
        ⟨delay 3t⟩ OP_CHECKSEQUENCEVERIFY OP_DROP
        2 ⟨Y_pubkeyA⟩ ⟨Y_pubkeyB⟩ 2 OP_CHECKMULTISIG
    OP_ELSE
        # Split
        ⟨delay t⟩ OP_CHECKSEQUENCEVERIFY OP_DROP
        2 ⟨Spl_pubkeyA⟩ ⟨Spl_pubkeyB⟩ 2  OP_CHECKMULTISIG
```

OP_ENDIF

OP_ENDIF

where ⟨Rev_pubkeyA⟩ and ⟨Spl_pubkeyA⟩ are public keys of *A* and ⟨R_pubkeyA⟩ and ⟨Y_pubkeyA⟩ are revocation key and publishing key of *A*, respectively. Also, ⟨Rev_pubkeyB⟩, ⟨Spl_pubkeyB⟩, ⟨R_pubkeyB⟩ and ⟨Y_pubkeyB⟩ are the corresponding parameters for *B*.

The script for the second output (auxiliary output) of the commit transaction is as following:

2 ⟨Aux_pubkeyA⟩ ⟨Aux_pubkeyB⟩ 2 OP_CHECKMULTISIG

where ⟨Aux_pubkeyA⟩ and ⟨Aux_pubkeyB⟩ are public keys of *A* and *B* respectively.

The script for the third output of the commit transaction is as following:

OP_RETURN ⟨Y_pubkeyA⟩ ⟨Y_pubkeyB⟩

where ⟨Y_pubkeyA⟩ and ⟨Y_pubkeyB⟩ are the same as the corresponding values used in the script in the first output of the commit transaction.

The revocation transaction spends the main output of a revoked commit transaction with the witness script of 0 ⟨Rev_pubkeyA_sig⟩ ⟨Rev_pubkeyB_sig⟩ ⟨R_pubkeyA_sig⟩ ⟨R_pubkeyB_sig⟩ 1. It also has one output with the script of 2 ⟨Y_pubkeyA⟩ ⟨Y_pubkeyB⟩ 2 OP_CHECKMULTISIG.

The auxiliary transaction spends the auxiliary output of the commit transaction using the witness script of 0 ⟨Aux_pubkeyA_sig⟩ ⟨Aux_pubkeyB_sig⟩. It also has two outputs. Its first output contains the following script:

⟨delay t⟩ OP_CHECKSEQUENCEVERIFY OP_DROP

2 ⟨Spl_pubkeyA⟩ ⟨Spl_pubkeyB⟩ 2

The second output of the auxiliary transaction is as follows:

OP_RETURN ⟨Rev_pubkeyA_sig⟩ ⟨Rev_pubkeyB_sig⟩ where ⟨Rev_pubkeyA_sig⟩ and ⟨Rev_pubkeyB_sig⟩ are the corresponding signatures used in the witness script of the corresponding revocation transaction.

The split transaction has two inputs. The first one takes the main output of the corresponding commit transaction with the witness script of 0 ⟨Spl_pubkeyA_Sig⟩ ⟨Spl_pubkeyB_Sig⟩ 0 0. The second input takes the first output of the corresponding auxiliary transaction with the witness script of 0 ⟨Spl_pubkeyA_Sig⟩ ⟨Spl_pubkeyB_Sig⟩.

6.7 Garrison Protocol

In this section, protocols for different phases of Garrison will be presented. In different steps of the protocol, channel participants generate (or verify) some signatures or pre-signatures on protocol transactions. When a signature or pre-signature is going to be generated (or verified) for j^{th} input of the transaction TX_i, the input message to the signing (or verification) algorithm is denoted by $f([\mathsf{TX}_i], j)$ [58].

Garrison channel creation protocol is as follows:

Preconditions: A and B own $a + \epsilon/2$ and $b + \epsilon/2$ coins on-chain in output of transactions with transaction identifiers $txid_A$ and $txid_B$ respectively. A and B know each other's public keys and values of ϵ, a and b that are going to be used in the channel.

1. Create $[\mathsf{TX}_{\mathsf{FU}}]$:
   (a) $P \in \{A, B\} \xrightarrow{txid_P} \bar{P}$
   (b) If P receives $txid_{\bar{P}}$, it creates $[\mathsf{TX}_{\mathsf{FU}}]$ according to 6.1. Else it stops.
2. Create $[\mathsf{TX}_{\mathsf{CM},0}]$:
   (a) $P \in \{A, B\}$ generates $(Y_{P,0}, y_{P,0}) \leftarrow \mathsf{GenR}$ and $(R_{P,0}, r_{P,0}) \leftarrow \mathsf{GenR}$.
   (b) $P \xrightarrow{Y_{P,0}, R_{P,0}} \bar{P}$
   (c) If P receives $Y_{\bar{P},0}$, it creates $[\mathsf{TX}_{\mathsf{CM},0}]$ according to 6.2. Else it stops.
3. Create $[\mathsf{TX}_{\mathsf{RV},0}]$:
   (a) Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{RV},0}]$ according to 6.3.
   (b) Party P computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{RV},0}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RV},0}]))$.
   (c) $P \xrightarrow{\sigma^{P}_{\mathsf{TX}_{\mathsf{RV},0}}} \bar{P}$.
   (d) If party P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{RV},0}}$ from $\bar{P}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{RV},0}], 1); \sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{RV},0}}) = 1$, it continues. Else it stops.
4. Create $[\mathsf{TX}_{\mathsf{AU},0}]$: Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{AU},0}]$ according to 6.4.
5. Create $[\mathsf{TX}_{\mathsf{SP},0}]$: Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{SP},0}]$ according to 6.5.
6. Create $\mathsf{TX}_{\mathsf{SP},0}$:
   (a) Party $P \in \{A, B\}$ computes $\sigma^{P,j}_{\mathsf{TX}_{\mathsf{SP},0}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{SP},0}], j))$ with $j := \{1, 2\}$.

(b) $P \xrightarrow{\sigma^{P,1}_{\mathsf{TX}_{\mathsf{SP},0}}, \sigma^{P,2}_{\mathsf{TX}_{\mathsf{SP},0}}} \bar{P}$

(c) If party P receives $\sigma^{\bar{P},j}_{\mathsf{TX}_{\mathsf{SP},0}}$ with $j := \{1,2\}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{SP},0}], j); \sigma^{\bar{P},j}_{\mathsf{TX}_{\mathsf{SP},0}}) = 1$ for $j := \{1,2\}$, it continues. Else it stops.

(d) Party P creates $\mathsf{TX}_{\mathsf{SP},0}$ according to 6.5.

7. Create $\mathsf{TX}_{\mathsf{AU},0}$:

(a) Party $P \in \{A, B\}$ computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{AU},0}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{AU},0}], 1))$.

(b) $P \xrightarrow{\sigma^{P}_{\mathsf{TX}_{\mathsf{AU},0}}} \bar{P}$

(c) If party P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{AU},0}}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{AU},0}], 1); \sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{AU},0}}) = 1$, it continues. Else it stops.

(d) Party P creates $\mathsf{TX}_{\mathsf{AU},0}$ according to 6.4.

8. Create $\mathsf{TX}_{\mathsf{CM},0}$:

(a) Party $P \in \{A, B\}$ computes $\tilde{\sigma}^{P}_{\mathsf{TX}_{\mathsf{CM},0}} = \mathsf{pSign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},0}], 1), Y_{\bar{P},0})$.

(b) $P \xrightarrow{\tilde{\sigma}^{P}_{\mathsf{TX}_{\mathsf{CM},0}}} \bar{P}$

(c) If party P receives $\tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},0}}$ s.t. $\mathsf{pVrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{CM},0}], 1), Y_{P,0}; \tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},0}}) = 1$, it computes $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},0}} = \mathsf{Adapt}(\tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},0}}, y_{P,0})$, computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{CM},0}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},0}], 1))$, creates $\mathsf{TX}_{\mathsf{CM},0}$ according to 6.2 and continues. Else it stops.

9. Create $\mathsf{TX}_{\mathsf{FU}}$:

(a) Party $P \in \{A, B\}$ computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{FU}}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{FU}}], j))$ where $j := 1$ if $P = A$ or $j := 2$ otherwise.

(b) $P \xrightarrow{\sigma^{P}_{\mathsf{TX}_{\mathsf{FU}}}} \bar{P}$

(c) If party P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{FU}}}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{FU}}], j); \sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{FU}}}) = 1$ with $j := 1$ if $P = B$ or $j := 2$ otherwise, it continues. Else it stops.

(d) Party P creates $\mathsf{TX}_{\mathsf{FU}}$.

10. Publish $\mathsf{TX}_{\mathsf{FU}}$: Party $P \in \{A, B\}$ publishes $\mathsf{TX}_{\mathsf{FU}}$ on-chain.

The Garrison channel update protocol is as follows:

Preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ is on-chain. The channel update phase has been completed i times and hence the channel is at state i.

1. Create $[\mathsf{TX}_{\mathsf{CM},i+1}]$:
   (a) $P \in \{A, B\}$ generates $(Y_{P,i+1}, y_{P,i+1}) \leftarrow \mathsf{GenR}$ and $(R_{P,i+1}, r_{P,i+1}) \leftarrow \mathsf{GenR}$.
   (b) $P \xrightarrow{Y_{P,i+1}, R_{P,i+1}} \bar{P}$
   (c) If P receives $Y_{\bar{P},i+1}$, it creates $[\mathsf{TX}_{\mathsf{CM},i+1}]$ according to 6.2. Else it stops.
2. Create $[\mathsf{TX}_{\mathsf{RV},i+1}]$:
   (a) Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{RV},i+1}]$ according to 6.3.
   (b) Party P computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{RV},i+1}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{RV},i+1}]))$.
   (c) $P \xrightarrow{\sigma^{P}_{\mathsf{TX}_{\mathsf{RV},i+1}}} \bar{P}$.
   (d) If party P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{RV},i+1}}$ from $\bar{P}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{RV},i+1}], 1); \sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{RV},i+1}}) = 1$, it continues. Else it stops.
3. Create $[\mathsf{TX}_{\mathsf{AU},i+1}]$: Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{AU},i+1}]$ according to 6.4.
4. Create $[\mathsf{TX}_{\mathsf{SP},i+1}]$: Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\mathsf{SP},i+1}]$ according to 6.5.
5. Create $\mathsf{TX}_{\mathsf{SP},i+1}$:
   (a) Party $P \in \{A, B\}$ computes $\sigma^{P,j}_{\mathsf{TX}_{\mathsf{SP},i+1}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{SP},i+1}], j))$ with $j := \{1, 2\}$.
   (b) $P \xrightarrow{\sigma^{P,1}_{\mathsf{TX}_{\mathsf{SP},i+1}}, \sigma^{P,2}_{\mathsf{TX}_{\mathsf{SP},i+1}}} \bar{P}$
   (c) If party P receives $\sigma^{\bar{P},j}_{\mathsf{TX}_{\mathsf{SP},i+1}}$ with $j := \{1, 2\}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{SP},i+1}], j); \sigma^{\bar{P},j}_{\mathsf{TX}_{\mathsf{SP},i+1}}) = 1$ for $j := \{1, 2\}$, it continues. Else it stops.
   (d) Party P creates $\mathsf{TX}_{\mathsf{SP},i+1}$ according to 6.5.
6. Create $\mathsf{TX}_{\mathsf{AU},i+1}$:
   (a) Party $P \in \{A, B\}$ computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{AU},i+1}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{AU},i+1}], 1))$.
   (b) $P \xrightarrow{\sigma^{P}_{\mathsf{TX}_{\mathsf{AU},i+1}}} \bar{P}$

(c) If party P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{AU},i+1}}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{AU},i+1}],1);\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{AU},i+1}}) = 1$, it continues. Else it stops.

(d) Party P creates $\mathsf{TX}_{\mathsf{AU},i+1}$ according to 6.4.

7. Create $\mathsf{TX}_{\mathsf{CM},i+1}$:

(a) Party $P \in \{A, B\}$ computes $\tilde{\sigma}^{P}_{\mathsf{TX}_{\mathsf{CM},i+1}} = \mathsf{pSign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},i+1}],1), Y_{\bar{P},i+1})$.

(b) $P \xhookrightarrow{\tilde{\sigma}^{P}_{\mathsf{TX}_{\mathsf{CM},i+1}}} \bar{P}$

(c) If party P receives $\tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},i+1}}$ s.t. $\mathsf{pVrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\mathsf{CM},i+1}],1), Y_{P,i+1}; \tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},i+1}}) = 1$, it computes $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},i+1}} = \mathsf{Adapt}(\tilde{\sigma}^{\bar{P}}_{\mathsf{TX}_{\mathsf{CM},i+1}}, y_{P,i+1})$, computes $\sigma^{P}_{\mathsf{TX}_{\mathsf{CM},i+1}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{CM},i+1}],1))$, creates $\mathsf{TX}_{\mathsf{CM},i+1}$ according to 6.2 and continues. Else it executes the non-collaborative closure phase (from P's point of view the channel is still at state i).

8. Revoke $\mathsf{TX}_{\mathsf{CM},i}$:

(a) $P \in \{A, B\} \xhookrightarrow{r_{P,i}} \bar{P}$.

(b) if P receives $r_{\bar{P},i}$ s.t. $(R_{\bar{P},i}, r_{\bar{P},i}) \in \mathscr{R}$, then continues. Else, it executes the non-collaborative closure phase (from P's point of view the channel is at state $i+1$).

Garrison channel collaborative closure protocol is as follows:

Preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ is on-chain. The channel is at state n.

1. Create $\mathsf{TX}_{\overline{\mathsf{SP}}}$:

(a) Party $P \in \{A, B\}$ creates $[\mathsf{TX}_{\overline{\mathsf{SP}}}]$ according to 6.6.

(b) P computes $\sigma^{P}_{\mathsf{TX}_{\overline{\mathsf{SP}}}} = \mathsf{Sign}_{sk_P}(f([\mathsf{TX}_{\overline{\mathsf{SP}}}],1))$.

(c) $P \xhookrightarrow{\sigma^{P}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}} \bar{P}$

(d) If P receives $\sigma^{\bar{P}}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}$ from $\bar{P}$ s.t. $\mathsf{Vrfy}_{pk_{\bar{P}}}(f([\mathsf{TX}_{\overline{\mathsf{SP}}}],1);\sigma^{\bar{P}}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}) = 1$, it continues. Else it executes the non-collaborative closure phase (from P's point of view the channel is still at state n).

2. Publish $\mathsf{TX}_{\overline{\mathsf{SP}}}$: Party $P \in \{A, B\}$ publishes $\mathsf{TX}_{\overline{\mathsf{SP}}}$ on-chain.

Garrison channel non-collaborative closure protocol is as follows:

Preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ is on-chain. The channel is at state n.

1. Party $P \in \{A, B\}$ publishes $\mathsf{TX}_{\mathsf{CM},n}$ on-chain.
2. Once $\mathsf{TX}_{\mathsf{CM},n}$ is recorded on-chain, P waits for t rounds and then publishes $\mathsf{TX}_{\mathsf{SP},n}$ on-chain.

The protocol for penalizing the cheating party is as following:

preconditions: The channel create phase is complete and $\mathsf{TX}_{\mathsf{FU}}$ is on-chain. The channel is at state $n > 0$. $\mathsf{TX}_{\mathsf{CM},i}$ with $i < n$ is recorded on-chain by a dishonest party. The honest party (or its watchtower) is always online.

1. The honest party P (or its watchtower) observes that $\mathsf{TX}_{\mathsf{CM},i}$ is on-chain. Party P (or its watchtower) extracts $y_{\bar{P},i} = \mathsf{Ext}(\sigma^{P}_{\mathsf{TX}_{\mathsf{CM},i}}, \tilde{\sigma}^{P}_{\mathsf{CM},i}, Y_{\bar{P},i})$.
2. If $\mathsf{TX}_{\mathsf{AU},i}$ is published within $3t$ rounds, Party P (or its watchtower) extracts $\sigma^{P}_{\mathsf{TX}_{\mathsf{RV},i}}$ and $\sigma^{\bar{P}}_{\mathsf{TX}_{\mathsf{RV},i}}$ from $\mathsf{TX}_{\mathsf{AU},i}$.Output[2], computes $\sigma'^{P}_{\mathsf{TX}_{\mathsf{RV},i}} = \mathsf{Sign}_{r_{P,i}}(f([\mathsf{TX}_{\mathsf{RV},i}], 1))$ and $\sigma'^{\bar{P}}_{\mathsf{TX}_{\mathsf{RV},i}} = \mathsf{Sign}_{r_{\bar{P},i}}(f([\mathsf{TX}_{\mathsf{RV},i}], 1))$, creates $\mathsf{TX}_{\mathsf{RV},i}$ according to 6.3, publishes it on the blockchain, claims its output and stops. Else, party P claims the output $\mathsf{TX}_{\mathsf{CM},i}$.Output[2] and stops.

6.8 Conclusion

In this chapter, we presented a payment channel with a watchtower, called Garrison, whose storage costs for both channel parties and the watchtower are logarithmic in the maximum number of channel updates. Garrison avoids state duplication. So, the number of transactions does not increase exponentially with the number of payment channels or applications built on top of each other. Furthermore, the number of transactions that are published on the blockchain upon fraud does not increase with the number of outputs in the published revoked state.

Regarding the properties defined in Chapter 4, Garrison provides agility as the watchtower contract might be made with any watchtower and it can initiate and terminate at any time. Like Monitor, DCWC, DCWC* and Outpost, Garrison provides strong privacy against the watchtower. Also, it potentially achieves weak watchtower privacy against the third party (Refer to Section 4.3.2 for more details.). The watchtower in Garrison is rewarded upon each channel update. So, Garrison provides watchtower fairness but since the watchtower does not lock any collateral per channel, the channel party might

Table 6.2: Different Properties of the Garrison Scheme.

	Bitcoin Support	Agility	Priv. ag. Watch.	Priv. ag. 3^{rd} Party	Watch. Fairness	α	β
Garrison	Yes	Yes	Strong	Weak	Yes	0	1

lose her funds and hence Garrison is unfair to the channel party. Correspondingly, it provides β-coverage with $\beta = 1$. Adding channel party fairness to Garrison has been left to future works. Table 6.2 summarises the mentioned properties for Garrison.

Chapter 7

Daric: a storage efficient channel with penalisation

We construct this chapter based on the full version of our published paper, "Daric: A Storage Efficient Payment Channel With Penalisation Mechanism" [63] (The full version paper is available at https://eprint.iacr.org/2022/1295.pdf).

7.1 Introduction

While the Lightning Network exhibits elegant design, it does have certain limitations. One such limitation is the linear increase in storage requirements for channel parties, as they must store all revocation secrets received from their counterparts as the number of channel updates grows. This leads to an escalation in the storage expenses, particularly for channel parties managing a large number of channels and frequent channel updates (e.g., hubs in the PCH use case). Additionally, to ensure the identification and penalization of misbehaving parties, the channel state is duplicated, resulting in each party having its own copy of the state. Then, if parties split their channel into sub-channels (for example in order to add an application like a *Virtual channel* [27] on top of the channel), the state of each sub-channel is duplicated and it must propagate on both duplicates of the parent channel. Thus, state duplication causes the number of transactions to exponentially rise with the number of applications k built on top of each other [14].

Towards a different direction, the payment channel eltoo [1] introduces `ANYPREVOUT` [31] (also known as `NOINPUT`) as a new Bitcoin signature type to deploy the concept of versioning. This allows channel parties to override the current channel state by creating a state with a higher version number, which can be published upon fraud. So, channel

parties in eltoo do not store any revocation secrets from old channel states. This simplifies the key management and offers more affordable watchtowers as the transaction with the highest version invalidates all previous states. Furthermore, if an honest party forgets about an update and publishes an outdated state, it does not result in the loss of funds.

However, eltoo is incentive incompatible because its lack of punishment might encourage a dishonest party to publish an old state; either the other side corrects it or the dishonest party wins [64]. The only discouraging factor–the fee for publishing the old state–is also determined by the dishonest party. Thus, she can set it to the minimum possible value, i.e. few cents for some blockchains such as Bitcoin hard forks (e.g. Litecoin and Bitcoin Cash) and less than 1 USD for Bitcoin. Moreover, the transaction fee is independent of the channel *capacity* (i.e. the total funds in the channel). Therefore, even for payment channels with a huge capacity of several BTCs (e.g., channels listed in [65]), the dishonest party's cost will be still below 1 USD (See Section 7.6.2 for detailed analysis). Additionally, enforcing a large transaction fee or restricting the channel capacity (proposed in [66]) might be unfavourable to the honest party.

Furthermore, a dishonest party in eltoo might publish multiple outdated states to delay the channel closure process [67]. Thus, eltoo fails to achieve *bounded closure*, i.e. honest party is not guaranteed that the channel closure completes within a bounded time. This compromises the security of time-based payments, e.g. *Hashed Time-Lock Contract* (HTLC) (See Section 7.6.1 for further analysis).

Therefore, the main motivation of this chapter is designing a Bitcoin payment channel that (1) provides constant storage, (2) achieves bounded closure, (3) provides incentive compatibility, and (4) avoids state duplication.

The contributions of this chapter are as follows:

- We present a new Bitcoin payment channel, called Daric, which (1) is provably secure in the Universal Composability (UC) framework, (2) achieves constant size storage for both channel parties and the watchtower, (3) provides bounded closure, (4) provides punishment mechanism and hence achieves incentive compatibility, (5) avoids state duplication without needing any particular property (e.g. adaptor signature properties) for the underlying digital signature, and (6) attains unlimited lifetime, given that channel parties on average pay each other at most once per second. Table 7.1 compares Daric with other Bitcoin payment channels.

- We compare Daric and eltoo and show Daric is robust against an attack [67] to eltoo, which we also formalise in this chapter. We further perform a cost-benefit

Table 7.1: Comparison of different payment channels with n channel updates and k recursive channel splitting.

Scheme	Party's St. Req.	Watch. St. Req.	Unl. Life.	Incent. Comp.	# of Txs	Ada. Sig. Avoid.	Bnd. Cls.
Lightning† [15]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	Y	Y	$\mathcal{O}(2^k)$	Y	Y
Generalized† [14]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	Y	Y	$\mathcal{O}(1)$	N	Y
Outpost [22]	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	N	Y	$\mathcal{O}(2^k)$	Y	Y
FPPW [54]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	Y	Y	$\mathcal{O}(1)$	N	Y
Cerberus [23]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	Y	Y	$\mathcal{O}(2^k)$	Y	Y
Sleepy† [28]	$\mathcal{O}(n)$	N/A	N	Y	$\mathcal{O}(2^k)$	Y	Y
eltoo [1]	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Y§	N	$\mathcal{O}(1)$	Y	N
Daric (this work)	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Y§	Y	$\mathcal{O}(1)$	Y	Y

†: If parties pre-generate n keys in a Merkle tree, their storage requirements decrease to $\mathcal{O}(\log n)$ but the channel lifetime becomes limited to n channel updates.

§: Given that the channel update rate is at most one update per second.

analysis to assess the attacker's revenue in practice. We also show (1) Daric provides a higher deterrent effect against profit-driven attackers than eltoo and (2) unlike eltoo, Daric's deterrent effect is flexible.

- We compare Daric, eltoo, Lightning, Generalized, Sleepy, Cerberus, FPPW and Outpost channels with respect to the amount of data that is published on the blockchain in different channel closure scenarios (See Table 7.2). We show that Daric in the dishonest closure scenario outperforms Lightning with at least 1 HTLC output as well as all other schemes. In the non-collaborative closure scenario, Daric outperforms Lightning with at least 7 HTLC outputs as well as Generalized, eltoo and FPPW. Moreover, we compute the number of operations required for each channel update and show that (1) Unlike Lightning, Daric values are independent of the number of HTLC outputs m and (2) Daric is comparable with other schemes (see Table 7.2).

7.2 Notations and Background

7.2.1 Notations

In this section, we add some new notations to the ones introduced in Section 3.3. We use $\overline{[\mathsf{TX}]}$ and $\overline{\mathsf{TX}}$ to denote ($\mathsf{TX.nLT}$, $\mathsf{TX.Output}$) and ($\mathsf{TX.nLT}$, $\mathsf{TX.Output}$, $\mathsf{TX.Witness}$), respectively. The absolute time-lock of i in an output condition is shown by $i^{\geq}$ and means the output cannot be spent unless the *nLockTime* parameter in the spending transaction

is equal to or greater than i. Since a transaction may only be recorded on the blockchain if its *nLockTime* is in the past, $i^{\geq}$ in an output condition ensures the output cannot be spent unless i is expired (i.e. i is in the past).

7.2.2 Background

7.2.2.1 Floating Transactions

Each signature in a Bitcoin transaction contains a flag, called SIGHASH, which specifies which part of the transaction has been signed. Typically, signatures are of type SIGHASH_ALL, meaning the signature authorises all inputs (i.e. references to previous outputs) and outputs. The SIGHASH of type ANYPREVOUT indicates that the signature does not authorise the inputs. This allows the signer to refer to any arbitrary UTXO whose condition is met by the transaction witness data. Such a transaction is called a *floating* transaction. The dotted arrow to TX″ in Fig. 7.1 shows that TX″ is a floating transaction whose signature matches the public key pk_C. This transaction is denoted by $\overline{\texttt{TX}''}$ to emphasise that since it is a floating transaction, its input is unspecified and can be any output with matching condition. The signature with the ANYPREVOUT flag is denoted by $\hat{\sigma}$.

7.2.2.2 eltoo [1]

An eltoo channel is created like a Lightning channel, but each state is represented by two transactions: (1) the *update* transaction and (2) the *settlement* transaction, where both parties have the same version of these two transactions. Each update transaction is a floating transaction that transfers all the channel funds to a new joint address. The update transaction's output can be spent by its corresponding settlement transaction, which splits the channel funds among parties. If A submits an old update transaction, she has to wait for a relative time-lock of T rounds before she can publish the corresponding settlement transaction. It gives some time to B to publish the latest update transaction (which is a floating transaction) and override the already published update transaction.

7.3 Daric Overview

To provide a high level overview of our solution, we start by reviewing the limitations of the Lightning channel and then gradually present our work.

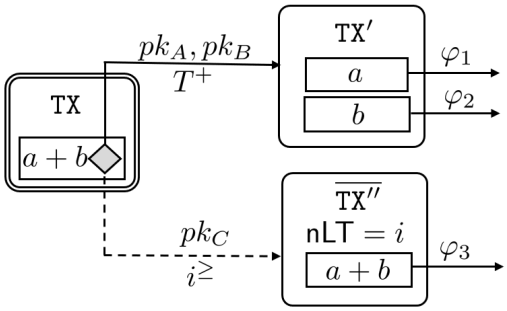

Figure 7.1: A sample transaction flow.

7.3.1 Revocation Per State

Parties' and their watchtower's storage in a Lightning channel increases over time as they should store some revocation-related data for each revoked state. Our main idea to reduce their storage is transforming the revocation transactions into floating transactions. Thereby, participants only need to store the latest revocation transaction with the largest version number and use it upon fraud. However, for a Lightning channel, (1) the monetary value of each revocation transaction typically differs from one state to another, and (2) each commit transaction might have multiple HTLC outputs and hence the number of revocation transactions might also differ from one state to another. So, since revocation transactions of different states differ in value and number, it is infeasible to replace them all with the latest revocation transactions.

Therefore, our first modification is following the *punish-then-split* mechanism, introduced in [14]. According to this mechanism, the commit transaction sends the channel funds to a new joint output, which is controlled by both parties. The output of this commit transaction can be spent by its corresponding split transaction after t rounds where outputs of the split transaction split the channel funds between A and B. If A publishes a revoked commit transaction, B must spend its output within t rounds with the corresponding revocation transaction. This revocation transaction gives all the channel funds to B. Fig. 7.2 depicts the transaction flows for this channel where each party stores a single revocation transaction with fixed monetary value (i.e. $a+b$ coins) per state. In this figure, $\mathsf{TX}_{\mathsf{FU}}$ denotes the funding transaction and $\mathsf{TX}^{A}_{\mathsf{CM},i}$, $\mathsf{TX}^{A}_{\mathsf{SP},i}$ and $\mathsf{TX}^{A}_{\mathsf{RV},i}$ (or respectively $\mathsf{TX}^{B}_{\mathsf{CM},i}$, $\mathsf{TX}^{B}_{\mathsf{SP},i}$ and $\mathsf{TX}^{B}_{\mathsf{RV},i}$) denote the commit, split and revocation transactions held by A (or respectively held by B) for state i.

7.3.2 Revocation Per Channel

In the scheme, depicted in Fig. 7.2, channel parties need to store a revocation transaction for each revoked state. Therefore, storage requirements of channel parties (or their

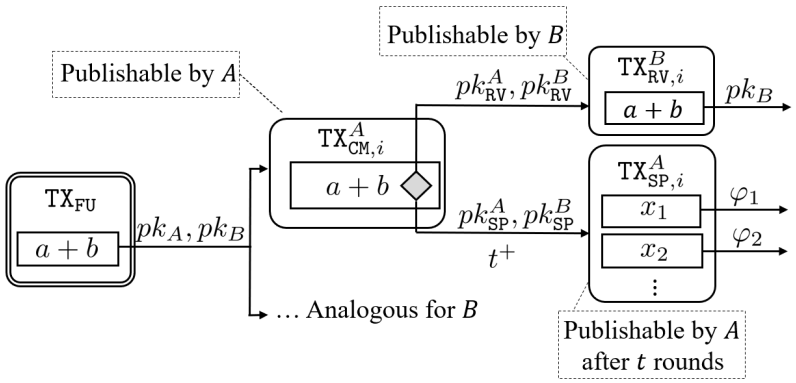

Figure 7.2: Transaction flows for a Lightning channel with punish-then-split mechanism.

watchtower) increase with each channel update. To solve this issue, we transform revocation transactions into floating transactions, i.e. the signatures in a revocation transaction, held by A, are of type ANYPREVOUT and meet the output condition of all commit transactions, held by B, and vice versa. It allows parties to only store the last revocation transaction.

7.3.3 Avoiding State Duplication

Since each state in the introduced scheme contains two split transactions (one for each party), the scheme suffers from state duplication. To avoid this, we transform split transactions into floating transactions. Then, each state contains one split transaction (held by both parties), which spends any of two commit transactions of that state.

7.3.4 State Ordering

Since split and revocation transactions are floating, it must be guaranteed that the latest commit transaction cannot be spent using any split or revocation transaction from previous states. Otherwise, the honest party, who has published the latest commit transaction, might lose some funds in the channel. To achieve this requirement, we repurpose [1] the *nLockTime* parameter of split and revocation transactions to store the *state number*: the number of times the channel has been updated to date. Furthermore, we add the state number to the output condition of each commit transaction as an absolute time-lock. Then, since the absolute time-lock in the output condition of the last commit transaction would be larger than the *nLockTime* parameter in any split or revocation transaction from previous states, the mentioned requirement is met.

7.3.5 Putting Pieces Together

The transaction flow for state i of Daric is depicted in Fig. 7.3. Let the channel be in state n. To close the channel, each party (e.g. A) can publish the latest commit transaction (e.g. $\mathsf{TX}^{A}_{\mathsf{CM},n}$), wait for T rounds and finally publish the latest split transaction $\mathsf{TX}_{\mathsf{SP},n}$. There is no revocation transaction for the latest state. If party B publishes a revoked commit transaction (i.e. $\mathsf{TX}^{B}_{\mathsf{CM},i}$ with $i < n$), then party A instantly publishes the latest revocation transaction $\mathsf{TX}^{A}_{\mathsf{RV},n-1}$ to take all the channel funds.

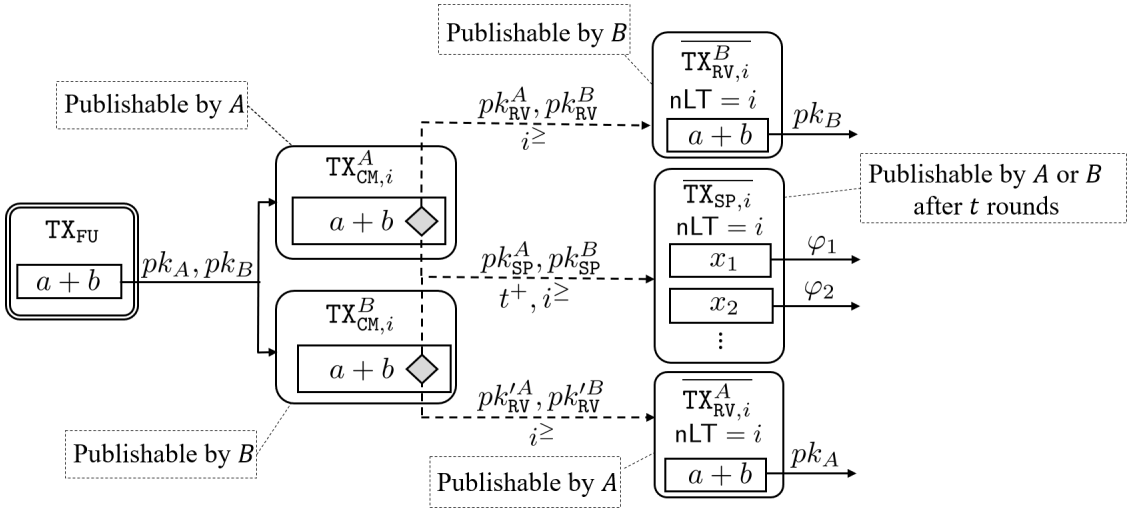

Figure 7.3: Transaction flows for state i of a Daric channel.

7.4 Daric Protocol Description

This section presents our protocol using the transaction flows depicted in Fig. 7.3. The lifetime of a Daric channel can be divided into 4 phases including create, update, close, and punish. We introduce these phases through sections 7.4.1 to 7.4.4. Section 7.11 provides the formal description of the protocol.

7.4.1 Create

To create the channel, A and B sign and publish the funding transaction $\mathsf{TX}_{\mathsf{FU}}$ on the blockchain. By publishing this transaction, A and B fund the channel with a and b coins, respectively, but since the output of the funding transaction can only be spent if both parties agree, one party might become unresponsive to raise a hostage situation. To avoid this, before signing the funding transaction, parties commit to the initial channel state, i.e. state 0, by exchanging signatures for the corresponding commit and split

transactions. Let us explain the different steps of the channel creation phase in more detail.

- **Step 1:** At the first step, A and B send their funding sources (i.e. $txid_A$ and $txid_B$) to each other. This enables them to create the body of the funding transaction $[\mathtt{TX}_{\mathtt{FU}}]$.
- **Step 2:** Having the transaction identifier of $\mathtt{TX}_{\mathtt{FU}}$, parties create the body of the commit transactions, i.e. $[\mathtt{TX}^{A}_{\mathtt{CM},0}]$ and $[\mathtt{TX}^{B}_{\mathtt{CM},0}]$.
- **Steps 3:** Parties exchange the required signatures (with SIGHASH of type ANYPREVOUT) to create the floating transaction $\overline{\mathtt{TX}_{\mathtt{SP},0}}$. This floating transaction could take output of $\mathtt{TX}^{A}_{\mathtt{CM},0}$ or $\mathtt{TX}^{B}_{\mathtt{CM},0}$ as its input.
- **Step 4:** Parties exchange the required signatures to create the commit transactions $\mathtt{TX}^{A}_{\mathtt{CM},0}$ and $\mathtt{TX}^{B}_{\mathtt{CM},0}$.
- **Step 5:** Parties exchange the required signatures to create the funding transactions $\mathtt{TX}_{\mathtt{FU}}$.
- **Step 6:** Parties publish the funding transaction on the blockchain.

The absolute time-lock in the output script of commit transactions and correspondingly the *nLockTime* parameter in the split transaction must be in the past. Otherwise, parties have to wait to publish such transactions. As explained in Section 7.3.4, the time-lock is set to the state number and hence its value increases with each channel update. Absolute time-locks lower than 500,000,000 specify the block number after which the transaction can be included in a block. According to the value of the current block height, if we set the initial time-lock to the first state number, i.e. 0, the channel can be updated around 700,000 times. However, absolute time-locks equal to or larger than 500,000,000 specify the UNIX timestamp after which the transaction will be valid.

According to the value of the current timestamp, if we set the initial time-lock (and correspondingly *nLockTime* parameter) to 500,000,000, the channel can be updated around 1 billion times [1]. Moreover, the current timestamp increases one unit per second, meaning if the average rate of the channel update is up to once per second, the channel can be updated an infinite number of times.

The above-mentioned transactions are further explained below.

- **Funding transaction**: Using this transaction, channel parties A and B open a Daric channel by funding a and b, coins into the channel, respectively. The funding

transaction is as follows[1]:

$$\begin{aligned}
\mathsf{TX}_{\mathsf{FU}}.\mathsf{nLT} &:= 0,\\
\mathsf{TX}_{\mathsf{FU}}.\mathsf{Input} &:= (txid_A \| x, txid_B \| y),\\
\mathsf{TX}_{\mathsf{FU}}.\mathsf{Output} &:= \{(a+b, pk_A \wedge pk_B)\},\\
\mathsf{TX}_{\mathsf{FU}}.\mathsf{Witness} &:= ((1, \sigma^{A,1}_{\mathsf{TX}_{\mathsf{FU}}}), (1, \sigma^{B,2}_{\mathsf{TX}_{\mathsf{FU}}})).
\end{aligned}$$

- **Commit transaction**: There exist two versions of commit transaction per state, each held by one of the parties, but only the first ones ($\mathsf{TX}^A_{\mathsf{CM},i}$ and $\mathsf{TX}^B_{\mathsf{CM},i}$ with $i = 0$) are created at this phase. Commit transactions send the channel funds to a joint account. Commit transactions for state i are as follows:

$$\begin{aligned}
\mathsf{TX}^A_{\mathsf{CM},i}.\mathsf{nLT} &:= 0,\\
\mathsf{TX}^A_{\mathsf{CM},i}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid} \| 1,\\
\mathsf{TX}^A_{\mathsf{CM},i}.\mathsf{Output} &:= \{(a+b, \varphi_1 \vee \varphi_2)\},\\
\mathsf{TX}^A_{\mathsf{CM},i}.\mathsf{Witness} &:= \{(1, \{\sigma^A_{\mathsf{TX}^A_{\mathsf{CM},i}}, \sigma^B_{\mathsf{TX}^A_{\mathsf{CM},i}}\})\}
\end{aligned}$$

and

$$\begin{aligned}
\mathsf{TX}^B_{\mathsf{CM},i}.\mathsf{nLT} &:= 0,\\
\mathsf{TX}^B_{\mathsf{CM},i}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid} \| 1,\\
\mathsf{TX}^B_{\mathsf{CM},i}.\mathsf{Output} &:= \{(a+b, \varphi_1 \vee \varphi'_2\},\\
\mathsf{TX}^B_{\mathsf{CM},i}.\mathsf{Witness} &:= \{(1, \{\sigma^A_{\mathsf{TX}^B_{\mathsf{CM},i}}, \sigma^B_{\mathsf{TX}^B_{\mathsf{CM},i}}\})\}
\end{aligned}$$

with $\varphi_1 := (pk^A_{\mathsf{SP}} \wedge pk^B_{\mathsf{SP}} \wedge t^+ \wedge i^{\geq})$, $\varphi_2 = (pk^A_{\mathsf{RV}} \wedge pk^B_{\mathsf{RV}} \wedge i^{\geq})$ and $\varphi'_2 = (pk'^A_{\mathsf{RV}} \wedge pk'^B_{\mathsf{RV}} \wedge i^{\geq})$. Meeting each sub-condition requires both parties' authorisation. The parameters $i^{\geq}$ and t^+ show absolute time-lock of i and relative time-lock of t rounds, respectively. The parameter t can be set to any value larger than the blockchain delay Δ.

- **Split transaction** There is one split transaction per state which is held by both channel parties, but only the first one ($\mathsf{TX}_{\mathsf{SP},i}$ with $i = 0$) is created at this phase. The split transaction for state i determines the balance of each channel party in

[1]We assume that funding sources of $\mathsf{TX}_{\mathsf{FU}}$ are two typical UTXOs owned by A and B.

the i^{th} channel state. For this transaction, we have:

$$\begin{aligned}
\mathsf{TX}_{\mathsf{SP},i}.\mathsf{nLT} &:= S_0 + i,\\
\mathsf{TX}_{\mathsf{SP},i}.\mathsf{Input} &:= \mathsf{TX}^{P}_{\mathsf{CM},i}.\mathsf{txid}\|1, \text{ with } P \in \{A, B\}.\\
\mathsf{TX}_{\mathsf{SP},i}.\mathsf{Output} &:= (\theta_1, \theta_2, ...),\\
\mathsf{TX}_{\mathsf{SP},i}.\mathsf{Witness} &:= \{(1, \{\hat{\sigma}^{A}_{\mathsf{TX}_{\mathsf{SP},i}}, \hat{\sigma}^{B}_{\mathsf{TX}_{\mathsf{SP},i}}\})\}
\end{aligned}$$

The transaction $\mathsf{TX}_{\mathsf{SP},i}$ is a floating transaction whose witness satisfies the first sub-condition of the output of $\mathsf{TX}^{A}_{\mathsf{CM},i}$ or $\mathsf{TX}^{B}_{\mathsf{CM},i}$ ($pk^{A}_{\mathsf{SP}} \wedge pk^{B}_{\mathsf{SP}} \wedge t^{+} \wedge i^{\geq}$). Normally, given that parties are honest and the channel is in state n, one of two commit transactions of the latest state, i.e. either $\mathsf{TX}^{A}_{\mathsf{CM},n}$ or $\mathsf{TX}^{B}_{\mathsf{CM},n}$, is published on-chain. Then, the first sub-condition of its output with the ($pk^{A}_{\mathsf{SP}} \wedge pk^{B}_{\mathsf{SP}} \wedge t^{+} \wedge i^{\geq}$) is satisfied by the corresponding split transaction $\mathsf{TX}_{\mathsf{SP},n}$ after t rounds.

To reduce the required communication between channel parties for each channel update, pk^{A}_{SP} and pk^{B}_{SP} do not change from one state to the next. However, since split transactions are floating, it must be guaranteed that the split transaction of state i cannot take the output of one of the commit transactions of the next states as its input because otherwise the output of the latest commit transaction, let's say $\mathsf{TX}^{A}_{\mathsf{CM},n}$, could be spent using an old split transaction $\mathsf{TX}_{\mathsf{SP},i}$ with $i < n$, which is undesirable. To meet this requirement, the *nLockTime* parameter of $\mathsf{TX}_{\mathsf{SP},i}$ is set to i. Then since the first sub-condition of commit transactions of state j with $j > i$ are time-locked using $j^{\geq}$ but $\mathsf{TX}_{\mathsf{SP},i}.\mathsf{nLT} \geq j$ does not hold, $\mathsf{TX}_{\mathsf{SP},i}$ cannot spend output of $\mathsf{TX}^{A}_{\mathsf{CM},j}$ or $\mathsf{TX}^{B}_{\mathsf{CM},j}$.

7.4.2 Update

Let the channel be in state $i \geq 0$ and channel parties decide to update it to state $i + 1$. The update process is performed in two sub-phases. The first sub-phase is similar to steps 2 to 4 of the channel creation phase where channel parties create two new commit transactions $\mathsf{TX}^{A}_{\mathsf{CM},i+1}$ and $\mathsf{TX}^{B}_{\mathsf{CM},i+1}$ as well as a new split transaction $\overline{\mathsf{TX}_{\mathsf{SP},i+1}}$ for the new state. In the second sub-phase, channel parties revoke the state i by signing two revocation transactions $\overline{\mathsf{TX}^{A}_{\mathsf{RV},i}}$ and $\overline{\mathsf{TX}^{B}_{\mathsf{RV},i}}$. The revocation transaction $\overline{\mathsf{TX}^{A}_{\mathsf{RV},i}}$ (or respectively $\overline{\mathsf{TX}^{B}_{\mathsf{RV},i}}$) contains no input yet and can spend output of any commit transaction $\overline{\mathsf{TX}^{B}_{\mathsf{CM},j}}$ (or respectively $\overline{\mathsf{TX}^{A}_{\mathsf{CM},j}}$) with $j \leq i$. With each channel update, the state number and hence the time-lock value in the output condition of each commit transaction and *nLockTime* in split and revocation transactions increase by one unit. Let us explain the different steps of the channel update phase in more detail.

- **Step 1:** Parties create the body of the commit transactions, i.e. $[\mathsf{TX}^{A}_{\mathsf{CM},i+1}]$ and $[\mathsf{TX}^{B}_{\mathsf{CM},i+1}]$.
- **Steps 2:** Parties exchange the required signatures (with SIGHASH of type ANYPREVOUT) to create the floating transaction $\overline{\mathsf{TX}_{\mathsf{SP},i+1}}$. This floating transaction takes output of $\mathsf{TX}^{A}_{\mathsf{CM},i+1}$ or $\mathsf{TX}^{B}_{\mathsf{CM},i+1}$ as its input.
- **Step 3:** Parties exchange the required signatures to create the commit transactions $\mathsf{TX}^{A}_{\mathsf{CM},i+1}$ and $\mathsf{TX}^{B}_{\mathsf{CM},i+1}$.
- **Step 4:** Parties exchange the required signatures (with SIGHASH of type ANYPREVOUT) to create the floating transactions $\overline{\mathsf{TX}^{A}_{\mathsf{RV},i}}$ and $\overline{\mathsf{TX}^{B}_{\mathsf{RV},i}}$.

One of the parties might receive the signature on the split or commit transactions in steps 2 or 3 (or respectively receive the signature on the revocation transaction in step 4) but avoid signing the corresponding transaction for the other party. In such situations, the honest party non-collaboratively closes the channel with the latest valid channel state, i.e. state i (or respectively state $i+1$). More technical details can be found in Section 7.11.

Revocation transaction will be introduced further below:

- **Revocation transaction** Once the channel is updated from state i to $i+1$, two versions of revocation transaction are created for state i, one version for each channel party. The revocation transaction held by party A and party B for state i are denoted by $\mathsf{TX}^{A}_{\mathsf{RV},i}$ and $\mathsf{TX}^{B}_{\mathsf{RV},i}$, respectively, where

$$\begin{aligned}
\mathsf{TX}^{A}_{\mathsf{RV},i}.\mathsf{nLT} &:= i,\\
\mathsf{TX}^{A}_{\mathsf{RV},i}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{CM},j,B}.\mathsf{txid}\|1, \text{ with } j \le i,\\
\mathsf{TX}^{A}_{\mathsf{RV},i}.\mathsf{Output} &:= \{(a+b, pk_A)\},\\
\mathsf{TX}^{A}_{\mathsf{RV},i}.\mathsf{Witness} &:= \{(2, \{\hat{\sigma}^{A}_{\mathsf{TX}^{A}_{\mathsf{RV},i}}, \hat{\sigma}^{B}_{\mathsf{TX}^{A}_{\mathsf{RV},i}}\})\}
\end{aligned}$$

and

$$\begin{aligned}
\mathsf{TX}^{B}_{\mathsf{RV},i}.\mathsf{nLT} &:= i,\\
\mathsf{TX}^{B}_{\mathsf{RV},i}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{CM},j,A}.\mathsf{txid}\|1, \text{ with } j \le i,\\
\mathsf{TX}^{B}_{\mathsf{RV},i}.\mathsf{Output} &:= \{(a+b, pk_B)\},\\
\mathsf{TX}^{B}_{\mathsf{RV},i}.\mathsf{Witness} &:= \{(2, \{\hat{\sigma}^{A}_{\mathsf{TX}^{B}_{\mathsf{RV},i}}, \hat{\sigma}^{B}_{\mathsf{TX}^{B}_{\mathsf{RV},i}}\})\}
\end{aligned}$$

The transaction $\mathsf{TX}^{A}_{\mathsf{RV},i}$ (or $\mathsf{TX}^{B}_{\mathsf{RV},i}$) is a floating transaction whose witness satisfies the second sub-condition of the output of $\mathsf{TX}_{\mathsf{CM},j,B}$ (or $\mathsf{TX}_{\mathsf{CM},j,A}$) with $j \le i$. Given that

the channel is in state n, if a dishonest channel party, let's say party B, publishes the old commit transaction $\mathsf{TX}_{\mathsf{CM},j,B}$ with $j < n$, he must wait for T rounds before being able to publish its corresponding split transaction $\mathsf{TX}_{\mathsf{SP},j}$. However, A can instantly publish the latest revocation transaction $\mathsf{TX}_{\mathsf{RV},n-1,A}$ and claim its output.

Since revocation transactions are floating and meet the second sub-condition of commit transactions, the public keys that are used in such sub-conditions must not change from one state to the next. However, it must be guaranteed that the revocation transaction of state i cannot take the output of commit transactions of the next states as input. Thus, similar to what we did for split transactions, the *nLockTime* parameter of revocation transactions of state i is set to S_0+i. Then since the second sub-condition of each commit transaction of state i is also time-locked using $CLTV_{S_0+i}$, the desired requirement is met.

A dishonest channel party, e.g. party A, must not be able to publish both a revoked commit transaction $\mathsf{TX}^A_{\mathsf{CM},i}$ with $i < n$ as well as a revocation transaction $\mathsf{TX}_{\mathsf{RV},j,A}$ with $j \geq i$ on the ledger. Otherwise, he can take all the channel funds. To prevent A from doing so, the revocation public keys that are used in commit transactions held by A (pk^A_{RV} and pk^B_{RV}) are different from those in commit transactions held by B (pk'^A_{RV} and pk'^B_{RV}). Therefore, the output of a revoked commit transaction held by A can only be spent by a revocation transaction held by B and vice versa.

7.4.3 Close

Assume while the channel between A and B is in state n, they decide to collaboratively close it. To do so, A and B exchange signatures for a new transaction, called modified split transaction $\mathsf{TX}_{\overline{\mathsf{SP}}}$, and publish it on the blockchain. This transaction takes the funding transaction's output as its input and splits the channel funds among channel parties. The transaction $\mathsf{TX}_{\overline{\mathsf{SP}}}$ is as following:

$$\begin{aligned}
\mathsf{TX}_{\overline{\mathsf{SP}}}.\mathsf{Input} &:= \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid} \| 1, \\
\mathsf{TX}_{\overline{\mathsf{SP}}}.\mathsf{Output} &:= \mathsf{TX}_{\mathsf{SP},n}.\mathsf{Output}, \\
\mathsf{TX}_{\overline{\mathsf{SP}}}.\mathsf{Witness} &:= \{(1, \{\sigma^A_{\mathsf{TX}_{\overline{\mathsf{SP}}}}, \sigma^B_{\mathsf{TX}_{\overline{\mathsf{SP}}}}\})\}.
\end{aligned}$$

If one of the channel parties, e.g. party B, becomes unresponsive, its counterparty A can still non-collaboratively close the channel by publishing $\mathsf{TX}^A_{\mathsf{CM},n}$, adding the output of $\mathsf{TX}^A_{\mathsf{CM},n}$ as an input to $\overline{\mathsf{TX}_{\mathsf{SP},n}}$ to transform it into $\mathsf{TX}_{\mathsf{SP},n}$, and finally publishing $\mathsf{TX}_{\mathsf{SP},n}$ after t rounds.

7.4.4 Punish

Let the channel be in state n. If a dishonest channel party, let's say A, publishes an old commit transaction $\mathsf{TX}^{A}_{\mathsf{CM},i}$ with $i < n$ on the ledger, party B adds the output of $\mathsf{TX}^{A}_{\mathsf{CM},i}$ as an input to $\overline{\mathsf{TX}^{B}_{\mathsf{RV},n-1}}$ in order to transform it into $\mathsf{TX}^{B}_{\mathsf{RV},n-1}$ and instantly publishes $\mathsf{TX}^{B}_{\mathsf{RV},n-1}$ on the blockchain.

7.5 Security Analysis Overview

In this section, we first provide some payment channel notations as well as our security model, which follows previous works on *layer-2* solutions [14, 68, 69, 29]. Then, we present desired properties of a payment channel and an ideal functionality $\mathcal{F}$ that attains those properties. Finally, we show that the Daric protocol is a realisation of the ideal functionality $\mathcal{F}$ and hence achieves its desired properties.

7.5.1 Notation and Security Model

We use an extended version of the *universal composability* framework [55] to formally model the security of our construction. This extended version [70], called the Global Universal Composability framework (GUC), supports a global setup. To simplify our model, we assume that the communication network is synchronous, meaning that the protocol is executed through multiple rounds and parties in the protocol are connected to each other via an authenticated communication channel which guarantees 1-round delivery. Transactions are recorded by a global ledger $\mathcal{L}(\Delta, \Sigma)$, where Σ is a signature scheme used by the blockchain and Δ is an upper bound on the blockchain delay: the number of rounds it takes a transaction to be accepted by the ledger. Section 7.9 provides more details on our security model.

We abbreviate a daric payment channel γ as an attribute tuple $\gamma := (\mathsf{id}, \mathsf{users}, \mathsf{cash}, \mathsf{st}, \mathsf{sn}, \mathsf{flag}, \mathsf{st}')$, where $\gamma.\mathsf{id} \in \{0,1\}^*$ defines the channel identifier, $\gamma.\mathsf{users}$ represents the identities of the channel users, $\gamma.\mathsf{cash} \in \mathbb{R}^{\geq 0}$ is the channel capacity, $\gamma.\mathsf{st} := (\theta_1, \ldots, \theta_l)$ is a list of l outputs defining the channel state after the last complete channel update and $\gamma.\mathsf{sn}$ is the state number. The flag $\gamma.\mathsf{flag} \in \{1,2\}$ and the state $\gamma.\mathsf{st}'$ will be explained below.

The initial value of $\gamma.\mathsf{flag}$ and $\gamma.\mathsf{st}'$ are 1 and $\perp$, respectively. Assume that the channel has been updated $n \geq 0$ times and the channel state after the n^{th} update is st and hence we have $\gamma.\mathsf{st} = st$. Now, assume that parties start the update process to update the state of the channel from state st to st'. From a particular point in the channel update process

onward, at least one of the parties has sufficient data to enforce the new state st' on the blockchain when parties have not completely revoked the state γ.st yet. The flag γ.flag is set to 2 to identify such occasions and γ.st$'$ is set to st' to maintain the new state. Thus, when γ.flag $= 2$, the channel might be finalised with either γ.st or γ.st$'$. At the end of the channel update process, once the state st was revoked by both parties, γ.st and γ.st$'$ are set to st' and $\perp$, respectively, and γ.flag is set to 1.

7.5.2 Ideal Functionality Properties

This section closely follows [14] to introduce desired security and efficiency properties of a payment channel as follows:

- **Consensus on creation:** A channel γ is created only if both channel parties in the set γ.users agree to create it.
- **Consensus on update:** A channel γ is updated only if both channel parties in the set γ.users agree to update it. Also, parties reach an agreement on update acceptance or rejection within a bounded number of rounds (the bound might depend on the ledger delay Δ).
- **Bounded closure with punish:** An honest user $P \in \gamma$.users has the assurance that within a bounded number of rounds (the bound might depend on the ledger delay Δ), she can finalise the channel state on the ledger either by enforcing a state that gives her γ.cash coins, or by enforcing γ.st if γ.flag $= 1$ or by enforcing either γ.st γ.st$'$ otherwise.
- **Optimistic update:** If both parties in γ.users are honest, the channel update completes with no ledger interaction.

Section 7.10 introduces an ideal functionality $\mathcal{F}$ that achieves these properties. Theorem 7.1 shows Daric protocol, denoted by π, is a realisation of $\mathcal{F}$ and hence achieves its desired properties. It follows from 14 Lemmas. Section 7.12 presents the full security proof.

Theorem 7.1. *Let Σ be an $\mathsf{EUF-CMA}$ secure signature scheme. Then, for any ledger delay $\Delta \in \mathbb{N}$, the protocol π UC-realises the ideal functionality $\mathcal{F}(T)$ with any $t > \Delta$.*

Proof. Theorem follows directly from Lemma 7.1, Lemma 7.4, Lemma 7.5 and Lemma 7.8, presented in Section 7.12. □

To enhance structure and prevent an overly lengthy section, we partition the necessary contents for our security proof into two separate chapters. We formally define π in Section 7.11 and then provide a simulator $\mathcal{S}$ in Section 7.12 where $\mathcal{S}$ has interaction with the ideal functionality $\mathcal{F}$ and $\mathcal{L}$. The simulator simulates the content and timing of all messages of the honest party to the adversary and also translates any message from the adversary into a message to the ideal functionality, such that an indistinguishable execution of the protocol in the ideal world is emulated. Also, in Section 7.12, we prove for any action that causes the ideal functionality to output Error with non-negligible probability, the simulator constructs a reduction against the existential unforgeability of the underlying signature scheme Σ with non-negligible success probability, which contradicts with our assumption regarding the security of Σ. This proves our protocol would be as secure as the ideal functionality $\mathcal{F}$ and provides its desirable properties.

7.6 Daric Versus Eltoo

In section 7.6.1, we present an attack to eltoo whose main purpose is to postpone the channel closure. We show this attack is practically profitable when applied to eltoo but it cannot be applied to Daric. In section 7.6.2, we analyse Daric and eltoo to compare their robustness against profit-driven attackers. We use the statistical data derived from the Lightning Network to enable such an analysis.

7.6.1 HTLC Security

This section presents an attack against HTLC security in eltoo (previously informally discussed in [67]) and analyses the attacker's revenue. Let the adversary represent two nodes on the PCN: node M_1 and node M_2. Assume that the adversary has established N channels from M_1 to victim nodes $V_1, \dots, V_N$ and N channels from victim nodes to M_2. The channel between M_1 and V_i is denoted with γ_i. The adversary performs N simultaneous HTLC payments from M_1 to M_2 through $V_1, \dots, V_N$. Let the payment value for all HTLCs be A coins and the time-lock for all these payments for M_1's channels be T. Assume that M_2 accepts the payments and provides the required secrets for all HTLC payments and hence M_2 is paid $N \cdot A$ coins in total. Then, victims provide the secrets to the node M_1. However, M_1 does not update her channels with victims. Therefore, victims attempt to claim all HTLCs on-chain. To prevent victims from closing their channels in time, M_1 takes the following steps:

1. Submit a valid *Delay* transaction $\mathsf{TX}_{\mathsf{De}}$ with $N+1$ inputs and $N+1$ outputs where the i^{th} input-output pair corresponds with an outdated state of the channel γ_i and

the last input-output pair adds further funds to be used as the transaction fee, which is set to any value larger than A.

2. If $\mathsf{TX}_{\mathsf{De}}$ is published and the time-lock t is still unexpired, go to step 1.

3. Once the time-lock t is expired, submit the latest channel state for all channels and claim their HTLC outputs.

In the above attack, to replace the already submitted transaction $\mathsf{TX}_{\mathsf{De}}$ with the latest state of the channel γ_i, V_i has to set a transaction fee that is larger than the total absolute transaction fee of $\mathsf{TX}_{\mathsf{De}}$ [71]. But since the transaction fee for $\mathsf{TX}_{\mathsf{De}}$ is larger than A, V_i will be unwilling to pay such a transaction fee.

Once the HTLC time-lock is expired and the latest channel state is added to the ledger, there will be a race between M_1 and each victim to claim the HTLC output. The adversary will have a better chance to win the race if she has a better network connection with a higher number of nodes.

Now we perform a cost-benefit analysis to determine if the attack is profitable to the attacker. For a fixed value of A, with setting N to the largest possible value, the adversary (1) reduces the fee per channel for each delay transaction and (2) reduces the pace at which outdated states are added to the blockchain. A Bitcoin transaction can contain up to 100,000 VBytes (where each VByte equals four weight units) and each input-output pair contains 222 bytes of witness data and 84 bytes of non-witness data (See Appendix H.4 in the full version of the Daric paper [63] for more details). Therefore, $\mathsf{TX}_{\mathsf{De}}$ can cover up to around $\frac{100{,}000}{0.25\times 222+84} \approx 715$ eltoo channels. The minimum possible fee rate is 1 Satoshi per VByte. Thus, if A is set to 100,000 Satoshi, the total fee for each delay transaction would be 100,000 Satoshi.

At the time of writing this chapter (in January 2023), the average transaction fee is quite low and hence transactions with the minimum fee rate are added to the blockchain in 30 minutes. It means if HTLC time-locks are set to 3 days, 144 delay transactions are published before time-locks getting expired. In other words, the adversary pays $144A$ as a transaction fee to earn up to $715A$. In more congested times, it might take several hours for a transaction with a minimum fee rate to be added to the blockchain. Thus, the attack could be even more profitable to the attacker. This attack is inapplicable to Daric because once the attacker publishes an old commit transaction, the only valid transactions are the revocation transactions held by her counterparty.

7.6.2 Punishment Mechanism

Prior to providing a formal analysis, we provide intuitions as follows. The only cost for a dishonest party in eltoo is the fee for publishing the old state, which could be (1) less than

1 USD for Bitcoin and (2) independent of the channel capacity. However, given that the balance of each party in a Daric channel cannot be less than 1% of the channel capacity (which is currently deployed in the Lightning Network), the minimum amount that a dishonest party might lose would have the following properties: (1) It is proportional to the channel capacity, (2) Its value (around 20 USD on average in the Lightning Network in April 2022) is typically significantly larger than the transaction fee and (3) It is easily raised by increasing the minimum possible balance of each channel party from 1% of the channel capacity to a higher proportion. Therefore, Daric's deterrent effect against profit-driven attackers is higher and more flexible than that of eltoo.

Now, we present a more formal comparison between eltoo and Daric. We assume the channel party either stays online or employs a watchtower that is fair w.r.t the hiring party [54] (i.e. the watchtower guarantees its client's funds in the channel). For the former case, let p denote the probability that the honest channel party successfully reacts upon fraud, i.e. $1-p$ is the probability that the honest party, due to crash failures or DoS attacks, fails to react. We show that (1) to discourage attacks by profit-driven parties, p for eltoo must be more significant than that of Daric, and (2) unlike Daric, an increase in the channel capacity in eltoo channels raises the minimum value of p that is required to prevent fraud. However, achieving large values of p (e.g. 0.9999) could be difficult for ordinary users. This indicates that eltoo needs a way to punish profit-driven attackers.

To monitor a channel, the watchtower's collateral equals the channel capacity [54, 23]. Let C denote the total capacity of the Bitcoin payment channel network and C_W denote the total capital that fair watchtowers have spent to watch their clients' channels. Then, the probability that a randomly selected payment channel is monitored by a fair watchtower is roughly computed as $\frac{C_W}{C}$.

Assume that a dishonest party $\mathscr{A}$ creates an eltoo channel with a channel capacity of $C_{\mathscr{A}}$ coins, where the initial balance of $\mathscr{A}$ and her counterparty are $C_{\mathscr{A}}$ and 0, respectively. For now, we assume that parties know if their counterparties are using a fair watchtower. We will relax this assumption later. If the channel is being monitored by a fair watchtower, $\mathscr{A}$ continues using the channel in an honest way. Otherwise, she sends all her balance to her counterparty in exchange for some products or services and then submits the initial channel state to the blockchain. In such a case, with a probability of $1-p$ and p, $\mathscr{A}$'s revenue and her loss would be $C_{\mathscr{A}}-f$ and f, respectively, where f denotes the transaction fee. Thus, $\mathscr{A}$ is discouraged to attack iff:

$$(C_{\mathscr{A}}-f)(1-p)-f\cdot p<0 \Leftrightarrow p>1-\frac{f}{C_{\mathscr{A}}}.$$

For a Daric channel, $\mathscr{A}$ is discouraged to attack iff:

$$0.99 \cdot C_{\mathscr{A}} \cdot (1-p) - 0.01 \cdot C_{\mathscr{A}} \cdot p < 0 \Leftrightarrow p > 0.99.$$

The threshold value for eltoo is typically more significant than that of Daric. At the time of writing this chapter, the average values of f for a transaction and $C_{\mathscr{A}}$ for a Lightning channel are around 0.000042 BTC and 0.069 BTC, respectively, leading to $1 - \frac{f}{C_{\mathscr{A}}} \approx 0.999$. But the adversary can practically set f to the lowest possible value (i.e. 1 Satoshi per VByte) leading to $f \approx 0.0000021$[2] BTC and $1 - \frac{f}{C_{\mathscr{A}}} \approx 0.9999$ for eltoo. Therefore, (1) to discourage attacks, the honest party would require to meet a higher p in eltoo than in Daric, (2) the threshold for eltoo depends on the channel capacity, and (3) the threshold for Daric can simply decrease from 0.99 to lower values.

In the above analysis, we assumed that $\mathscr{A}$ knows whether her counterparty is hiring any fair watchtower. Considering the opposite case, the probability that the channel is not being monitored by any fair watchtower and the honest party fails to react upon fraud would be $p_0 := (1 - \frac{C_W}{C})(1-p)$. Thus, with a probability of p_0 and $1-p_0$, $\mathscr{A}$'s revenue and her loss in an eltoo channel would be $C_{\mathscr{A}} - f$ and f, respectively. Thus, $\mathscr{A}$ is discouraged to attack iff:

$$(C_{\mathscr{A}} - f) \cdot p_0 - f \cdot (1-p_0) < 0 \Leftrightarrow p > 1 - \frac{\frac{f}{C_{\mathscr{A}}}}{1 - \frac{C_W}{C}}.$$

Similarly, for a Daric channel, we have:

$$0.99 \cdot C_{\mathscr{A}} \cdot p_0 - 0.01 \cdot C_{\mathscr{A}} \cdot (1-p_0) < 0 \Leftrightarrow p > 1 - \frac{0.01}{1 - \frac{C_W}{C}}.$$

As explained earlier, the threshold value for eltoo depends on $C_{\mathscr{A}}$ and is typically more significant than that of Daric.

7.7 Performance Analysis

Table 7.2 shows the total number of weight units of transactions, published on the blockchain for different payment channels in different channel closure scenarios. Since the weight units of a transaction directly impact its fee, we use this parameter to compare different schemes. Payment channels perform similarly in the collaborative channel closure, so we do not consider this scenario in our analysis. Since the funding transaction

[2]Each update transaction in eltoo contains 332 byte of witness data and 125 bytes of non-witness data leading to 208 VBytes. See Appendix H in the full version of the Daric paper [63] for more details

is the same in all schemes, we do not involve it in our comparison results either. To do a consistent comparison, we assume that each transaction output is either P2WSH[3] or P2WPKH[4], each public key and signature are respectively 33 bytes and 73 bytes, shared outputs are implemented using the `OP_CHECKMULTSIG` opcode (rather than using multi-party signing), and each state contain m HTLC outputs with $0 \leq m \leq 966$ [72] where each party is the payer for $\frac{m}{2}$ HTLC outputs and the payee for the rest.

Once a dishonest party in a Lightning channel publishes a revoked commit transaction, $m+1$ revoked outputs are created. For simplicity, we assume that the victim claims all the revoked outputs through one transaction. Cerberus [23], Sleepy [28] and Outpost [22] have not explained ways HTLC is added to these schemes and discussing it is out of the scope of this chapter, so Table 7.2 contains their figures with $m = 0$.

As Table 7.2 shows, in the dishonest closure scenario, (1) the weight units for Lightning and eltoo increase linearly with the number of HTLC outputs m compared to Daric, Generalized and FPPW and (2) Daric (with weight unit equal to 1239) is more cost-effective than other schemes with $m \geq 1$. In the non-collaborative closure scenario with $m \neq 0$, Daric outperforms Generalized, eltoo and FPPW channels with any value of m and Lightning channel with $m > 6$.

Table 7.2 also compares the number of operations performed by each party for a channel update. To count the operations, we additionally assume that (1) channel parties delegate the monitoring task to a watchtower and (2) they do not compute a signature unless it is supposed to be sent to their counterparty or their watchtower. Appendix H in the full version of the Daric paper [63] provides complete details regarding the way figures of Table 7.2 have been computed (The full version paper is available at https://eprint.iacr.org/2022/1295.pdf).

7.8 Daric Transactions Scripts

Funding transaction has one output with the following script where `Com_pubkeyA` and `Com_pubkeyB` are public keys of A and B, respectively:

`2 ⟨Com_pubkeyA⟩ ⟨Com_pubkeyB⟩ 2 OP_CHECKMULTISIG`

Commit transactions have one input that takes the output of the funding transaction with the witness script of `0 ⟨Com_pubkeyA_sig⟩ ⟨Com_pubkeyB_sig⟩`. The commit transaction $\mathtt{TX}^{A}_{\mathtt{CM},i}$ has one output with the following script:

$\langle$`absolute time` $\mathtt{S}_0 + \mathtt{i}$ $\rangle$ `OP_CHECKLOCKTIMEVERIFY OP_DROP`

[3]Pay-to-Witness-Script-Hash: Used to lock bitcoin to a SegWit script hash.
[4]Pay-to-Witness-Public-Key-Hash: Used to lock bitcoin to a SegWit public key hash.

Table 7.2: On-chain cost of different closure scenarios and the number of operations performed by each party for a channel update for different payment channels with m HTLC outputs ($0 \leq m \leq 966$).

	dishonest closure		non-coll. closure		num. of operations		
Scheme	#Tx	weight units	#Tx	weight units	Sign	Verify	Exp.
Lightning [15]	≥ 2	$\geq$1209+582.5m	1+m	724+793m	2+2m	1+$\frac{m}{2}$	2
Generalized [14]	2	1342	2+m	1432+696m	3	2	1
FPPW [54]	2	2045	2+m	1562+696m	6	10	1
Cerberus [23]	2	1798	1	772	3	6	0
Outpost [22]	3	2632	3	3018	4	4	0
Sleepy [28]	3	2172	3	2558	5	5	0
eltoo [1]	3	2268+696m	2+m	1588+696m	2	2	1
Daric (this work)	2	1239	2+m	1363+696m	4	3	0

```
OP_IF
   # Revocation
   2 ⟨Rev_pubkeyA⟩ ⟨Rev_pubkeyB⟩ 2 OP_CHECKMULTISIG
OP_ELSE
   # Split
   ⟨delay t⟩ OP_CHECKSEQUENCEVERIFY OP_DROP
   2 ⟨Spl_pubkeyA⟩ ⟨Spl_pubkeyB⟩ 2 OP_CHECKMULTISIG
OP_ENDIF
```

where ⟨Rev_pubkeyA⟩ and ⟨Spl_pubkeyA⟩ are public keys of A and ⟨Rev_pubkeyB⟩ and ⟨Spl_pubkeyB⟩ are public keys of B. The script of the commit transaction $\mathsf{TX}^{B}_{\mathsf{CM},i}$ is similar to that of $\mathsf{TX}^{A}_{\mathsf{CM},i}$ but its revocation keys are ⟨Rev′_pubkeyA⟩ and ⟨Rev′_pubkeyB⟩.

The split transaction $\mathsf{TX}_{\mathsf{SP},i}$ spends the output of $\mathsf{TX}^{A}_{\mathsf{CM},i}$ or $\mathsf{TX}^{B}_{\mathsf{CM},i}$ with the witness script:

$$0\ \langle \mathsf{Spl\_pubkeyA\_Sig} \rangle\ \langle \mathsf{Spl\_pubkeyB\_Sig} \rangle\ 0$$

The revocation transactions $\mathsf{TX}^{A}_{\mathsf{RV},i}$ and $\mathsf{TX}^{B}_{\mathsf{RV},i}$ spend the output of a revoked commit transaction with the witness scripts

$$0\ \langle \mathsf{Rev'\_pubkeyA\_sig} \rangle\ \langle \mathsf{Rev'\_pubkeyB\_sig} \rangle\ 1$$

and

$$0\ \langle \mathsf{Rev\_pubkeyA\_sig} \rangle\ \langle \mathsf{Rev\_pubkeyB\_sig} \rangle\ 1,$$

respectively. The transactions $\mathtt{TX}^{A}_{\mathsf{RV},i}$ and $\mathtt{TX}^{B}_{\mathsf{RV},i}$ have one output with the scripts ⟨pubkeyA⟩ OP_CHECKSIG and ⟨pubkeyB⟩ OP_CHECKSIG, respectively.

7.9 UC Framework

We model the security of our protocol in the synchronous version of the global UC framework (GUC) [70] which is an extension of the standard UC framework [55]. In the synchronous version, we can have a global setup that is used to model the ledger. The model in this work closely follows that of some works on layer-2 solutions to the scalability of blockchains [14, 68, 69].

Let π be a protocol executed among parties of a set $\mathscr{P} = \{P_1, \dots, P_n\}$. Assume that there exists an adversary $\mathscr{A}$ that takes as input a security parameter $\lambda \in \mathbb{N}$ and an auxiliary input $z \in \{0,1\}^*$. Before execution of the protocol, π, the adversary $\mathscr{A}$ can select any party $P_i \in \mathscr{P}$ to learn their internal state and fully control them. Anything outside the protocol execution is modelled by the environment $\mathscr{E}$. Each protocol party as well as the adversary take their inputs from the environment. Outputs of all parties are also observed by the environment. There are also ideal functionalities $\mathscr{F}_1, \dots, \mathscr{F}_m$ whose functions might be called by parties. Then, the protocol π is denoted by $\pi^{\mathscr{F}_1,\dots,\mathscr{F}_m}$.

To simplify our model, we assume that the communication network is synchronous, meaning that we let the protocol be executed through several rounds. The ideal functionality $\mathscr{F}_{clock}$ represents a global clock that increases by one unit once all parties are prepared to proceed to the next round. All entities know the value of the current round.

We assume that parties of the protocol are connected to each other via an authenticated communication channel which guarantees that messages are delivered to recipient parties after exactly one round. In other words, if party P sends a message m to the party Q in round τ, the message reaches Q in the beginning of round $\tau + 1$ and Q can ensure that the sender is P. The adversary $\mathscr{A}$ observes the message m and can even change the order of messages that are sent in the same round. However, the adversary cannot drop, delay or change any transmitted message or insert new messages. The ideal functionality $\mathscr{F}_{GDC}$ models such a communication channel between the channel parties. Other communications in which some other entities, e.g. $\mathscr{A}$ or $\mathscr{E}$, are involved take zero rounds to complete. Moreover, to simplify the model, we assume that any required computation is also performed within zero rounds.

In this thesis, we focus on UTXO-based cryptocurrencies such as Bitcoin. The *global* ideal functionality $\mathscr{L}$ models such cryptocurrencies where it is parameterised by two parameters: (1) a digital signature scheme Σ, and (2) a delay parameter Δ which is an upper bound on the number of rounds it takes for a valid transaction that has been

posted to the blockchain network to be published on the blockchain. To simplify the model, we assume that the set of parties $\mathscr{P}$ is fixed and transactions are published one by one rather than being published on the blockchain in blocks. Badertscher et. al [73] provided a more accurate model of the Bitcoin blockchain.

The environment $\mathscr{E}$ initiates the ledger functionality $\mathscr{L}$ by (1) instructing $\mathscr{L}$ to generate the public parameters of the signature scheme Σ, (2) instructing each party $P \in \mathscr{P}$ to create a key pair (pk_P, sk_P) and submit its public key to $\mathscr{L}$, and (3) creating an initial state TX that contains all the accepted transactions. The set TX is accessible to everyone including the protocol parties, the environment and the adversary. Once a party $P \in \mathscr{P}$ posts a transaction tx to the blockchain, $\mathscr{L}$ waits for $\tau \leq \Delta$ rounds where τ is selected by the adversary. Then, if the validity of tx is successfully verified, it is added to the set TX.

Ideal Functionality $\mathscr{L}(\Delta, \Sigma)$

The set $\mathscr{P}$ defines the set of all parties who can send messages to the functionality. The functionality maintains the set PKI for the parties in $\mathscr{P}$. The sets TX and UTXO respectively define all the transactions accepted to date and all the unspent transaction outputs. The set of valid output conditions is represented by $\mathscr{V}$.

Public key Registration: Upon $(\text{register}, pk_P) \overset{\tau_0}{\hookleftarrow} P$, check if it is the first registration message received from $P \in \mathscr{P}$. If not drop the message, else add (pk_P, P) to PKI.

Post transaction: Upon $(\text{post}, \text{tx}) \overset{\tau_0}{\hookleftarrow} P$, check if $|\text{PKI}| = |\mathscr{P}|$, If not drop the message, else wait for $\tau \leq \Delta$ rounds where τ is selected by the adversary. Then, check if:

1. id uniqueness: For all $(t, \text{tx}') \in \text{TX}$, $\text{tx}'.\text{txid} \neq \text{tx.txid}$ holds.

2. Input and witness validity: For each $(txid \| i) \in \text{tx.Input}$, there exists $(t, txid, i, \theta) \in$ UTXO s.t. tx.Witness with inputs tx.nLT, the current round $\tau_0 + \tau$ and t satisfies $\theta.\varphi$.

3. Output validity: For each $\theta \in \text{tx.Output}$, $\theta.\text{Cash} > 0$ and $\theta.\varphi \in \mathscr{V}$ hold.

4. Value validity: Let $I := \{\text{utxo} := (t, txid, i, \theta) \mid \text{utxo} \in \text{UTXO} \wedge (txid \| i) \in \text{tx.Input}\}$. Then, $\Sigma_{\theta' \in \text{tx.Output}} \theta'.\text{cash} \leq \Sigma_{\text{utxo} \in I} \text{utxo}.\theta.\text{cash}$ holds.

5. Absolute time-lock validity: For the transaction tx, $\text{tx.nLT} \leq \tau_0 + \tau$ holds.

If any of the above checks fail, drop the message. Else, set $\mathsf{TX} := \mathsf{TX} \bigcup (\tau_0 + \tau, \mathsf{tx})$, $\mathsf{UTXO} := \mathsf{UTXO} \backslash I$ and $\mathsf{UTXO} := \mathsf{UTXO} \bigcup \{(\tau_0 + \tau, \mathsf{tx.txid}, i, \theta_i)\}_{i \in [n]}$ for $(\theta_1, \ldots, \theta_n) := \mathsf{tx.Output}$.

Let π be a protocol that has access to the global ledger $\mathcal{L}(\Delta, \Sigma)$ as well as the global clock $\mathcal{F}_{clock}$ and $\varphi_{\mathcal{F}}$ denote the ideal protocol for an ideal functionality $\mathcal{F}$ with access to the same global functionalities. Let $\mathrm{EXE}^{\mathcal{L}(\Delta,\Sigma),\mathcal{F}_{clock}}_{\pi,\mathcal{A},\mathcal{E}}(\lambda, z)$ denote the output of the environment $\mathcal{E}$ which interacts with a protocol π and an adversary $\mathcal{A}$ on input a security parameter λ and an auxiliary input z and similarly $\mathrm{EXE}^{\mathcal{L}(\Delta,\Sigma),\mathcal{F}_{clock}}_{\varphi_{\mathcal{F}},\mathcal{S},\mathcal{E}}(n, z)$ denote the output of the environment $\mathcal{E}$ which interacts with a protocol $\varphi_{\mathcal{F}}$ and an adversary $\mathcal{S}$ (also called the simulator) on input a security parameter λ and an auxiliary input z.

The following definition is informally saying that should a protocol π UC-realise $\mathcal{F}$, any attack against the protocol π can be transformed into an attack against the ideal protocol $\varphi_{\mathcal{F}}$ and vice versa.

Definition 7.1. *A protocol π UC-realises an ideal functionality $\mathcal{F}$ with respect to a global ledger $\mathcal{L}(\Delta, \Sigma)$ and a global clock $\mathcal{F}_{clock}$ if for every adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that we have*

$$\begin{aligned} &\{\mathrm{EXE}^{\mathcal{L}(\Delta,\Sigma),\mathcal{F}_{clock}}_{\pi,\mathcal{A},\mathcal{E}}(\lambda, z)\}_{n \in \mathbb{N}, z \in \{0,1\}^*} \approx \\ &\{\mathrm{EXE}^{\mathcal{L}(\Delta,\Sigma),\mathcal{F}_{clock}}_{\varphi_{\mathcal{F}},\mathcal{S},\mathcal{E}}(\lambda, z)\}_{n \in \mathbb{N}, z \in \{0,1\}^*} \end{aligned} \tag{7.1}$$

where $\approx$ denotes computational indistinguishability.

7.10 Ideal Functionality

This section defines an ideal functionality $\mathcal{F}(t)$ with $t > \Delta$ that achieves the desired properties stated in Section 7.5.2. To simplify the notations, we abbreviate $\mathcal{F} := \mathcal{F}(t)$. The ideal functionality $\mathcal{F}$ stores a set Γ of all the created channels and their corresponding funding transactions. The set Γ can also be treated as a function s.t. $\Gamma(id) = (\gamma, \mathsf{TX})$ with $\gamma.\mathrm{id} = id$ if γ exists and $\Gamma(id) = \perp$ otherwise. Before presenting the ideal functionality $\mathcal{F}$ in detail, we briefly introduce its different phases and explain the way $\mathcal{F}$ achieves the desired properties.

a) Create: In this phase, $\mathcal{F}$ receives messages $(\mathtt{INTRO}, \gamma, tid_P)$ and $(\mathtt{CREATE}, \gamma.\mathrm{id})$ from both parties in rounds τ_0 and $\tau_0 + 1$, respectively, where tid_P specifies the funding source of the user P. Then, if the corresponding funding transaction appears on the ledger $\mathcal{L}$ within $2 + \Delta$ rounds, $\mathcal{F}$ sends the message $(\mathtt{CREATED}, \gamma.\mathrm{id})$ to both parties and stores γ and the funding transaction in $\Gamma(\gamma.\mathrm{id})$. If the CREATE message is not received from both

parties but the funding transaction appears on $\mathscr{L}$ within $2+\Delta$ rounds, $\mathscr{F}$ outputs Error. Since the message CREATED might be sent to the parties only if they both have sent the message CREATE to $\mathscr{F}$, the ideal functionality achieves consensus on creation.

b) Update: One of the parties, denoted by P, initiates this phase by sending the message $(\texttt{UPDATE}, id, \vec{\theta}, t_{stp})$ to $\mathscr{F}$, where id is the channel identifier, $\vec{\theta}$ is the new channel state and t_{stp} is the number of rounds needed to prepare prerequisites of the channel update (e.g. preparing the needed HTLCs). Due to disagreeing with the new state or failure in preparing its prerequisites, party Q can stop it by not sending the message $(\texttt{UPDATE}-\texttt{OK}, id)$ in step 2. Abort by P or Q in the next steps causes the procedure ForceClose(id) to be executed. The property optimistic update is satisfied because if both parties act honestly, the channel can be updated without any blockchain interaction. Furthermore, if P or Q disagree to update the channel, they can stop sending the UPDATE or $\texttt{UPDATE}-\texttt{OK}$ messages, respectively. This stops the channel update process without changing the latest channel state. Also, in cases where either P or Q stop cooperating, the procedure ForceClose(id) is executed. This procedure takes at most Δ rounds to complete. This also guarantees consensus on update.

c) Close: If $\mathscr{F}$ receives the message $(\texttt{CLOSE}, id)$ from both parties, a transaction TX is expected to appear on $\mathscr{L}$ within $\Delta+1$ rounds. This transaction spends the output of the funding transaction and its outputs equal the latest channel state γ.st. If the CLOSE message is received only from one of the parties, $\mathscr{F}$ executes the procedure ForceClose(id). In both cases, the output of the funding transaction must be spent within $\Delta+1$ rounds. Otherwise, $\mathscr{F}$ outputs Error.

d) Punish: If a transaction TX spends the funding transaction's output of a channel γ, one of the following events is expected to occur: (1) another transaction appears on $\mathscr{L}$ within Δ rounds where this transaction spends output of TX and sends γ.cash coins to the honest party P; or (2) another transaction whose outputs correspond to the channel state γ.st or γ.st′ appears on $\mathscr{L}$ within $t+\Delta$ rounds. Otherwise, $\mathscr{F}$ outputs Error. According to its definition, bounded closure with punish is achieved, if $\mathscr{F}$ returns no Error in the close and punish phases.

We describe the ideal functionality below. Normally, once $\mathscr{F}$ receives a message, it performs several validations on the message. But to simplify the description, we assume that messages are well-formed. Data exchange between $\mathscr{F}$ and other parties is represented by directed arrows. If $\mathscr{F}$ sends the message m to party P in round τ_0, we denote it with $m \overset{\tau_0}{\hookrightarrow} P$. Similarly, if $\mathscr{F}$ is supposed to receive the message m from party P in round τ_0, we denote it with $m \overset{\tau_0}{\hookleftarrow} P$.

Ideal Functionality $\mathscr{F}(t)$

Create

upon $(\texttt{INTRO}, \gamma, tid_P) \overset{\tau_0}{\hookleftarrow} P$:

- If $(\texttt{INTRO}, \gamma, tid_Q) \overset{\tau_0}{\hookleftarrow} Q$, then continue. Else stop.
- If $(\texttt{CREATE}, id) \overset{\tau_0+1}{\longleftrightarrow} \gamma.users$:
  - Wait if in round $\tau_1 \leq \tau_0 + 3 + \Delta$ a transaction $\mathsf{TX}_{\mathsf{FU}}$ with $\mathsf{TX}_{\mathsf{FU}}.\mathsf{Input} = (tid_P, tid_Q)$ and $\mathsf{TX}_{\mathsf{FU}}.\mathsf{Output} = \{(\gamma.\mathsf{cash}, \varphi)\}$ appears on the ledger $\mathscr{L}$. If yes, set $\Gamma(\gamma.\mathrm{id}) := (\gamma, \mathsf{TX}_{\mathsf{FU}})$ and $(\texttt{CREATED}, \gamma.\mathrm{id}) \overset{\tau_1}{\hookrightarrow} \gamma.users$. Else stop.

  Otherwise:

  - Wait if in round $\tau_1 \leq \tau_0+3+\Delta$ a transaction $\mathsf{TX}_{\mathsf{FU}}$ with $\mathsf{TX}_{\mathsf{FU}}.\mathsf{Input} = (tid_P, tid_Q)$ and $\mathsf{TX}_{\mathsf{FU}}.\mathsf{Output} = \{(\gamma.\mathsf{cash}, \varphi)\}$ appears on the ledger $\mathscr{L}$. If yes, Output $\texttt{Error} \overset{\tau_1}{\hookrightarrow} \gamma.\mathrm{users}$. Else, stop.

Update

Upon $(\texttt{UPDATE}, id, \vec{\theta}, t_{stp}) \overset{\tau_0}{\hookleftarrow} P$, parse $(\gamma, \mathsf{TX}) := \Gamma(id)$ and proceed as follows:

1. Send $(\texttt{UPDATE}-\texttt{REQ}, id, \vec{\theta}, t_{stp}) \overset{\tau_0+1}{\hookrightarrow} Q$.
2. If $(\texttt{UPDATE}-\texttt{OK}, id) \overset{\tau_1 \leq \tau_0+1+t_{stp}}{\longleftarrow} Q$, then set $\gamma.\mathrm{flag} := 2$ and $\gamma.\mathrm{st}' := \vec{\theta}$ and send $(\texttt{SETUP}, id) \overset{\tau_1+1}{\hookrightarrow} P$. Else stop.
3. If $(\texttt{SETUP}-\texttt{OK}, id) \overset{\tau_1+1}{\hookleftarrow} P$, then $(\texttt{SETUP}', id) \overset{\tau_1+2}{\hookrightarrow} Q$. Else $\texttt{ForceClose}(id)$ and stop.
4. If $(\texttt{SETUP}'-\texttt{OK}, id) \overset{\tau_1+2}{\hookleftarrow} Q$, then $(\texttt{UPDATE}-\texttt{OK}, id) \overset{\tau_1+3}{\hookrightarrow} P$. Else execute $\texttt{ForceClose}(id)$ and stop.
5. If $(\texttt{REVOKE}, id) \overset{\tau_1+3}{\hookleftarrow} P$, then $(\texttt{REVOKE}-\texttt{REQ}, id) \overset{\tau_1+4}{\hookrightarrow} Q$. Else execute $\texttt{ForceClose}(id)$ and stop.
6. If $(\texttt{REVOKE}', id) \overset{\tau_1+4}{\hookleftarrow} Q$, set $\gamma.st := \vec{\theta}$, $\gamma.\mathrm{flag} := 1$, $\gamma.\mathrm{st}' := \perp$, $\gamma.\mathrm{sn} := \gamma.\mathrm{sn} + 1$, $\Gamma(id) := (\gamma, \mathsf{TX})$, $(\texttt{UPDATED}, id) \overset{\tau_1+5}{\hookrightarrow} \gamma.\mathrm{Users}$ and stop. Else execute $\texttt{ForceClose}(id)$ and stop.

Close

upon $(\texttt{CLOSE}, id) \overset{\tau_0}{\hookleftarrow} P$, distinguish:

Both agreed: If $(\texttt{CLOSE}, id) \overset{\tau_0}{\hookleftarrow} Q$, let $(\gamma, \texttt{TX}_{\texttt{FU}}) := \Gamma(id)$ and distinguish:

- If in round $\tau_1 \leq \tau_0 + 1 + \Delta$, $\texttt{TX}_{\overline{\texttt{SP}}}$, with $\texttt{TX}_{\overline{\texttt{SP}}}.\texttt{Output} = \gamma.\texttt{st}$ and $\texttt{TX}_{\overline{\texttt{SP}}}.\texttt{Input} = \texttt{TX}_{\texttt{FU}}.\texttt{txid}\|1$ appears on $\mathscr{L}$, set $\Gamma(id) := (\bot, \texttt{TX}_{\texttt{FU}})$, $(\texttt{CLOSED}, id) \overset{\tau_1}{\hookrightarrow} \gamma.\texttt{users}$ and stop.
- If in round $\tau_0 + 1 + \Delta$, the $\texttt{TX}_{\texttt{FU}}$ is still unspent, output $\texttt{Error} \overset{\tau_0+1+\Delta}{\hookrightarrow} \gamma.\texttt{users}$ and stop.

Q **disagreed:** Else, execute $\texttt{ForceClose}(id)$ in round $\tau_0 + 1$.

Punish (executed at the end of every round τ_0)

For each $(\gamma_i, \texttt{TX}_i) \in \Gamma$ check if there is a transaction $\texttt{TX}$ on the ledger $\mathscr{L}$ s.t. $\texttt{TX}.\texttt{Input} = \texttt{TX}_i.\texttt{txid}\|1$ and $\gamma_i \neq \bot$. If yes, distinguish:

1. **Punish**: For the honest $P \in \gamma_i.\texttt{users}$, in round $\tau_1 \leq \tau_0 + \Delta$, a transaction $\texttt{TX}_j$ with $\texttt{TX}_j.\texttt{Input} = \texttt{TX}.\texttt{txid}\|1$ and $\texttt{TX}_j.\texttt{Output} = (\gamma.\texttt{cash}, pk_P)$ appears on $\mathscr{L}$. Then, $(\texttt{PUNISHED}, id) \overset{\tau_1}{\hookrightarrow} P$, set $\Gamma(id) := (\bot, \texttt{TX}_i)$ and stop.
2. **Close**: In round $\tau_1 \leq \tau_0 + t + \Delta$ a transaction $\texttt{TX}_j$ appears on $\mathscr{L}$ where one of the following two sets of conditions hold: (1) $\gamma.\texttt{flag} = 1$, $\texttt{TX}_j.\texttt{Input} = \texttt{TX}.\texttt{txid}\|1$ and $\texttt{TX}_j.\texttt{Output} = \gamma.\texttt{st}$ or (2) $\gamma.\texttt{flag} = 2$, $\texttt{TX}_j.\texttt{Input} = \texttt{TX}.\texttt{txid}\|1$ and either $\texttt{TX}_j.\texttt{Output} = \gamma.\texttt{st}$ or $\texttt{TX}_j.\texttt{Output} = \gamma.\texttt{st}'$. Then, set $\Gamma(id) := (\bot, \texttt{TX}_i)$ and $(\texttt{CLOSED}, id) \overset{\tau_1}{\hookrightarrow} \gamma.\texttt{users}$.
3. **Error**: Otherwise, $\texttt{Error} \overset{\tau_0+t+\Delta}{\hookrightarrow} \gamma.\texttt{users}$.

Subprocedure $\texttt{ForceClose}(id)$

Let τ_0 be the current round and $(\gamma, \texttt{TX}_{\texttt{FU}}) := \Gamma(id)$. If within Δ rounds, $\texttt{TX}_{\texttt{FU}}.\texttt{Output}$ is still an unspent output on $\mathscr{L}$, then output $\texttt{Error} \overset{\tau_0+\Delta}{\hookrightarrow} \gamma.\texttt{users}$.

7.10.1 Functionality Wrapper

The functionality $\mathscr{F}$ is supposed to perform several checks once he receives a message from another party. The functionality $\mathscr{F}$ must perform those checks in order to ensure that the received messages are well-formed. The following wrapper summarises those checks.

Functionality Wrapper:$\mathscr{W}_{\mathscr{F}}$

Create

Upon $(\mathtt{INTRO}, \gamma, tid_P) \overset{\tau_0}{\hookleftarrow} P$ check if: $P \in \gamma.\mathsf{users}$; $\Gamma(\gamma.\mathsf{id}) \neq \perp$; there is no channel γ' with $\gamma.\mathsf{id} = \gamma'.\mathsf{id}$, $\gamma.\mathsf{sn} = 0$; $\gamma.\mathsf{st} = \{(c_P, \mathtt{One}-\mathtt{Sig}_{pk_P}), (c_Q, \mathtt{One}-\mathtt{Sig}_{pk_Q})\}$ with $c_P, c_Q \in \mathbb{R}^{>0}$ and $c_P + c_Q = \gamma.\mathsf{cash}$; there exist $(t, id, i, \theta) \in \mathscr{L}.\mathtt{UTXO}$ such that $\theta = (c_P, \mathtt{One}-\mathtt{Sig}_P)$ with $id \| i = tid$; and none of the other channels that are being created at the moment, must use tid_P. Drop the message if any above checks fail. Else proceed as $\mathscr{F}$.

Upon $(\mathtt{CREATE}, id) \overset{\tau}{\hookleftarrow} P$ check if: you accepted messages $(\mathtt{INTRO}, \gamma, tid_P) \overset{\tau_0}{\hookleftarrow} P$ and $(\mathtt{INTRO}, \gamma, tid_Q) \overset{\tau_0}{\hookleftarrow} Q$ with $P, Q \in \gamma.\mathsf{users}$, $\tau_0 + 1 = \tau$ and $id = \gamma.\mathsf{id}$. Else proceed as $\mathscr{F}$.

Update

Upon $(\mathtt{UPDATE}, id, \vec{\theta}, t_{stp}) \overset{\tau_0}{\hookleftarrow} P$ set $(\gamma, \mathsf{TX}_{\mathsf{FU}}) := \Gamma(id)$ and check if: $\gamma \neq \perp$, there is no other update being preformed; let $\vec{\theta} = (\theta_1, \ldots, \theta_l) = ((c_1, \varphi_1), \ldots, (c_l, \varphi_l))$, then $\Sigma_{i \in [l] c_i = \gamma.\mathsf{cash}}$ and $\varphi_i \in \mathscr{L}.\mathscr{V}$. Drop the message if any above checks fail. Else proceed as $\mathscr{F}$.

Upon $(\mathtt{UPDATE}-\mathtt{OK}, id) \overset{\tau}{\hookleftarrow} P$ check if: the message is a reply to the message $(\mathtt{UPDATE}-\mathtt{REQ}, id, \vec{\theta}, t_{stp})$ sent to P in round τ. If not, drop the message. Else proceed as $\mathscr{F}$.

Upon $(\mathtt{SETUP}-\mathtt{OK}, id) \overset{\tau}{\hookleftarrow} P$ check if: the message is a reply to the message $(\mathtt{SETUP}, id)$ sent to P in round τ_0 where $\tau = \tau_0 + t_{stp}$. If not, drop the message. Else proceed as $\mathscr{F}$.

Upon $(\mathtt{SETUP}'-\mathtt{OK}, id) \overset{\tau}{\hookleftarrow} P$ check if: the message is a reply to the message $(\mathtt{SETUP}', id)$ sent to P in round τ. If not, drop the message. Else proceed as $\mathscr{F}$.

Upon $(\mathtt{REVOKE}, id) \overset{\tau}{\hookleftarrow} P$ check if: the message is a reply to the message $(\mathtt{UPDATE}-\mathtt{OK}, id)$ sent to P in round τ. If not, drop the message. Else proceed as $\mathscr{F}$.

Upon $(\mathtt{REVOKE}', id) \overset{\tau}{\hookleftarrow} P$ check if: the message is a reply to the message $(\mathtt{REVOKE-REQ}, id)$ sent to P in round τ. If not, drop the message. Else proceed as $\mathscr{F}$.

Close

Upon $(\mathtt{CLOSE}, id) \overset{\tau}{\hookleftarrow} P$, set $(\gamma, \mathtt{TX}_{\mathsf{FU}}) := \Gamma(id)$ and check if: $P \in \gamma.\text{users}$, $\Gamma(\gamma.\text{id}) \neq \perp$ and $\gamma.\text{flag} = 1$. Drop the message if any above checks fail. Else proceed as $\mathscr{F}$.

7.11 Daric Protocol

In this section, details of different phases of Daric will be presented. Before presenting the protocol, some notations are introduced. In different steps of the protocol, channel participants generate (or verify) some signatures on protocol transactions. When a signature with SIGHASH of type SIGHASH_ALL or ANYPREVOUT is going to be generated (or verified) for the transaction TX, the input message to the signing (or verification) algorithm is denoted by $f(\mathtt{TX})$ or $\hat{f}(\overline{[\mathtt{TX}]})$, respectively [58].

The set Γ^P, maintained by each party $P \in \mathscr{P}$, stores information of the latest channel state for all the open channels that P is involved in, where $\Gamma^P(id)$ corresponds with the channel with the identifier of id. In the channel update phase, while $\gamma.\text{flag} = 2$, the channel has two active states where the information about the new state is maintained by $P \in \mathscr{P}$ in $\Gamma'^P(id)$ with $id = \gamma.\text{id}$. To refer to the i^{th} element of $\Gamma'^P(id)$ we use $\Gamma^P(id)[i]$ with $i \geq 1$. The party P also maintains a signature from his counterparty on the latest revocation transaction for each open channel that P is a party of. These signatures are maintained by P in the set Θ^P where $\Theta^P(id)$ corresponds with the channel with the identifier of id. We use directional arrows to show an exchange of messages. To simplify the protocol description, we remove some validations that must be normally done by channel parties. The protocol wrapper $\mathscr{W}_{\mathscr{P}}$ in Section 7.11.1 defines those validations.

Daric protocol: π

Create

Party P upon $(\mathtt{INTRO}, \gamma, tid_P) \overset{\tau_0}{\hookleftarrow} \mathscr{E}$

1. Set $id := \gamma.\text{id}$, generate $(pk^P_{\mathsf{SP}}, sk^P_{\mathsf{SP}}) \leftarrow \text{Gen}$, $(pk^P_{\mathsf{RV}}, sk^P_{\mathsf{RV}}) \leftarrow \text{Gen}$ and $(pk'^P_{\mathsf{RV}}, sk'^P_{\mathsf{RV}}) \leftarrow \text{Gen}$ and send $(\text{createInfo}, id, tid_P, pk^P_{\mathsf{SP}}, pk^P_{\mathsf{RV}}, pk'^P_{\mathsf{RV}}) \overset{\tau_0}{\hookrightarrow} Q$.

2. If $(\text{createInfo}, id, tid_Q, pk^Q_{\mathsf{SP}}, pk^Q_{\mathsf{RV}}, pk'^Q_{\mathsf{RV}}) \overset{\tau_0+1}{\hookleftarrow} Q$, create:

$$
\begin{aligned}
[\mathsf{TX}_{\mathsf{FU}}] &:= \mathsf{GenFund}((tid_P, tid_Q), \gamma) \\
([\mathsf{TX}^P_{\mathsf{CM},0}], [\mathsf{TX}^Q_{\mathsf{CM},0}]) &:= \\
&\mathsf{GenCommit}([\mathsf{TX}_{\mathsf{FU}}].\text{txid} \| 1, I_P, I_Q, 0) \\
\overline{[\mathsf{TX}_{\mathsf{SP},0}]} &:= \mathsf{GenSplit}(\gamma.\text{st}, 0)
\end{aligned}
$$

   for $I_P := (pk^P_{\mathsf{SP}}, pk^P_{\mathsf{RV}}, pk'^P_{\mathsf{RV}})$ and $I_Q := (pk^Q_{\mathsf{SP}}, pk^Q_{\mathsf{RV}}, pk'^Q_{\mathsf{RV}})$. Else stop.

3. Compute $\hat{\sigma}^P_{\mathsf{TX}_{\mathsf{SP},0}} := \text{Sign}_{sk^P_{\mathsf{SP}}}(\hat{f}(\overline{[\mathsf{TX}_{\mathsf{SP},0}]}))$ and $\sigma^P_{\mathsf{TX}^Q_{\mathsf{CM},0}} := \text{Sign}_{sk_P}(f([\mathsf{TX}^Q_{\mathsf{CM},0}]))$ and send $(\text{createCom}, id, \hat{\sigma}^P_{\mathsf{TX}_{\mathsf{SP},0}}, \sigma^P_{\mathsf{TX}^Q_{\mathsf{CM},0}}) \overset{\tau_0+1}{\hookrightarrow} Q$.

4. If $(\text{createCom}, id, \hat{\sigma}^Q_{\mathsf{TX}_{\mathsf{SP},0}}, \sigma^Q_{\mathsf{TX}^P_{\mathsf{CM},0}}) \overset{\tau_0+2}{\hookleftarrow} Q$, s.t. $\text{Vrfy}_{pk^Q_{\mathsf{SP}}}(\hat{f}(\overline{[\mathsf{TX}_{\mathsf{SP},0}]}); \hat{\sigma}^Q_{\mathsf{TX}_{\mathsf{SP},0}}) = 1$ and also $\text{Vrfy}_{pk_Q}(f([\mathsf{TX}^P_{\mathsf{CM},0}]); \sigma^Q_{\mathsf{TX}^P_{\mathsf{CM},0}}) = 1$, then $\sigma^P_{\mathsf{TX}_{\mathsf{FU}}} := \text{Sign}_{sk_P}(f([\mathsf{TX}_{\mathsf{FU}}]))$ and send $(\text{createFund}, id, \sigma^P_{\mathsf{TX}_{\mathsf{FU}}}) \overset{\tau_0+2}{\hookrightarrow} Q$. Else stop.

5. If $(\text{createFund}, id, \sigma^Q_{\mathsf{TX}_{\mathsf{FU}}}) \overset{\tau_0+3}{\hookleftarrow} Q$, s.t. $\text{Vrfy}_{pk_Q}(f([\mathsf{TX}_{\mathsf{FU}}]); \sigma^Q_{\mathsf{TX}_{\mathsf{FU}}}) = 1$, create the transaction $\mathsf{TX}_{\mathsf{FU}} := (\mathscr{H}([\mathsf{TX}_{\mathsf{FU}}]), [\mathsf{TX}_{\mathsf{FU}}], ((x, \sigma^P_{\mathsf{TX}_{\mathsf{FU}}}), (y, \sigma^Q_{\mathsf{TX}_{\mathsf{FU}}})))$ and $(\text{post}, \mathsf{TX}_{\mathsf{FU}}) \overset{\tau_0+3}{\hookrightarrow} \mathscr{L}$. Else, create a transaction TX with $\mathsf{TX}.\text{Input} := tid_P$ and $\mathsf{TX}.\text{Output}.\varphi := pk_P$ and $(\text{post}, \mathsf{TX}) \overset{\tau_0+3}{\hookrightarrow} \mathscr{L}$.

6. If $\mathsf{TX}_{\mathsf{FU}}$ is accepted by $\mathscr{L}$ in round $\tau_1 \leq \tau_0 + 3 + \Delta$, compute $\sigma^P_{\mathsf{TX}^P_{\mathsf{CM},0}} = \text{Sign}_{sk_P}(f([\mathsf{TX}^P_{\mathsf{CM},0}]))$, create $\mathsf{TX}^P_{\mathsf{CM},0} := (\mathscr{H}([\mathsf{TX}^P_{\mathsf{CM},0}]), [\mathsf{TX}^P_{\mathsf{CM},0}], (1, \{\sigma^P_{\mathsf{TX}^P_{\mathsf{CM},0}}, \sigma^Q_{\mathsf{TX}^P_{\mathsf{CM},0}}\})$, set $\overline{\mathsf{TX}_{\mathsf{SP},0}} := (\overline{[\mathsf{TX}_{\mathsf{SP},0}]}, (1, \{\hat{\sigma}^P_{\mathsf{TX}_{\mathsf{SP},0}}, \hat{\sigma}^Q_{\mathsf{TX}_{\mathsf{SP},0}}\}))$, store $\Gamma^P(\gamma.id) := (\gamma, \mathsf{TX}_{\mathsf{FU}}, \mathsf{TX}^P_{\mathsf{CM},0}, [\mathsf{TX}^Q_{\mathsf{CM},0}], \overline{\mathsf{TX}_{\mathsf{SP},0}})$ and $(\mathsf{CREATED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$. Else $\Gamma^P(\gamma.id) := (\bot, \mathsf{TX}_{\mathsf{FU}}, \bot, \bot, \bot)$ and stop.

Update

Party P upon $(\mathsf{UPDATE}, id, \vec{\theta}, t_{stp}) \overset{\tau_0}{\hookleftarrow} \mathscr{E}$

1. Send $(\text{updateReq}, id, \vec{\theta}, t_{stp}) \overset{\tau_0}{\hookrightarrow} Q$.

Party Q upon $(\text{updateReq}, id, \vec{\theta}, t_{stp}) \overset{t_0}{\hookleftarrow} P$

2. Send $(\mathtt{UPDATE-REQ}, id, \vec{\theta}, t_{stp}) \overset{t_0}{\hookrightarrow} \mathcal{E}$.

3. If $(\mathtt{UPDATE-OK}, id) \overset{t_1 \leq t_0 + t_{stp}}{\longleftarrow} \mathcal{E}$, then extract $\mathtt{TX}_{\mathsf{FU}}$ and $i := \gamma.\text{sn}$ from $\Gamma^Q(id)$, create $([\mathtt{TX}^P_{\mathsf{CM},i+1}], [\mathtt{TX}^Q_{\mathsf{CM},i+1}]) := \mathtt{GenCommit}([\mathtt{TX}_{\mathsf{FU}}].\text{txid} \| 1, I_P, I_Q, i+1)$, and $\overline{[\mathtt{TX}_{\mathsf{SP},i+1}]} := \mathtt{GenSplit}(\vec{\theta}, i+1)$ for $I_P := (pk^P_{\mathsf{SP}}, pk^P_{\mathsf{RV}}, pk'^P_{\mathsf{RV}})$ and $I_Q := (pk^Q_{\mathsf{SP}}, pk^Q_{\mathsf{RV}}, pk'^Q_{\mathsf{RV}})$, compute $\hat{\sigma}^Q_{\mathtt{TX}_{\mathsf{SP},i+1}} = \text{Sign}_{sk^Q_{\mathsf{SP}}}(\hat{f}(\overline{[\mathtt{TX}_{\mathsf{SP},i+1}]}))$, and send $(\text{updateInfo}, id, \hat{\sigma}^Q_{\mathtt{TX}_{\mathsf{SP},i+1}}) \overset{t_1}{\hookrightarrow} P$.

Party P upon $(\text{updateInfo}, id, \hat{\sigma}^Q_{\mathtt{TX}_{\mathsf{SP},i+1}}) \overset{\tau_1 \leq \tau_0 + 2 + t_{stp}}{\longleftarrow} Q$

4. Extract $\mathtt{TX}_{\mathsf{FU}}$ and $i := \gamma.sn$ from $\Gamma^P(id)$ and create the transactions $([\mathtt{TX}^P_{\mathsf{CM},i+1}], [\mathtt{TX}^Q_{\mathsf{CM},i+1}]) := \mathtt{GenCommit}([\mathtt{TX}_{\mathsf{FU}}].\text{txid} \| 1, I_P, I_Q, i+1)$ and $\overline{[\mathtt{TX}_{\mathsf{SP},i+1}]} := \mathtt{GenSplit}(\vec{\theta}, i+1)$ for $I_P := (pk^P_{\mathsf{SP}}, pk^P_{\mathsf{RV}}, pk'^P_{\mathsf{RV}})$ and $I_Q := (pk^Q_{\mathsf{SP}}, pk^Q_{\mathsf{RV}}, pk'^Q_{\mathsf{RV}})$. If $\text{Vrfy}_{pk^Q_{\mathsf{SP}}}(\hat{f}(\overline{[\mathtt{TX}_{\mathsf{SP},i+1}]}); \hat{\sigma}^Q_{\mathtt{TX}_{\mathsf{SP},i+1}}) = 1$, compute $\hat{\sigma}^P_{\mathtt{TX}_{\mathsf{SP},i+1}} := \text{Sign}_{sk^P_{\mathsf{SP}}}(\hat{f}(\overline{[\mathtt{TX}_{\mathsf{SP},i+1}]}))$, set $\overline{\mathtt{TX}_{\mathsf{SP},i+1}} := (\overline{[\mathtt{TX}_{\mathsf{SP},i+1}]}, (1, \{\hat{\sigma}^P_{\mathtt{TX}_{\mathsf{SP},i+1}}, \hat{\sigma}^Q_{\mathtt{TX}_{\mathsf{SP},i+1}}\}))$, store $\Gamma'^P(id) = (\perp, [\mathtt{TX}^Q_{\mathsf{CM},i+1}], \overline{\mathtt{TX}_{\mathsf{SP},i+1}})$, set $\gamma.\text{flag} = 2$ and $\gamma.\text{st}' = \vec{\theta}$ and send $(\mathtt{SETUP}, id) \overset{\tau_1}{\hookrightarrow} \mathcal{E}$. Else stop.

5. If $(\mathtt{SETUP-OK}, id) \overset{\tau_1}{\hookleftarrow} \mathcal{E}$, sign $\sigma^P_{\mathtt{TX}^Q_{\mathsf{CM},i+1}} := \text{Sign}_{sk_P}(f([\mathtt{TX}^Q_{\mathsf{CM},i+1}]))$ and output $(\text{updateComP}, id, \hat{\sigma}^P_{\mathtt{TX}_{\mathsf{SP},i+1}}, \sigma^P_{\mathtt{TX}^Q_{\mathsf{CM},i+1}}) \overset{\tau_1}{\hookrightarrow} Q$. Else, execute $\mathtt{ForceClose}^P(id)$ and stop.

Party Q

6. If $(\text{updateComP}, id, \hat{\sigma}^P_{\mathtt{TX}_{\mathsf{SP},i+1}}, \sigma^P_{\mathtt{TX}^Q_{\mathsf{CM},i+1}}) \overset{t_1+2}{\longleftarrow} P$, such that $\text{Vrfy}_{pk^P_{\mathsf{SP}}}(\hat{f}(\overline{[\mathtt{TX}_{\mathsf{SP},i+1}]}); \hat{\sigma}^P_{\mathtt{TX}_{\mathsf{SP},i+1}}) = 1$ and also $\text{Vrfy}_{pk_P}(f([\mathtt{TX}^Q_{\mathsf{CM},i+1}]); \sigma^P_{\mathtt{TX}^Q_{\mathsf{CM},i+1}}) = 1$, compute $\sigma^Q_{\mathtt{TX}^Q_{\mathsf{CM},i+1}} = \text{Sign}_{sk_Q}(f([\mathtt{TX}^Q_{\mathsf{CM},i+1}]))$, set $\overline{\mathtt{TX}_{\mathsf{SP},i+1}} := (\overline{[\mathtt{TX}_{\mathsf{SP},i+1}]}, (1, \{\hat{\sigma}^P_{\mathtt{TX}_{\mathsf{SP},i+1}}, \hat{\sigma}^Q_{\mathtt{TX}_{\mathsf{SP},i+1}}\}))$, set $\mathtt{TX}^Q_{\mathsf{CM},i+1} = ([\mathtt{TX}^Q_{\mathsf{CM},i+1}], (1, \{\sigma^P_{\mathtt{TX}^Q_{\mathsf{CM},i+1}}, \sigma^Q_{\mathtt{TX}^Q_{\mathsf{CM},i+1}}\}))$, $\gamma.\text{flag} = 2$ and $\gamma.\text{st}' = \vec{\theta}$, store $\Gamma'^Q(id) = (\mathtt{TX}^Q_{\mathsf{CM},i+1}, [\mathtt{TX}^P_{\mathsf{CM},i+1}], \overline{\mathtt{TX}_{\mathsf{SP},i+1}})$, and output $(\mathtt{SETUP}', id) \overset{t_1+2}{\hookrightarrow} \mathcal{E}$. Else, execute $\mathtt{ForceClose}^Q(id)$ and stop.

7. If $(\mathtt{SETUP}' - \mathtt{OK}, id) \overset{t_1+2}{\longleftarrow} \mathcal{E}$, sign $\sigma^Q_{\mathtt{TX}^P_{\mathsf{CM},i+1}} = \text{Sign}_{sk_Q}(f([\mathtt{TX}^P_{\mathsf{CM},i+1}]))$ and send $(\text{updateComQ}, id, \sigma^Q_{\mathtt{TX}^P_{\mathsf{CM},i+1}}) \overset{t_1+2}{\hookrightarrow} P$. Else, execute $\mathtt{ForceClose}^Q(id)$ and stop.

Party P

8. If $(\text{updateComQ}, id, \sigma^{Q}_{\mathsf{TX}^{P}_{\mathsf{CM},i+1}}) \xleftarrow{\tau_1+2} Q$, s.t. $\mathsf{Vrfy}_{pk_Q}(f([\mathsf{TX}^{P}_{\mathsf{CM},i+1}]); \sigma^{Q}_{\mathsf{TX}^{P}_{\mathsf{CM},i+1}}) = 1$, compute $\sigma^{P}_{\mathsf{TX}^{P}_{\mathsf{CM},i+1}} := \mathsf{Sign}_{sk_P}(f([\mathsf{TX}^{P}_{\mathsf{CM},i+1}], 1))$, set $\mathsf{TX}^{P}_{\mathsf{CM},i+1} := (\mathscr{H}([\mathsf{TX}^{P}_{\mathsf{CM},i+1}]), [\mathsf{TX}^{P}_{\mathsf{CM},i+1}], (1, \{\sigma^{P}_{\mathsf{TX}^{P}_{\mathsf{CM},i+1}}, \sigma^{Q}_{\mathsf{TX}^{P}_{\mathsf{CM},i+1}}\}))$, set $\Gamma'^{P}(id)[1] := \mathsf{TX}^{P}_{\mathsf{CM},i+1}$ and then output $(\mathsf{UPDATE}-\mathsf{OK}, id) \xhookrightarrow{\tau_1+2} \mathscr{E}$. Else, execute the procedure $\mathsf{ForceClose}^{P}(id)$ and stop.

9. If $(\mathsf{REVOKE}, id) \xleftarrow{\tau_1+2} \mathscr{E}$, create $(\overline{[\mathsf{TX}^{P}_{\mathsf{RV},i}]}, \overline{[\mathsf{TX}^{Q}_{\mathsf{RV},i}]}) := \mathsf{GenRevoke}(pk_P, pk_Q, \gamma.\text{cash}, i+1)$, compute $\hat{\sigma}^{P}_{\mathsf{TX}^{Q}_{\mathsf{RV},i}} := \mathsf{Sign}_{sk^{P}_{\mathsf{RV}}}(\hat{f}(\overline{[\mathsf{TX}^{Q}_{\mathsf{RV},i}]}))$ if $P = A$ or $\hat{\sigma}^{P}_{\mathsf{TX}^{Q}_{\mathsf{RV},i}} = \mathsf{Sign}_{sk'^{P}_{\mathsf{RV}}}(\hat{f}(\overline{[\mathsf{TX}^{Q}_{\mathsf{RV},i}]}))$ otherwise and send $(\text{revokeP}, id, \hat{\sigma}^{P}_{\mathsf{TX}^{Q}_{\mathsf{RV},i}}) \xhookrightarrow{\tau_1+2} Q$. Else, execute the procedure $\mathsf{ForceClose}^{P}(id)$ and stop.

Party Q

10. Create $(\overline{[\mathsf{TX}^{P}_{\mathsf{RV},i}]}, \overline{[\mathsf{TX}^{Q}_{\mathsf{RV},i}]}) := \mathsf{GenRevoke}(pk_P, pk_Q, \gamma.\text{cash}, i+1)$. If $(\text{revokeP}, id, \hat{\sigma}^{P}_{\mathsf{TX}^{Q}_{\mathsf{RV},i}}) \xleftarrow{t_1+4} P$, such that $\mathsf{Vrfy}_{pk^{P}_{\mathsf{RV}}}(\hat{f}([\mathsf{TX}^{Q}_{\mathsf{RV},i}]); \hat{\sigma}^{P}_{\mathsf{TX}^{Q}_{\mathsf{RV},i}}) = 1$ if $Q = B$ or $\mathsf{Vrfy}_{pk'^{P}_{\mathsf{RV}}}(\hat{f}([\mathsf{TX}^{Q}_{\mathsf{RV},i}]); \hat{\sigma}^{P}_{\mathsf{TX}^{Q}_{\mathsf{RV},i}}) = 1$ otherwise, set $\Theta^{Q}(id) := (\hat{\sigma}^{P}_{\mathsf{TX}^{Q}_{\mathsf{RV},i}}), \gamma.\text{sn} := i+1, \gamma.\text{st} := \vec{\theta}$, $\Gamma^{Q}(id) := (\gamma, \mathsf{TX}_{\mathsf{FU}}, \mathsf{TX}^{Q}_{\mathsf{CM},i+1}, [\mathsf{TX}^{P}_{\mathsf{CM},i+1}], \mathsf{TX}_{\mathsf{SP},i+1})$, $\gamma.\text{flag} = 1$, and $\Gamma'^{Q}(id) = (\bot, \bot, \bot)$ and send $(\mathsf{REVOKE}-\mathsf{REQ}, id) \xhookrightarrow{t_1+4} \mathscr{E}$. Else, execute the procedure $\mathsf{ForceClose}^{Q}(id)$ and stop.

11. If $(\mathsf{REVOKE}', id) \xleftarrow{t_1+4} \mathscr{E}$, then compute $\hat{\sigma}^{Q}_{\mathsf{TX}^{P}_{\mathsf{RV},i}} := \mathsf{Sign}_{sk^{Q}_{\mathsf{RV}}}(\hat{f}([\mathsf{TX}^{P}_{\mathsf{RV},i}]))$ if $Q = A$ or compute $\hat{\sigma}^{Q}_{\mathsf{TX}^{P}_{\mathsf{RV},i}} := \mathsf{Sign}_{sk'^{Q}_{\mathsf{RV}}}(\hat{f}([\mathsf{TX}^{P}_{\mathsf{RV},i}]))$ otherwise, output $(\text{revokeQ}, id, \hat{\sigma}^{Q}_{\mathsf{TX}^{P}_{\mathsf{RV},i}}) \xhookrightarrow{t_1+4} P$ and $(\mathsf{UPDATED}, id) \xhookrightarrow{t_1+5} \mathscr{E}$. Else, execute the procedure $\mathsf{ForceClose}^{Q}(id)$ and stop.

Party P

12. If $(\text{revokeQ}, id, \hat{\sigma}^{Q}_{\mathsf{TX}^{P}_{\mathsf{RV},i}}) \xleftarrow{\tau_1+4} Q$, such that it holds that $\mathsf{Vrfy}_{pk^{Q}_{\mathsf{RV}}}(\hat{f}([\mathsf{TX}^{P}_{\mathsf{RV},i}]); \hat{\sigma}^{Q}_{\mathsf{TX}^{P}_{\mathsf{RV},i}}) = 1$ if $P = B$ or $\mathsf{Vrfy}_{pk'^{Q}_{\mathsf{RV}}}(\hat{f}([\mathsf{TX}^{P}_{\mathsf{RV},i}]); \hat{\sigma}^{Q}_{\mathsf{TX}^{P}_{\mathsf{RV},i}}) = 1$ otherwise, assign $\Theta^{P}(id) := (\hat{\sigma}^{Q}_{\mathsf{TX}^{P}_{\mathsf{RV},i}})$, $\gamma.\text{sn} := i+1, \gamma.\text{st} := \vec{\theta}, \Gamma^{P}(id) := (\gamma, \mathsf{TX}_{\mathsf{FU}}, \mathsf{TX}^{P}_{\mathsf{CM},i+1}, [\mathsf{TX}^{Q}_{\mathsf{CM},i+1}], \overline{\mathsf{TX}_{\mathsf{SP},i+1}})$, $\gamma.\text{flag} := 1$, and $\Gamma'^{P}(id) := (\bot, \bot, \bot)$ and send $(\mathsf{UPDATED}, id) \xhookrightarrow{\tau_1+4} \mathscr{E}$. Else, execute the procedure $\mathsf{ForceClose}^{P}(id)$ and stop.

Close

Party P upon $(\mathtt{CLOSE}, id) \overset{\tau_0}{\hookleftarrow} \mathscr{E}$

1. Extract $\mathtt{TX}_{\mathsf{FU}}$, $i := \gamma.sn$, $\mathtt{TX}_{\mathsf{SP},i}$ from $\Gamma^P(id)$ and create $[\mathtt{TX}_{\overline{\mathsf{SP}}}] := \mathtt{GenFinSplit}(\mathtt{TX}_{\mathsf{FU}}.\mathrm{txid} \| 1, \gamma.\mathrm{st})$.
2. Compute $\sigma^P_{\mathtt{TX}_{\overline{\mathsf{SP}}}} := \mathrm{Sign}_{sk_P}(f([\mathtt{TX}_{\overline{\mathsf{SP}}}]))$ and send $(\mathrm{CloseP}, id, \sigma^P_{\mathtt{TX}_{\overline{\mathsf{SP}}}}) \overset{\tau_0}{\hookrightarrow} Q$.
3. If $(\mathrm{CloseQ}, id, \sigma^Q_{\mathtt{TX}_{\overline{\mathsf{SP}}}}) \overset{\tau_0+1}{\longleftarrow} Q$ s.t. $\mathrm{Vrfy}_{pk_Q}(f([\mathtt{TX}_{\overline{\mathsf{SP}}}]); \sigma^Q_{\mathtt{TX}_{\overline{\mathsf{SP}}}}) = 1$, create the transaction $\mathtt{TX}_{\overline{\mathsf{SP}}} := (\mathscr{H}([\mathtt{TX}_{\overline{\mathsf{SP}}}]), [\mathtt{TX}_{\overline{\mathsf{SP}}}], (1, \{\sigma^P_{\mathtt{TX}_{\overline{\mathsf{SP}}}}, \sigma^Q_{\mathtt{TX}_{\overline{\mathsf{SP}}}}\}))$ and send $(\mathrm{post}, \mathtt{TX}_{\overline{\mathsf{SP}}}) \overset{\tau_0+1}{\longrightarrow} \mathscr{L}$. Else, execute the procedure $\mathtt{ForceClose}^P(id)$ and stop.
4. If in round $\tau_1 \leq \tau_0 + 1 + \Delta$, the transaction $\mathtt{TX}_{\overline{\mathsf{SP}}}$ is accepted by $\mathscr{L}$, set $\Gamma^P(id) := \perp$, $\Theta^P(id) := \perp$ and send $(\mathtt{CLOSED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$.

Punish (executed at the end of every round τ_0)

Party P

For each $id \in \{0,1\}^*$, extract $i := \gamma.\mathrm{sn}$ and $flag := \gamma.\mathrm{flag}$ from $\Gamma^P(id)$.

If $flag = 1$:

- Set $(\gamma, \mathtt{TX}_{\mathsf{FU}}, \mathtt{TX}^P_{\mathsf{CM},i}, [\mathtt{TX}^Q_{\mathsf{CM},i}], \overline{\mathtt{TX}_{\mathsf{SP},i}}) := \Gamma^P(id)$ and $I := \{[\mathtt{TX}^P_{\mathsf{CM},i}], [\mathtt{TX}^Q_{\mathsf{CM},i}]\}$. Check if $\mathtt{TX}_{\mathsf{FU}}$.Output is spent by a transaction $\mathtt{TX}$ s.t. $[\mathtt{TX}] \notin I$. If yes:
  - Create $(\overline{[\mathtt{TX}^P_{\mathsf{RV},i-1}]}, \overline{[\mathtt{TX}^Q_{\mathsf{RV},i-1}]}) := \mathtt{GenRevoke}(pk_P, pk_Q, \gamma.\mathrm{cash}, i)$ and then set $[\mathtt{TX}^P_{\mathsf{RV},i-1}] := (\mathtt{TX}.\mathrm{txid} \| 1, \overline{[\mathtt{TX}^P_{\mathsf{RV},i-1}]})$.
  - compute $\hat{\sigma}^P_{\mathtt{TX}^P_{\mathsf{RV},i-1}} = \mathrm{Sign}_{sk^P_{\mathsf{RV}}}(\hat{f}(\overline{[\mathtt{TX}^P_{\mathsf{RV},i-1}]}))$ if $P = B$ or $\hat{\sigma}^P_{\mathtt{TX}^P_{\mathsf{RV},i-1}} = \mathrm{Sign}_{sk'^P_{\mathsf{RV}}}(\hat{f}(\overline{[\mathtt{TX}^P_{\mathsf{RV},i-1}]}))$ otherwise.
  - Set $\hat{\sigma}^Q_{\mathtt{TX}^P_{\mathsf{RV},i-1}} := \Theta^P(id)$, and create $\mathtt{TX}^P_{\mathsf{RV},i-1} := (\mathscr{H}([\mathtt{TX}^P_{\mathsf{RV},i-1}]), [\mathtt{TX}^P_{\mathsf{RV},i-1}], (2, \{\hat{\sigma}^P_{\mathtt{TX}^P_{\mathsf{RV},i-1}}, \hat{\sigma}^Q_{\mathtt{TX}^P_{\mathsf{RV},i-1}}\}))$.

- Post $(\text{post}, \mathsf{TX}^P_{\mathsf{RV},i-1}) \overset{\tau_0}{\hookrightarrow} \mathscr{L}$.
- Let $\mathsf{TX}^P_{\mathsf{RV},i-1}$ be accepted by $\mathscr{L}$ in round $\tau_1 \leq \tau_0 + \Delta$. Set $\Theta^P(id) := \perp$, $\Gamma^P(id) := \perp$ and output $(\mathtt{PUNISHED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$.

If no, set $[\mathsf{TX}_{\mathsf{SP},i}] := (\mathsf{TX}.\text{txid}\|1, \overline{\mathsf{TX}_{\mathsf{SP},i}}.\mathsf{nLT}, \overline{\mathsf{TX}_{\mathsf{SP},i}}.\text{Output})$, set $\mathsf{TX}_{\mathsf{SP},i} := (\mathscr{H}([\mathsf{TX}_{\mathsf{SP},i}]), \mathsf{TX}.\text{txid}\|1, \overline{\mathsf{TX}_{\mathsf{SP},i}})$ and then post $(\text{post}, \mathsf{TX}_{\mathsf{SP},i}) \overset{\tau_0+t}{\hookrightarrow} \mathscr{L}$. Let TX be spent in round $\tau_1 \leq \tau_0 + t + \Delta$. Set $\Theta^P(id) := \perp$, $\Gamma^P(id) := \perp$ and output $(\mathtt{CLOSED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$.

If $flag = 2$:

- Set $(\gamma, \mathsf{TX}_{\mathsf{FU}}, \mathsf{TX}^P_{\mathsf{CM},i}, [\mathsf{TX}^Q_{\mathsf{CM},i}], \overline{\mathsf{TX}_{\mathsf{SP},i}}) := \Gamma^P(id)$, and $(\mathsf{TX}^P_{\mathsf{CM},i+1}, [\mathsf{TX}^Q_{\mathsf{CM},i+1}], \overline{\mathsf{TX}_{\mathsf{SP},i+1}}) := \Gamma'^P(id)$ and also $I := \{[\mathsf{TX}^P_{\mathsf{CM},i}], [\mathsf{TX}^Q_{\mathsf{CM},i}], [\mathsf{TX}^P_{\mathsf{CM},i+1}], [\mathsf{TX}^Q_{\mathsf{CM},i+1}]\}$. Check if $\mathsf{TX}_{\mathsf{FU}}.\text{Output}$ is spent by a transaction TX s.t. $[\mathsf{TX}] \notin I$. If yes:
  - Create $(\overline{[\mathsf{TX}^P_{\mathsf{RV},i-1}]}, \overline{[\mathsf{TX}^Q_{\mathsf{RV},i-1}]}) := \mathsf{GenRevoke}(pk_P, pk_Q, \gamma.\text{cash}, i)$. and then set $[\mathsf{TX}^P_{\mathsf{RV},i-1}] := (\mathsf{TX}.\text{txid}\|1, \overline{[\mathsf{TX}^P_{\mathsf{RV},i-1}]})$.
  - compute $\hat{\sigma}^P_{\mathsf{TX}^P_{\mathsf{RV},i-1}} := \text{Sign}_{sk^P_{\mathsf{RV}}}(\hat{f}(\overline{[\mathsf{TX}^P_{\mathsf{RV},i-1}]}))$ if $P = B$ or $\hat{\sigma}^P_{\mathsf{TX}^P_{\mathsf{RV},i-1}} := \text{Sign}_{sk'^P_{\mathsf{RV}}}(\hat{f}(\overline{[\mathsf{TX}^P_{\mathsf{RV},i-1}]}))$ otherwise.
  - Set $\hat{\sigma}^Q_{\mathsf{TX}^P_{\mathsf{RV},i-1}} := \Theta^P(id)$, and create $\mathsf{TX}^P_{\mathsf{RV},i-1} := (\mathscr{H}([\mathsf{TX}^P_{\mathsf{RV},i-1}]), [\mathsf{TX}^P_{\mathsf{RV},i-1}], (2, \{\hat{\sigma}^P_{\mathsf{TX}^P_{\mathsf{RV},i-1}}, \hat{\sigma}^Q_{\mathsf{TX}^P_{\mathsf{RV},i-1}}\}))$.
  - Post $(\text{post}, \mathsf{TX}^P_{\mathsf{RV},i-1}) \overset{\tau_0}{\hookrightarrow} \mathscr{L}$.
  - Let $\mathsf{TX}^P_{\mathsf{RV},i-1}$ be accepted by $\mathscr{L}$ in round $\tau_1 \leq \tau_0 + \Delta$. Set $\Theta^P(id) := \perp$, $\Gamma^P(id) := \perp$ and output $(\mathtt{PUNISHED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$.

  If no, set $[\mathsf{TX}_{\mathsf{SP},i+1}] := (\mathsf{TX}.\text{txid}\|1, \overline{\mathsf{TX}_{\mathsf{SP},i+1}}.\mathsf{nLT}, \overline{\mathsf{TX}_{\mathsf{SP},i+1}}.\text{Output})$, $\mathsf{TX}_{\mathsf{SP},i+1} := (\mathscr{H}([\mathsf{TX}_{\mathsf{SP},i+1}]), \mathsf{TX}.\text{txid}\|1, \overline{\mathsf{TX}_{\mathsf{SP},i+1}})$, wait for t rounds and post $(\text{post}, \mathsf{TX}_{\mathsf{SP},i+1}) \overset{\tau_0+t}{\hookrightarrow} \mathscr{L}$. Let $\mathsf{TX}_{\mathsf{SP},i+1}$ be accepted by $\mathscr{L}$ in round $\tau_1 \leq \tau_0 + t + \Delta$. Then, set $\Theta^P(id) := \perp$, $\Gamma^P(id) := \perp$ and output $(\mathtt{CLOSED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$.

Subprocedures

$\mathtt{ForceClose}^P(id)$:

Let τ_0 be the current round. Extract $i := \gamma.\mathsf{sn}$, $flag := \gamma.\mathsf{flag}$, $\mathsf{TX}^P_{\mathsf{CM},i}$ and $\overline{\mathsf{TX}_{\mathsf{SP},i}}$ from $\Gamma^P(id)$. Also, extract $\mathsf{TX}^P_{\mathsf{CM},i+1}$ and $\overline{\mathsf{TX}_{\mathsf{SP},i+1}}$ from $\Gamma'^P(id)$ if $flag = 2$.

If (1) $flag = 1$ or (2) $flag = 2$ and $\mathsf{TX}^{P}_{\mathsf{CM},i+1} = \perp$, then post $(\text{post}, \mathsf{TX}^{P}_{\mathsf{CM},i}) \overset{\tau_0}{\hookrightarrow} \mathscr{L}$. Otherwise, post $(\text{post}, \mathsf{TX}^{P}_{\mathsf{CM},i+1}) \overset{\tau_0}{\hookrightarrow} \mathscr{L}$.

(Publishing the corresponding split transaction takes place in the Punish phase.)

$\mathsf{GenFund}((tid_P, tid_Q), \gamma)$:

Return $[\mathsf{TX}_{\mathsf{FU}}]$ where $[\mathsf{TX}_{\mathsf{FU}}].\mathsf{Input} := (tid_P, tid_Q)$, $[\mathsf{TX}_{\mathsf{FU}}].\mathsf{nLT} := 0$ and $[\mathsf{TX}_{\mathsf{FU}}].\mathsf{Output} := (\gamma.\mathsf{Cash}, pk_P \wedge pk_Q)$

$\mathsf{GenCommit}([\mathsf{TX}_{\mathsf{FU}}].\mathsf{txid} \| 1, (pk^{P}_{\mathsf{SP}}, pk^{P}_{\mathsf{RV}}, pk'^{P}_{\mathsf{RV}}), (pk^{Q}_{\mathsf{SP}}, pk^{Q}_{\mathsf{RV}}, pk'^{Q}_{\mathsf{RV}}), i)$:

Return $[\mathsf{TX}^{P}_{\mathsf{CM},i}]$ and $[\mathsf{TX}^{Q}_{\mathsf{CM},i}]$ where $\mathsf{TX}^{P}_{\mathsf{CM},i}.\mathsf{nLT} := 0$, $\mathsf{TX}^{P}_{\mathsf{CM},i}.\mathsf{Input} := \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid} \| 1$, and $\mathsf{TX}^{P}_{\mathsf{CM},i}.\mathsf{Output} := \{(\mathsf{TX}_{\mathsf{FU}}.\mathsf{Output.cash}, (\varphi_1 \vee \varphi_2)\}$, $\mathsf{TX}^{Q}_{\mathsf{CM},i}.\mathsf{nLT} := 0$, $\mathsf{TX}^{Q}_{\mathsf{CM},i}.\mathsf{Input} := \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid} \| 1$, and $\mathsf{TX}^{Q}_{\mathsf{CM},i}.\mathsf{Output} := \{(\mathsf{TX}_{\mathsf{FU}}.\mathsf{Output.cash}, (\varphi_1 \vee \varphi'_2)\}$ with $\varphi_1 := (pk^{P}_{\mathsf{SP}} \wedge pk^{Q}_{\mathsf{SP}} \wedge CSV_t \wedge CLTV_{S_0+i})$, $\varphi_2 := (pk^{P}_{\mathsf{RV}} \wedge pk^{Q}_{\mathsf{RV}} \wedge CLTV_{S_0+i})$, $\varphi'_2 := (pk'^{P}_{\mathsf{RV}} \wedge pk'^{Q}_{\mathsf{RV}} \wedge CLTV_{S_0+i})$.

$\mathsf{GenSplit}(\vec{\theta}, i)$:

Return $\overline{[\mathsf{TX}_{\mathsf{SP},i}]}$ where $\overline{[\mathsf{TX}_{\mathsf{SP},i}]}.\mathsf{nLT} := S_0 + i$ and $\overline{[\mathsf{TX}_{\mathsf{SP},i}]}.\mathsf{Output} := \vec{\theta}$.

$\mathsf{GenRevoke}(pk_P, pk_Q, \gamma.\mathsf{cash}, i+1)$:

Return $\overline{[\mathsf{TX}^{P}_{\mathsf{RV},i}]}$ and $\overline{[\mathsf{TX}^{Q}_{\mathsf{RV},i}]}$ where $\overline{[\mathsf{TX}^{P}_{\mathsf{RV},i}]}.\mathsf{nLT} := S_0 + i$, $\overline{[\mathsf{TX}^{P}_{\mathsf{RV},i}]}.\mathsf{Output} := \{(\gamma.\mathsf{cash}, pk_P)\}$, $\overline{[\mathsf{TX}^{Q}_{\mathsf{RV},i}]}.\mathsf{nLT} := S_0 + i$ and $\overline{[\mathsf{TX}^{Q}_{\mathsf{RV},i}]}.\mathsf{Output} := \{(\gamma.\mathsf{cash}, pk_Q)\}$

7.11.1 Protocol Wrapper

Each party in Daric protocol π is supposed to perform several checks once he receives a message from another party. Parties perform those checks to ensure that the received messages are well-formed. The following wrapper summarises those checks.

Protocol Wrapper: $\mathscr{W}_{\mathscr{P}}$

Create

Upon $(\texttt{INTRO}, \gamma, tid_P) \overset{\tau_0}{\hookleftarrow} \mathscr{E}$ check if: $P \in \gamma.\text{users}$; $\Gamma(\gamma.\text{id}) \neq \perp$; there is no channel γ' with $\gamma.\text{id} = \gamma'.\text{id}$, $\gamma.\text{sn} = 0$; $\gamma.\text{st} = \{(c_P, \texttt{One} - \texttt{Sig}_{pk_P}), (c_Q, \texttt{One} - \texttt{Sig}_{pk_Q})\}$ with $c_P, c_Q \in \mathbb{R}^{>0}$ and $c_P + c_Q = \gamma.\text{cash}$; there exist $(t, id, i, \theta) \in \mathscr{L}.\texttt{UTXO}$ such that $\theta = (c_P, \texttt{One} - \texttt{Sig}_P)$ with $id \| i = tid$; and none of the other channels that are being created at the moment, must use tid_P. Drop the message if any above checks fails. Else proceed as P in Daric protocol.

Upon $(\texttt{CREATE}, id) \overset{\tau}{\hookleftarrow} \mathscr{E}$ check if: you accepted messages $(\texttt{INTRO}, \gamma, tid_P) \overset{\tau_0}{\hookleftarrow} P$ and $(\texttt{INTRO}, \gamma, tid_Q) \overset{\tau_0}{\hookleftarrow} Q$ with $P, Q \in \gamma.\text{users}$, $\tau_0 + 1 = \tau$ and $id = \gamma.\text{id}$. Else proceed as P in Daric protocol.

Update

Upon $(\texttt{UPDATE}, id, \vec{\theta}, t_{stp}) \overset{\tau_0}{\hookleftarrow} \mathscr{E}$ set $(\gamma, \texttt{TX}_{\texttt{FU}}) := \Gamma(id)$ and check if: $\gamma \neq \perp$, there is no other update being preformed; let $\vec{\theta} = (\theta_1, \ldots, \theta_l) = ((c_1, \varphi_1), \ldots, (c_l, \varphi_l))$, then $\Sigma_{i \in [l] c_i = \gamma.\text{cash}}$ and $\varphi_i \in \mathscr{L}.\mathscr{V}$. Drop the message if any above checks fails. Else proceed as P in Daric protocol.

Upon $(\texttt{UPDATE} - \texttt{OK}, id) \overset{\tau}{\hookleftarrow} \mathscr{E}$ check if: the message is a reply to the message $(\texttt{UPDATE} - \texttt{REQ}, id, \vec{\theta}, t_{stp})$ sent to $\mathscr{E}$ in round τ. If not, drop the message. Else proceed as P in Daric protocol.

Upon $(\texttt{SETUP} - \texttt{OK}, id) \overset{\tau}{\hookleftarrow} \mathscr{E}$ check if: the message is a reply to the message $(\texttt{SETUP}, id)$ sent to $\mathscr{E}$ in round τ_0 where $\tau = \tau_0 + t_{stp}$. If not, drop the message. Else proceed as P in Daric protocol.

Upon $(\texttt{SETUP}' - \texttt{OK}, id) \overset{\tau}{\hookleftarrow} \mathscr{E}$ check if: the message is a reply to the message $(\texttt{SETUP}', id)$ sent to $\mathscr{E}$ in round τ. If not, drop the message. Else proceed as P in Daric protocol.

Upon $(\texttt{REVOKE}, id) \overset{\tau}{\hookleftarrow} \mathscr{E}$ check if: the message is a reply to the message $(\texttt{UPDATE} - \texttt{OK}, id)$ sent to $\mathscr{E}$ in round τ. If not, drop the message. Else proceed as P in Daric protocol.

Upon $(\texttt{REVOKE}', id) \overset{\tau}{\hookleftarrow} \mathscr{E}$ check if: the message is a reply to the message $(\texttt{REVOKE} - \texttt{REQ}, id)$ sent to $\mathscr{E}$ in round τ. If not, drop the message. Else proceed as P in Daric protocol.

Close

Upon $(\texttt{CLOSE}, id) \overset{\tau}{\hookleftarrow} \mathscr{E}$, set $(\gamma, \texttt{TX}_{\texttt{FU}}) := \Gamma(id)$ and check if: $P \in \gamma.\text{users}$, $\Gamma(\gamma.\text{id}) \neq \perp$ and and $\gamma.\text{flag} = 1$. Drop the message if any above checks fails. Else proceed as P in Daric protocol.

7.12 Security Analysis

In this section, we prove the Theorem 7.1. To do so, a simulator for the protocol π in the ideal world is provided and then we formally show that the Daric protocol (introduced in Section 7.11) UC-realises the ideal functionality $\mathcal{F}$ (introduced in Section 7.10).

Simulator:

Create

Case A is honest and B is corrupted.

Upon A sending $(\texttt{INTRO}, \gamma, tid_A) \overset{\tau_0}{\hookrightarrow} \mathcal{F}$,

1. Set $id := \gamma.\text{id}$, generate $(pk^A_{\mathsf{SP}}, sk^A_{\mathsf{SP}}) \leftarrow \text{Gen}$, $(pk^A_{\mathsf{RV}}, sk^A_{\mathsf{RV}}) \leftarrow \text{Gen}$ and $(pk'^A_{\mathsf{RV}}, sk'^A_{\mathsf{RV}}) \leftarrow \text{Gen}$ and send $(\text{createInfo}, id, tid_A, pk^A_{\mathsf{SP}}, pk^A_{\mathsf{RV}}, pk'^A_{\mathsf{RV}}) \overset{\tau_0}{\hookrightarrow} B$.

2. If B sends $(\text{createInfo}, id, tid_B, pk^B_{\mathsf{SP}}, pk^B_{\mathsf{RV}}, pk'^B_{\mathsf{RV}}) \overset{\tau_0}{\hookrightarrow} A$, then $(\texttt{INTRO}, \gamma, tid_B) \overset{\tau_0}{\hookrightarrow} \mathcal{F}$ on behalf of B. Else stop.

3. If A sends $(\texttt{CREATE}, id) \overset{\tau_0+1}{\hookrightarrow} \mathcal{F}$, then create $[\mathsf{TX_{FU}}] := \texttt{GenFund}((tid_A, tid_B), \gamma)$, $([\mathsf{TX}^A_{\mathsf{CM},0}], [\mathsf{TX}^B_{\mathsf{CM},0}]) := \texttt{GenCommit}([\mathsf{TX_{FU}}].\text{txid} \| 1, I_A, I_B, 0)$, and $\overline{[\mathsf{TX}_{\mathsf{SP},0}]} := \texttt{GenSplit}(\gamma.\text{st}, 0)$ for $I_A := (pk^A_{\mathsf{SP}}, pk^A_{\mathsf{RV}}, pk'^A_{\mathsf{RV}})$ and $I_B := (pk^B_{\mathsf{SP}}, pk^B_{\mathsf{RV}}, pk'^B_{\mathsf{RV}})$. Else stop.

4. Compute $\hat{\sigma}^A_{\mathsf{TX}_{\mathsf{SP},0}} = \text{Sign}_{sk^A_{\mathsf{SP}}}(\hat{f}(\overline{[\mathsf{TX}_{\mathsf{SP},0}]}))$ and $\sigma^A_{\mathsf{TX}^B_{\mathsf{CM},0}} = \text{Sign}_{sk^A_{\mathsf{CM}}}(f([\mathsf{TX}^B_{\mathsf{CM},0}]))$ and send $(\text{createCom}, id, \hat{\sigma}^A_{\mathsf{TX}_{\mathsf{SP},0}}, \sigma^A_{\mathsf{TX}^B_{\mathsf{CM},0}}) \overset{\tau_0+1}{\hookrightarrow} B$.

5. If B sends $(\text{createCom}, id, \hat{\sigma}^B_{\mathsf{TX}_{\mathsf{SP},0}}, \sigma^B_{\mathsf{TX}^A_{\mathsf{CM},0}}) \overset{\tau_0+1}{\hookrightarrow} A$, s.t. $\text{Vrfy}_{pk^B_{\mathsf{SP}}}(\hat{f}(\overline{[\mathsf{TX}_{\mathsf{SP},0}]}); \hat{\sigma}^B_{\mathsf{TX}_{\mathsf{SP},0}}) = 1$ and $\text{Vrfy}_{pk^B_{\mathsf{CM}}}(f([\mathsf{TX}^A_{\mathsf{CM},0}]); \sigma^B_{\mathsf{TX}^A_{\mathsf{CM},0}}) = 1$, send $(\texttt{CREATE}, id) \overset{\tau_0+1}{\hookrightarrow} \mathcal{F}$ on behalf of B. Else stop.

6. Compute $\sigma^A_{\mathsf{TX_{FU}}} = \text{Sign}_{sk_A}(f([\mathsf{TX_{FU}}]))$ and send $(\text{createFund}, id, \sigma^A_{\mathsf{TX_{FU}}}) \overset{\tau_0+2}{\hookrightarrow} B$.

7. If B sends $(\text{createFund}, id, \sigma^B_{\mathsf{TX_{FU}}}) \overset{\tau_0+2}{\hookrightarrow} A$, s.t. $\text{Vrfy}_{pk_B}(f([\mathsf{TX_{FU}}]); \sigma^B_{\mathsf{TX_{FU}}}) = 1$, create $\mathsf{TX_{FU}} := (\mathscr{H}([\mathsf{TX_{FU}}]), [\mathsf{TX_{FU}}], ((x, \sigma^A_{\mathsf{TX_{FU}}}), (y, \sigma^B_{\mathsf{TX_{FU}}})))$ and $(\text{post}, \mathsf{TX_{FU}}) \overset{\tau_0+3}{\hookrightarrow} \mathcal{L}$. Else create a transaction TX with $\mathsf{TX}.\text{input} := tid_A$ and $\mathsf{TX}.\text{Output}.\varphi := pk_A$ and $(\text{post}, \mathsf{TX}) \overset{\tau_0+3}{\hookrightarrow} \mathcal{L}$.

8. If $\mathsf{TX}_{\mathsf{FU}}$ is accepted by $\mathscr{L}$ in round $\tau_1 \leq \tau_0 + 3 + \Delta$, compute $\sigma^A_{\mathsf{TX}^A_{\mathsf{CM},0}} = \mathrm{Sign}_{sk^A_{\mathsf{CM}}}(f([\mathsf{TX}^A_{\mathsf{CM},0}]))$, create $\mathsf{TX}^A_{\mathsf{CM},0} := (\mathscr{H}([\mathsf{TX}^A_{\mathsf{CM},0}]), [\mathsf{TX}^A_{\mathsf{CM},0}], (1, \{\sigma^A_{\mathsf{TX}^A_{\mathsf{CM},0}}, \sigma^B_{\mathsf{TX}^A_{\mathsf{CM},0}}\})$, set $\overline{\mathsf{TX}_{\mathsf{SP},0}} := (\overline{[\mathsf{TX}_{\mathsf{SP},0}]}, (1, \{\hat{\sigma}^A_{\mathsf{TX}_{\mathsf{SP},0}}, \hat{\sigma}^B_{\mathsf{TX}_{\mathsf{SP},0}}\}))$, store $\Gamma^A(\gamma.id) := (\gamma, \mathsf{TX}_{\mathsf{FU}}, \mathsf{TX}^A_{\mathsf{CM},0}, [\mathsf{TX}^B_{\mathsf{CM},0}], \overline{\mathsf{TX}_{\mathsf{SP},0}})$ and $(\mathtt{CREATED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$. Else $\Gamma^A(\gamma.id) := (\bot, \mathsf{TX}_{\mathsf{FU}}, \bot, \bot, \bot)$ and stop.

Update

Case A is honest and B is corrupted.

Upon A sending $(\mathtt{UPDATE}, id, \vec{\theta}, t_{stp}) \overset{\tau_0}{\hookrightarrow} \mathscr{F}$, proceed as follows:

1. Send $(\text{updateReq}, id, \vec{\theta}, t_{stp}) \overset{\tau_0}{\hookrightarrow} B$.

2. If B sends $(\text{updateInfo}, id, \hat{\sigma}^B_{\mathsf{SP},i+1}) \xrightarrow{\tau_1 \leq \tau_0+1+t_{stp}} A$, extract $\mathsf{TX}_{\mathsf{FU}}$ and $i := \gamma.sn$ from $\Gamma^A(id)$ and create $([\mathsf{TX}^A_{\mathsf{CM},i+1}], [\mathsf{TX}^B_{\mathsf{CM},i+1}]) := \mathtt{GenCommit}([\mathsf{TX}_{\mathsf{FU}}].\text{txid}\|1, I_A, I_B, i+1)$, and $\overline{[\mathsf{TX}_{\mathsf{SP},i+1}]} := \mathtt{GenSplit}(\gamma.\text{st}, i+1)$ for $I_A := (pk^A_{\mathsf{SP}}, pk^A_{\mathsf{RV}}, pk'^A_{\mathsf{RV}})$ and $I_B := (pk^B_{\mathsf{SP}}, pk^B_{\mathsf{RV}}, pk'^B_{\mathsf{RV}})$. If $\mathrm{Vrfy}_{pk^B_{\mathsf{SP}}}(\hat{f}(\overline{[\mathsf{TX}_{\mathsf{SP},i+1}]}); \hat{\sigma}^B_{\mathsf{TX}_{\mathsf{SP},i+1}}) = 1$, then compute $\hat{\sigma}^A_{\mathsf{TX}_{\mathsf{SP},i+1}} := \mathrm{Sign}_{sk^A_{\mathsf{SP}}}(\hat{f}(\overline{[\mathsf{TX}_{\mathsf{SP},i+1}]}))$, set $\overline{\mathsf{TX}_{\mathsf{SP},i+1}} := (\overline{[\mathsf{TX}_{\mathsf{SP},i+1}]}, (1, \{\hat{\sigma}^A_{\mathsf{TX}_{\mathsf{SP},i+1}}, \hat{\sigma}^B_{\mathsf{TX}_{\mathsf{SP},i+1}}\}))$, store $\Gamma'^A(id) := (\bot, [\mathsf{TX}^B_{\mathsf{CM},i+1}], \overline{\mathsf{TX}_{\mathsf{SP},i+1}})$, set $\gamma.\text{flag} := 2$ and $\gamma.\text{st}' := \vec{\theta}$ and send $(\mathtt{UPDATE-OK}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{F}$ on behalf of B. Else stop.

3. If A sends $(\mathtt{SETUP-OK}, id) \xrightarrow{\tau_1+1} \mathscr{F}$, compute $\sigma^A_{\mathsf{TX}^B_{\mathsf{CM},i+1}} := \mathrm{Sign}_{sk^A_{\mathsf{SP}}}(f([\mathsf{TX}^B_{\mathsf{CM},i+1}]))$ and send $(\text{updateComA}, id, \hat{\sigma}^A_{\mathsf{TX}_{\mathsf{SP},i+1}}, \sigma^A_{\mathsf{TX}^B_{\mathsf{CM},i+1}}) \xrightarrow{\tau_1+1} B$. Else, execute the procedure $\mathtt{ForceClose}^A(id)$ and stop.

4. If B sends $(\text{updateComB}, id, \sigma^B_{\mathsf{TX}^A_{\mathsf{CM},i+1}}) \xrightarrow{\tau_1+2} A$, s.t. $\mathrm{Vrfy}_{pk^B_{\mathsf{CM}}}(f([\mathsf{TX}^A_{\mathsf{CM},i+1}]); \sigma^B_{\mathsf{TX}^A_{\mathsf{CM},i+1}}) = 1$, compute $\sigma^A_{\mathsf{TX}^A_{\mathsf{CM},i+1}} = \mathrm{Sign}_{sk^A_{\mathsf{SP}}}(f([\mathsf{TX}^A_{\mathsf{CM},i+1}]))$, set $\mathsf{TX}^A_{\mathsf{CM},i+1} := (\mathscr{H}([\mathsf{TX}^A_{\mathsf{CM},i+1}]), [\mathsf{TX}^A_{\mathsf{CM},i+1}], (1, \{\sigma^A_{\mathsf{TX}^A_{\mathsf{CM},i+1}}, \sigma^B_{\mathsf{TX}^A_{\mathsf{CM},i+1}}\}))$, store $\Gamma'^A(id)[1] := \mathsf{TX}^A_{\mathsf{CM},i+1}$, and send $(\mathtt{SETUP'-OK}, id) \xrightarrow{\tau_1+2} \mathscr{F}$ on behalf of B. Else, execute the procedure $\mathtt{ForceClose}^A(id)$ and stop.

5. If A sends $(\mathtt{REVOKE}, id) \xrightarrow{\tau_1+3} \mathscr{F}$, then create $(\overline{[\mathsf{TX}^A_{\mathsf{RV},i}]}, \overline{[\mathsf{TX}^B_{\mathsf{RV},i}]}) := \mathtt{GenRevoke}(pk_A, pk_B, \gamma.\text{cash}, i+1)$, compute $\hat{\sigma}^A_{\mathsf{TX}^B_{\mathsf{RV},i}} = \mathrm{Sign}_{sk^A_{\mathsf{RV}}}(\hat{f}(\overline{[\mathsf{TX}^B_{\mathsf{RV},i}]}))$ and send

(revokeA, id, $\hat{\sigma}^A_{\mathsf{TX}^B_{\mathsf{RV},i}}$) $\overset{\tau_1+3}{\hookrightarrow}$ B. Else, execute the procedure $\mathtt{ForceClose}^A(id)$ and stop.

6. If B sends (revokeB, id, $\hat{\sigma}^B_{\mathsf{TX}^A_{\mathsf{RV},i}}$) $\overset{\tau_1+4}{\hookrightarrow}$ A, s.t. $\mathsf{Vrfy}_{pk^B_{\mathsf{RV}}}(\hat{f}([\mathsf{TX}^A_{\mathsf{RV},i+1}]); \hat{\sigma}^B_{\mathsf{TX}^A_{\mathsf{RV},i}}) = 1$, then set $\Theta^A(id) := (\hat{\sigma}^B_{\mathsf{TX}^A_{\mathsf{RV},i}})$, $\gamma.\mathsf{sn} := i+1$, $\gamma.\mathsf{st} := \vec{\theta}$, $\Gamma^A(id) := (\gamma, \mathsf{TX}_{\mathsf{FU}}, \mathsf{TX}^A_{\mathsf{CM},i+1}, [\mathsf{TX}^B_{\mathsf{CM},i+1}], \overline{\mathsf{TX}_{\mathsf{SP},i+1}})$, $\gamma.\mathsf{flag} = 1$, and $\Gamma'^A(id) = (\bot, \bot, \bot)$ and send ($\mathtt{REVOKE'}$, id) $\overset{\tau_1+4}{\hookrightarrow}$ $\mathcal{F}$ on behalf of B. Else, execute $\mathtt{ForceClose}^A(id)$ and stop.

Case B is honest and A is corrupted.

Upon A sending $(updateReq, id, \vec{\theta}, t_{stp}) \overset{\tau_0}{\hookrightarrow} B$, proceed as follows:

1. Send ($\mathtt{UPDATE}, id, \vec{\theta}, t_{stp}$) $\overset{\tau_0}{\hookrightarrow}$ $\mathcal{F}$ on behalf of A.

2. If B sends ($\mathtt{UPDATE-OK}, id$) $\overset{\tau_1 \leq \tau_0+1+t_{stp}}{\hookrightarrow}$ $\mathcal{F}$, extract $\mathsf{TX}_{\mathsf{FU}}$ and $i := \gamma.sn$ from $\Gamma^B(id)$, create $([\mathsf{TX}^A_{\mathsf{CM},i+1}], [\mathsf{TX}^B_{\mathsf{CM},i+1}]) := \mathtt{GenCommit}([\mathsf{TX}_{\mathsf{FU}}].\mathsf{txid} \| 1, I_A, I_B, i+1)$, and $\overline{[\mathsf{TX}_{\mathsf{SP},i+1}]} := \mathtt{GenSplit}(\gamma.\mathsf{st}, i+1)$ for $I_A := (pk^A_{\mathsf{SP}}, pk^A_{\mathsf{RV}}, pk'^A_{\mathsf{RV}})$ and $I_B := (pk^B_{\mathsf{SP}}, pk^B_{\mathsf{RV}}, pk'^B_{\mathsf{RV}})$, compute $\hat{\sigma}^B_{\mathsf{TX}_{\mathsf{SP},i+1}} = \mathsf{Sign}_{sk^B_{\mathsf{SP}}}(\hat{f}(\overline{[\mathsf{TX}_{\mathsf{SP},i+1}]}))$ and send (updateInfo, id, $\hat{\sigma}^B_{\mathsf{TX}_{\mathsf{SP},i+1}}$) $\overset{\tau_1}{\hookrightarrow}$ A.

3. If A sends (updateComA, id, $\hat{\sigma}^A_{\mathsf{TX}_{\mathsf{SP},i+1}}$, $\sigma^A_{\mathsf{TX}^B_{\mathsf{CM},i+1}}$) $\overset{\tau_1+1}{\hookrightarrow}$ B, such that $\mathsf{Vrfy}_{pk^A_{\mathsf{SP}}}(\hat{f}(\overline{[\mathsf{TX}_{\mathsf{SP},i+1}]}); \hat{\sigma}^A_{\mathsf{TX}_{\mathsf{SP},i+1}}) = 1$ and $\mathsf{Vrfy}_{pk^A_{\mathsf{CM}}}(f([\mathsf{TX}^B_{\mathsf{CM},i+1}]); \sigma^A_{\mathsf{TX}^B_{\mathsf{CM},i+1}}) = 1$, compute $\sigma^B_{\mathsf{TX}^B_{\mathsf{CM},i+1}} = \mathsf{Sign}_{sk^B_{\mathsf{CM}}}(\hat{f}([\mathsf{TX}^B_{\mathsf{CM},i+1}]))$, set $\overline{\mathsf{TX}_{\mathsf{SP},i+1}} = (\overline{[\mathsf{TX}_{\mathsf{SP},i+1}]}, (1, \{\hat{\sigma}^A_{\mathsf{TX}_{\mathsf{SP},i+1}}, \hat{\sigma}^B_{\mathsf{TX}_{\mathsf{SP},i+1}}\}))$, $\mathsf{TX}^B_{\mathsf{CM},i+1} = ([\mathsf{TX}^B_{\mathsf{CM},i+1}], (1, \{\sigma^A_{\mathsf{TX}^B_{\mathsf{CM},i+1}}, \sigma^B_{\mathsf{TX}^B_{\mathsf{CM},i+1}}\}))$, $\gamma.\mathsf{flag} = 2$ and $\gamma.\mathsf{st}' = \vec{\theta}$, store $\Gamma'^B(id) = (\mathsf{TX}^B_{\mathsf{CM},i+1}, [\mathsf{TX}^A_{\mathsf{CM},i+1}], \overline{\mathsf{TX}_{\mathsf{SP},i+1}})$, set $\gamma.\mathsf{flag} = 2$ and $\gamma.\mathsf{st}' = \vec{\theta}$ and send ($\mathtt{SETUP-OK}, id$) $\overset{\tau_1+1}{\hookrightarrow}$ $\mathcal{F}$ on behalf of A. Else execute the procedure $\mathtt{ForceClose}^B(id)$ and stop.

4. If ($\mathtt{SETUP'-OK}, id$) $\overset{\tau_1+2}{\hookleftarrow}$ B, compute $\sigma^B_{\mathsf{TX}^A_{\mathsf{CM},i+1}} = \mathsf{Sign}_{sk^B_{\mathsf{SP}}}(\hat{f}([\mathsf{TX}^A_{\mathsf{CM},i+1}]))$ and send (updateComB, id, $\sigma^B_{\mathsf{TX}^A_{\mathsf{CM},i+1}}$) $\overset{\tau_1+2}{\hookrightarrow}$ A. Else, execute $\mathtt{ForceClose}^B(id)$ and stop.

5. Create $(\overline{[\mathsf{TX}^A_{\mathsf{RV},i}]}, \overline{[\mathsf{TX}^B_{\mathsf{RV},i}]}) := \mathtt{GenRevoke}(pk_A, pk_B, \gamma.\mathsf{cash}, i+1)$. If A sends (revokeA, id, $\hat{\sigma}^A_{\mathsf{TX}^B_{\mathsf{RV},i}}$) $\overset{\tau_1+3}{\hookrightarrow}$ B, s.t. $\mathsf{Vrfy}_{pk^A_{\mathsf{RV}}}(\hat{f}([\mathsf{TX}^B_{\mathsf{RV},i+1}]); \hat{\sigma}^A_{\mathsf{TX}^B_{\mathsf{RV},i+1}}) = 1$, set $\Theta^B(id) := (\hat{\sigma}^A_{\mathsf{TX}^B_{\mathsf{RV},i}})$, $\gamma.\mathsf{sn} := i+1$, $\gamma.\mathsf{st} := \vec{\theta}$, $\Gamma^B(id) := (\gamma, \mathsf{TX}_{\mathsf{FU}}, \mathsf{TX}^B_{\mathsf{CM},i+1},$

$[\mathsf{TX}^{A}_{\mathsf{CM},i+1}], \mathsf{TX}_{\mathsf{SP},i+1})$, $\gamma.\mathsf{flag} = 1$, and $\Gamma'^{B}(id) = (\bot, \bot, \bot)$ and send $(\mathtt{REVOKE}, id) \overset{\tau_1+3}{\hookrightarrow} \mathcal{F}$ on behalf of A. Else, execute the procedure $\mathtt{ForceClose}^{B}(id)$ and stop.

6. If B sends $(\mathtt{REVOKE}', id) \overset{\tau_1+4}{\hookrightarrow} \mathcal{F}$, compute $\hat{\sigma}^{B}_{\mathsf{TX}^{A}_{\mathsf{RV},i}} = \mathsf{Sign}_{sk^{B}_{\mathsf{RV}}}(\hat{f}([\mathsf{TX}^{A}_{\mathsf{RV},i}], 1))$, send $(\text{revokeB}, id, \hat{\sigma}^{B}_{\mathsf{TX}^{A}_{\mathsf{RV},i}}) \overset{\tau_1+4}{\hookrightarrow} A$. Else, execute $\mathtt{ForceClose}^{B}(id)$ and stop.

Close

Case A is honest and B is corrupted.

1. Upon A sending $(\mathtt{CLOSE}, id) \overset{\tau_0}{\hookrightarrow} \mathcal{F}$, extract $\mathsf{TX}_{\mathsf{FU}}$, $i := \gamma.sn$, $\mathsf{TX}_{\mathsf{SP},i}$ from $\Gamma^{A}(id)$ and create $[\mathsf{TX}_{\overline{\mathsf{SP}}}] := \mathtt{GenFinSplit}(\mathsf{TX}_{\mathsf{FU}}.\mathsf{txid} \| 1, \gamma.\mathsf{st}))$.
2. Compute $\sigma^{A}_{\mathsf{TX}_{\overline{\mathsf{SP}}}} := \mathsf{Sign}_{sk^{A}_{\mathsf{CM}}}(f(\overline{[\mathsf{TX}_{\overline{\mathsf{SP}}}]}))$ and send $(\text{CloseA}, id, \sigma^{A}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}) \overset{\tau_0}{\hookrightarrow} B$.
3. If B sends $(\text{CloseB}, id, \sigma^{B}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}) \overset{\tau_0}{\hookrightarrow} A$ s.t. $\mathsf{Vrfy}_{pk^{B}_{\mathsf{CM}}}(f([\mathsf{TX}_{\overline{\mathsf{SP}}}]); \sigma^{B}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}) = 1$, then send $(\mathtt{CLOSE}, id) \overset{\tau_0}{\hookrightarrow} \mathcal{F}$ on behalf of B. Otherwise, execute the simulator code of the procedure $\mathtt{ForceClose}^{A}(id)$ and stop.
4. Create $\mathsf{TX}_{\overline{\mathsf{SP}}} := ([\mathsf{TX}_{\overline{\mathsf{SP}}}], (1, \{\sigma^{A}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}, \sigma^{B}_{\mathsf{TX}_{\overline{\mathsf{SP}}}}\}))$ and send $(\text{post}, \mathsf{TX}_{\overline{\mathsf{SP}}}) \overset{\tau_0+1}{\hookrightarrow} \mathcal{L}$.
5. If $\tau_1 \leq \tau_0 + 1 + \Delta$ is the round in which $\mathsf{TX}_{\overline{\mathsf{SP}}}$ is accepted by $\mathcal{L}$, set $\Gamma^{A}(id) = \bot$, $\Theta^{A}(id) = \bot$.

Punish

Case A is honest and B is corrupted.

For each $id \in \{0,1\}^*$, extract $i := \gamma.\text{sn}$ and $flag := \gamma.\text{flag}$ from $\Gamma^{A}(id)$.

If $flag = 1$:

- Parse $(\gamma, \mathsf{TX}_{\mathsf{FU}}, \mathsf{TX}^{A}_{\mathsf{CM},i}, [\mathsf{TX}^{B}_{\mathsf{CM},i}], \overline{\mathsf{TX}_{\mathsf{SP},i}}) := \Gamma^{A}(id)$ and set $I := \{[\mathsf{TX}^{A}_{\mathsf{CM},i}], [\mathsf{TX}^{B}_{\mathsf{CM},i}]\}$. Check if $\mathsf{TX}_{\mathsf{FU}}$.Output is spent by a transaction TX s.t. $[\mathsf{TX}] \notin I$ and TX.Output $\neq \gamma.\text{st}$. If yes:

1. Create $(\overline{[\mathsf{TX}^A_{\mathsf{RV},i-1}]}, \overline{[\mathsf{TX}^B_{\mathsf{RV},i-1}]}) := \mathsf{GenRevoke}(pk^A_{\mathsf{SP}}, pk^B_{\mathsf{SP}}, \gamma.\mathsf{cash}, i)$ and then set $[\mathsf{TX}^A_{\mathsf{RV},i-1}] := (\mathsf{TX}.\mathsf{txid} \| 1, \overline{[\mathsf{TX}^A_{\mathsf{RV},i-1}]})$.
2. compute $\hat{\sigma}^A_{\mathsf{TX}^A_{\mathsf{RV},i-1}} = \mathrm{Sign}_{sk'^A_{\mathsf{RV}}}(\hat{f}(\overline{[\mathsf{TX}^A_{\mathsf{RV},i-1}]}))$ (for the case where B is honest and A is corrupted, sk^B_{RV} is used to compute the signature).
3. Set $\hat{\sigma}^B_{\mathsf{TX}^A_{\mathsf{RV},i-1}} := \Theta^A(id)$, and create $\mathsf{TX}^A_{\mathsf{RV},i-1} := (\mathscr{H}([\mathsf{TX}^A_{\mathsf{RV},i-1}]), [\mathsf{TX}^A_{\mathsf{RV},i-1}], (1, \{\hat{\sigma}^A_{\mathsf{TX}^A_{\mathsf{RV},i-1}}, \hat{\sigma}^B_{\mathsf{TX}^A_{\mathsf{RV},i-1}}\}))$.
4. Post $(\mathrm{post}, \mathsf{TX}^A_{\mathsf{RV},i-1}) \overset{\tau_0}{\hookrightarrow} \mathscr{L}$.
5. Let $\mathsf{TX}^A_{\mathsf{RV},i-1}$ be accepted by $\mathscr{L}$ in round $\tau_1 \leq \tau_0 + \Delta$. Set $\Theta^A(id) :=\perp$, $\Gamma^A(id) :=\perp$ and output $(\mathtt{PUNISHED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$.

Otherwise, if $\mathsf{TX}.\mathsf{Output} = \gamma.\mathsf{st}$, then set $\Theta^A(id) :=\perp$, $\Gamma^A(id) :=\perp$ and output $(\mathtt{CLOSED}, id) \overset{\tau_0}{\hookrightarrow} \mathscr{E}$. Else, set $[\mathsf{TX}_{\mathsf{SP},i}] := (\mathsf{TX}.\mathsf{txid} \| 1, \overline{\mathsf{TX}_{\mathsf{SP},i}}.\mathsf{nLT}, \overline{\mathsf{TX}_{\mathsf{SP},i}}.\mathsf{Output})$, $\mathsf{TX}_{\mathsf{SP},i} := (\mathscr{H}([\mathsf{TX}_{\mathsf{SP},i}]), \mathsf{TX}.\mathsf{txid} \| 1, \overline{\mathsf{TX}_{\mathsf{SP},i}})$ and post $(\mathrm{post}, \mathsf{TX}_{\mathsf{SP},i}) \overset{\tau_0+t}{\hookrightarrow} \mathscr{L}$. Let $\mathsf{TX}_{\mathsf{SP},i}$ be accepted by $\mathscr{L}$ in round $\tau_1 \leq \tau_0 + t + \Delta$. Set $\Theta^A(id) :=\perp$, $\Gamma^A(id) :=\perp$ and output $(\mathtt{CLOSED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$.

If $flag = 2$:

- Parse $(\gamma, \mathsf{TX}_{\mathsf{FU}}, \mathsf{TX}^A_{\mathsf{CM},i}, [\mathsf{TX}^B_{\mathsf{CM},i}], \overline{\mathsf{TX}_{\mathsf{SP},i}}) := \Gamma^A(id)$ as well as $(\mathsf{TX}^A_{\mathsf{CM},i+1}, [\mathsf{TX}^B_{\mathsf{CM},i+1}], \overline{\mathsf{TX}_{\mathsf{SP},i+1}}) := \Gamma'^A(id)$ and set $I = \{[\mathsf{TX}^A_{\mathsf{CM},i}], [\mathsf{TX}^B_{\mathsf{CM},i}], [\mathsf{TX}^A_{\mathsf{CM},i+1}], [\mathsf{TX}^B_{\mathsf{CM},i+1}]\}$. Check if $\mathsf{TX}_{\mathsf{FU}}.\mathsf{Output}$ is spent by a transaction TX s.t. $[\mathsf{TX}] \notin I$, $\mathsf{TX}.\mathsf{Output} \neq \gamma.\mathsf{st}$, and $\mathsf{TX}.\mathsf{Output} \neq \gamma.\mathsf{st}'$. If yes:

  1. Create $(\overline{[\mathsf{TX}^A_{\mathsf{RV},i-1}]}, \overline{[\mathsf{TX}^B_{\mathsf{RV},i-1}]}) := \mathsf{GenRevoke}(pk_A, pk_B, \gamma.\mathsf{cash}, i)$ and then set $[\mathsf{TX}^A_{\mathsf{RV},i-1}] := (\mathsf{TX}.\mathsf{txid} \| 1, \overline{[\mathsf{TX}^A_{\mathsf{RV},i-1}]})$.
  2. compute $\hat{\sigma}^A_{\mathsf{TX}^A_{\mathsf{RV},i-1}} := \mathrm{Sign}_{sk'^A_{\mathsf{RV}}}(\hat{f}(\overline{[\mathsf{TX}^A_{\mathsf{RV},i-1}]}))$ (for the case where B is honest and A is corrupted, sk^B_{RV} is used to compute the signature).
  3. Set $\hat{\sigma}^B_{\mathsf{TX}^A_{\mathsf{RV},i-1}} := \Theta^A(id)$, and create $\mathsf{TX}^A_{\mathsf{RV},i-1} := (\mathscr{H}([\mathsf{TX}^A_{\mathsf{RV},i-1}]), [\mathsf{TX}^A_{\mathsf{RV},i-1}], (2, \{\hat{\sigma}^A_{\mathsf{TX}^A_{\mathsf{RV},i-1}}, \hat{\sigma}^B_{\mathsf{TX}^A_{\mathsf{RV},i-1}}\}))$.
  4. Post $(\mathrm{post}, \mathsf{TX}^A_{\mathsf{RV},i-1}) \overset{\tau_0}{\hookrightarrow} \mathscr{L}$.
  5. Let $\mathsf{TX}^A_{\mathsf{RV},i-1}$ be accepted by $\mathscr{L}$ in round $\tau_1 \leq \tau_0 + \Delta$. Set $\Theta^A(id) :=\perp$, $\Gamma^A(id) :=\perp$ and output $(\mathtt{PUNISHED}, id) \overset{\tau_1}{\hookrightarrow} \mathscr{E}$.

  Otherwise, if $\mathsf{TX}.\mathsf{Output} = \gamma.\mathsf{st}$ or $\mathsf{TX}.\mathsf{Output} = \gamma.\mathsf{st}'$ hold, then set $\Theta^A(id) :=\perp$, $\Gamma^A(id) :=\perp$ and output $(\mathtt{CLOSED}, id) \overset{\tau_0}{\hookrightarrow} \mathscr{E}$. Else, set $[\mathsf{TX}_{\mathsf{SP},i+1}] = (\mathsf{TX}.\mathsf{txid} \| 1,$

$\overline{\mathsf{TX}_{\mathsf{SP},i+1}}.\mathsf{nLT}, \overline{\mathsf{TX}_{\mathsf{SP},i+1}}.\mathsf{Output})$, $\mathsf{TX}_{\mathsf{SP},i+1} := (\mathscr{H}([\mathsf{TX}_{\mathsf{SP},i+1}]), \mathsf{TX}.\mathsf{txid}\|1, \overline{\mathsf{TX}_{\mathsf{SP},i+1}})$ and post $(\text{post}, \mathsf{TX}_{\mathsf{SP},i+1}) \xhookrightarrow{\tau_0+t} \mathscr{L}$. Let $\mathsf{TX}_{\mathsf{SP},i+1}$ be accepted by $\mathscr{L}$ in round $\tau_1 \leq \tau_0 + t + \Delta$. Then, set $\Theta^P(id) := \bot$, $\Gamma^A(id) := \bot$ and $(\mathtt{CLOSED}, id) \xhookrightarrow{\tau_1} \mathscr{E}$.

Subprocedure $\mathtt{ForceClose}^P(id)$

Let τ_0 be the current round. Extract $i := \gamma.\mathsf{sn}$, $flag := \gamma.\mathsf{flag}$, $\mathsf{TX}^P_{\mathsf{CM},i}$ and $\overline{\mathsf{TX}_{\mathsf{SP},i}}$ from $\Gamma^P(id)$ and $\mathsf{TX}^P_{\mathsf{CM},i+1}$ and $\overline{\mathsf{TX}_{\mathsf{SP},i+1}}$ from $\Gamma'^P(id)$.

If $flag = 1$:

1. Post $(\text{post}, \mathsf{TX}^P_{\mathsf{CM},i}) \xhookrightarrow{\tau_0} \mathscr{L}$.
2. Let $\tau_1 \leq \tau_0 + \Delta$ be the round in which $\mathsf{TX}^P_{\mathsf{CM},i}$ is accepted by the blockchain. Wait for t rounds, set $[\mathsf{TX}_{\mathsf{SP},i}] = (\mathsf{TX}^P_{\mathsf{CM},i}.\mathsf{txid}\|1, \overline{\mathsf{TX}_{\mathsf{SP},i}}.\mathsf{nLT}, \overline{\mathsf{TX}_{\mathsf{SP},i}}.\mathsf{Output})$, $\mathsf{TX}_{\mathsf{SP},i} := (\mathscr{H}([\mathsf{TX}_{\mathsf{SP},i}]), \mathsf{TX}^P_{\mathsf{CM},i}.\mathsf{txid}\|1, \overline{\mathsf{TX}_{\mathsf{SP},i}})$ and post $(\text{post}, \mathsf{TX}_{\mathsf{SP},i}) \xhookrightarrow{\tau_2 := \tau_1 + t} \mathscr{L}$.
3. Once $\mathsf{TX}_{\mathsf{SP},i}$ is accepted by $\mathscr{L}$ in round $\tau_3 \leq \tau_2 + \Delta$ set $\Theta^P(id) := \bot$ and $\Gamma^P(id) := \bot$ and output $(\mathtt{CLOSED}, id) \xhookrightarrow{\tau_3} \gamma.\mathsf{users}$.

Otherwise, extract $\mathsf{TX}^P_{\mathsf{CM},i+1}$ and $\overline{\mathsf{TX}_{\mathsf{SP},i+1}}$ from $\Gamma'^P(id)$:

1. If $\mathsf{TX}^P_{\mathsf{CM},i+1} = \bot$, Send $(\text{post}, \mathsf{TX}^P_{\mathsf{CM},i}) \xhookrightarrow{\tau_0} \mathscr{L}$. Else, Send $(\text{post}, \mathsf{TX}^P_{\mathsf{CM},i+1}) \xhookrightarrow{\tau_0} \mathscr{L}$.
2. Let $\tau_1 \leq \tau_0 + \Delta$ be the round in which either $\mathsf{TX}^P_{\mathsf{CM},i}$ or $\mathsf{TX}^P_{\mathsf{CM},i+1}$ is accepted by the blockchain. Wait for t rounds, set $[\mathsf{TX}_{\mathsf{SP},i+1}] = (\mathsf{TX}.\mathsf{txid}\|1, \overline{\mathsf{TX}_{\mathsf{SP},i+1}}.\mathsf{nLT}, \overline{\mathsf{TX}_{\mathsf{SP},i+1}}.\mathsf{Output})$, $\mathsf{TX}_{\mathsf{SP},i+1} := (\mathscr{H}([\mathsf{TX}_{\mathsf{SP},i+1}]), \mathsf{TX}.\mathsf{txid}\|1, \overline{\mathsf{TX}_{\mathsf{SP},i+1}})$ and then post $(\text{post}, \mathsf{TX}_{\mathsf{SP},i+1}) \xhookrightarrow{\tau_2 := \tau_1 + t} \mathscr{L}$.
3. Once $\mathsf{TX}_{\mathsf{SP},i+1}$ is accepted by $\mathscr{L}$ in round $\tau_3 \leq \tau_2 + \Delta$ set $\Theta^P(id) := \bot$ and $\Gamma^P(id) := \bot$ and output $(\mathtt{CLOSED}, id) \xhookrightarrow{\tau_3} \gamma.\mathsf{users}$.

Lemma 7.1. *Let Σ be a secure signature scheme. Then, the Create phase of protocol π GUC-emulates the Create phase of functionality $\mathscr{F}$.*

Proof. We define the following message.

- $m_0 := (\text{createCom}, id, \hat{\sigma}^B_{\mathsf{TX}_{\mathsf{SP},0}}, \sigma^B_{\mathsf{TX}^A_{\mathsf{CM},0}})$,

The proof is composed of multiple hybrids, where we gradually modify the initial experiment.

Hybrid $\mathscr{H}_0^{Cr}$: This corresponds to the Create phase of protocol π.

Hybrid $\mathscr{H}_1^{Cr}$: For the honest party A in hybrid $\mathscr{H}_0^{Cr}$, if the corrupted party B publishes the funding transaction $\mathsf{TX}_{\mathsf{FU}}$ on the ledger $\mathscr{L}$ without sending the message m_0 to A in round $\tau_0 + 1$, then the experiment outputs `Error` and fails.

Simulator $\mathscr{S}^{Cr}$: This corresponds to the Create phase of the simulator, as defined in beginning of this section.

Lemma 7.2 proves the indistinguishability of the neighbouring experiments Hybrid $\mathscr{H}_0^{Cr}$ and Hybrid $\mathscr{H}_1^{Cr}$. Lemma 7.3 also proves the indistinguishability of the neighbouring experiments Hybrid $\mathscr{H}_1^{Cr}$ and Simulator $\mathscr{S}^{Cr}$. This concludes the proof of Lemma 7.1. □

Lemma 7.2. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\text{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_0^{Cr},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx$$
$$\{\text{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_1^{Cr},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$$

Proof. In addition to the message m_0 defined in Lemma 7.1, we define m_1 as follows:

- $m_1 := (\text{createFund}, id, \sigma^A_{\mathsf{TX}_{\mathsf{FU}}})$,

Two hybrids $\mathscr{H}_0^{Cr}$ and $\mathscr{H}_1^{Cr}$ differ if the experiment outputs `Error`. Thus, we must bound the probability that this event occurs. The experiment outputs `Error`, if and only if the corrupted party B publishes the funding transaction $\mathsf{TX}_{\mathsf{FU}}$ on the ledger $\mathscr{L}$ without sending the message m_0 to A. Furthermore, according to the protocol π, if A does not receive the message m_0, he does not send the message m_1 including the signature $\sigma^A_{\mathsf{TX}_{\mathsf{FU}}}$ to B. However, this signature must be part of $\mathsf{TX}_{\mathsf{FU}}$.Witness if $\mathsf{TX}_{\mathsf{FU}}$ is published on $\mathscr{L}$ and hence the signature $\sigma^A_{\mathsf{TX}_{\mathsf{FU}}}$ must be created by the adversary. Thus, given that the experiment outputs `Error` with non-negligible probability, as will be shown in the next paragraph, we construct a reduction against the existential unforgeability of the underlying signature scheme Σ with non-negligible success probability which contradicts with our assumption regarding the security of Σ.

Assume that $\Pr[\texttt{Error} \mid \mathscr{H}_0^{Cr}] \geq \frac{1}{\text{poly}(\lambda)}$. The reduction receives as input a public key pk from the challenger and registers it by sending the message (register, pk) to $\mathscr{L}$. Now assume that the honest party A, upon receiving the message $(\texttt{INTRO}, \gamma, tid_A)$ from $\mathscr{E}$,

initiates the Create phase of protocol π. If in this process, m_0 is received from the adversary, the hybrid $\mathscr{H}_1^{Cr}$ does not output Error and the reduction aborts. If the experiment outputs Error, meaning that for this channel the corresponding funding transaction $\mathsf{TX}_{\mathsf{FU}}$ is accepted by $\mathscr{L}$, then the reduction outputs (m^*, σ^*) with $m^* = [\mathsf{TX}_{\mathsf{FU}}]$ and $\sigma^* \in \mathsf{TX}_{\mathsf{FU}}$.Witness. This reduction is clearly efficient, and whenever $\mathscr{H}_1^{Cr}$ outputs Error, the reduction succeeds in forging the signature. Moreover, the reduction has never called the signing oracle for any messages. Therefore, the reduction outputs a valid forgery with probability at least $\frac{1}{\mathsf{poly}(\lambda)}$, which contradicts with our assumption regarding the security of the signature scheme Σ. This proves that $\Pr[\texttt{Error} \mid \mathscr{H}_0^{Cr}] < \frac{1}{\mathsf{poly}(\lambda)}$.

□

Lemma 7.3. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_1^{Cr},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx$$
$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\varphi_{\mathscr{F}},\mathcal{S}^{Cr},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof. The two experiments are identical, and hence, indistinguishability follows. □

Lemma 7.4. *The Update phase of protocol π GUC-emulates the Update phase of functionality $\mathscr{F}$.*

Proof. The two experiments are identical, and hence, indistinguishability follows.

□

Lemma 7.5. *The Close phase of protocol π GUC-emulates the Close phase of functionality $\mathscr{F}$.*

Proof. The proof is composed of multiple hybrids, where we gradually modify the initial experiment.

Hybrid $\mathscr{H}_0^{Cl}$: This corresponds to the Close phase of protocol π.

Hybrid $\mathscr{H}_1^{Cl}$: If the honest party A initiates hybrid $\mathscr{H}_0^{Cl}$ in round τ_0 and in round $\tau_0+1+\Delta$, the output $\mathsf{TX}_{\mathsf{FU}}$.Output is still unspent, then the experiment outputs Error and fails.

Simulator $\mathcal{S}^{Cl}$: This corresponds to the Close phase of the simulator, as defined at the beginning of this section.

Lemma 7.6 proves the indistinguishability of the neighbouring experiments Hybrid $\mathscr{H}_0^{Cl}$ and Hybrid $\mathscr{H}_1^{Cl}$. Lemma 7.7 also proves the indistinguishability of the neighbouring experiments Hybrid $\mathscr{H}_1^{Cl}$ and Simulator $\mathcal{S}^{Cl}$. This concludes the proof of Lemma 7.5. □

Lemma 7.6. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_0^{Cl},\mathscr{A},\mathscr{E}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx$$
$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_1^{Cl},\mathscr{A},\mathscr{E}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof. Similar to the proof of Lemma 7.2, we must bound the probability that $\mathscr{H}_1^{Cl}$ outputs `Error`. According to $\mathscr{H}_1^{Cl}$, the honest party A sends either $(\mathrm{post}, \mathsf{TX}_{\overline{\mathsf{SP}}}) \xhookrightarrow{\tau_0+1} \mathscr{L}$ or $(\mathrm{post}, \mathsf{TX}^A_{\mathsf{CM},i}) \xhookrightarrow{\tau_0+1} \mathscr{L}$. Since both $\mathsf{TX}_{\overline{\mathsf{SP}}}$ and $\mathsf{TX}^A_{\mathsf{CM},i}$ are valid and both take output of $\mathsf{TX}_{\mathsf{FU}}$ as their input, based on the ledger functionality $\mathscr{L}$, $\mathsf{TX}_{\mathsf{FU}}$.Output becomes spent within at most Δ rounds. Therefore, $\mathscr{H}_1^{Cl}$ will never output `Error`. This completes the proof. □

Lemma 7.7. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_1^{Cl},\mathscr{A},\mathscr{E}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx$$
$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\varphi_{\mathscr{F}},\mathscr{S}^{Cl},\mathscr{E}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof. The two experiments are identical, and hence, indistinguishability follows. □

Lemma 7.8. *Let Σ be a secure signature scheme. Then, the Punish phase of protocol π GUC-emulates the Punish phase of functionality $\mathscr{F}$.*

Proof. The proof is composed of multiple hybrids, where we gradually modify the initial experiment.

- Hybrid $\mathscr{H}_0^{Pu}$: This corresponds to the Punish phase of the protocol π.
- Hybrid $\mathscr{H}_1^{Pu}$: For the honest party A in hybrid $\mathscr{H}_0^{Pu}$, if a transaction TX is published s.t. $[\mathsf{TX}] = [\mathsf{TX}^B_{\mathsf{CM},j}]$ with $j \in [0, i-1]$ but $\mathsf{TX}^A_{\mathsf{RV},i-1}$ is not accepted by $\mathscr{L}$ within Δ rounds, then the experiment outputs `Error` and fails.
- Hybrid $\mathscr{H}_2^{Pu}$: For the honest party A in hybrid $\mathscr{H}_1^{Pu}$, if a transaction TX is published s.t. $[\mathsf{TX}] = [\mathsf{TX}^B_{\mathsf{CM},i}]$ given that $\gamma.\mathsf{flag} = 1$ or $[\mathsf{TX}] \in \{[\mathsf{TX}^B_{\mathsf{CM},i}], [\mathsf{TX}^B_{\mathsf{CM},i+1}]\}$ given that $\gamma.\mathsf{flag} = 2$, and then a transaction TX' is published s.t. $\mathsf{TX}'.\mathsf{Input} = \mathsf{TX}.\mathrm{txid}\|1$ and $\mathsf{TX}'.\mathsf{Witness}.\eta = 2$ (i.e. $\mathsf{TX}'.\mathsf{Witness}$ satisfies the second sub-condition of $\mathsf{TX}.\mathsf{Output}$), then the experiment outputs `Error` and fails.
- Hybrid $\mathscr{H}_3^{Pu}$: For the honest party A in hybrid $\mathscr{H}_2^{Pu}$, if either of the following cases occurs, the experiment outputs `Error` and fails.

  - While γ.flag $= 1$, a transaction TX is published s.t. $[\mathsf{TX}] = [\mathsf{TX}^{A}_{\mathsf{CM},i}]$, and then a transaction TX' is published s.t. TX'.Input $= \mathsf{TX}.\mathsf{txid}\|1$ and TX'.Witness.$\eta = 2$ (i.e. TX'.Witness satisfies the second sub-condition of TX.Output).
  - While γ.flag $= 2$, a transaction TX is published s.t. $[\mathsf{TX}] = [\mathsf{TX}^{A}_{\mathsf{CM},i}]$ given that $\Gamma'^{A}(id)[1] = \perp$ or $[\mathsf{TX}] = [\mathsf{TX}^{A}_{\mathsf{CM},i+1}]$ otherwise, and then a transaction TX' is published s.t. TX'.Input $= \mathsf{TX}.\mathsf{txid}\|1$ and TX'.Witness.$\eta = 2$ (i.e. TX'.Witness satisfies the second sub-condition of TX.Output).

- Hybrid $\mathcal{H}_4^{Pu}$: For the honest party A in hybrid $\mathcal{H}_3^{Pu}$, if either of the following cases occurs, the experiment outputs `Error` and fails.
  - While γ.flag $= 1$, a transaction TX with $[\mathsf{TX}] \in \{[\mathsf{TX}^{A}_{\mathsf{CM},i}], [\mathsf{TX}^{B}_{\mathsf{CM},i}]\}$ is published, but $\mathsf{TX}_{\mathsf{SP},i}$ is not accepted by $\mathscr{L}$ within $t + \Delta$ rounds.
  - While γ.flag $= 2$, a transaction TX with $[\mathsf{TX}] \in \{[\mathsf{TX}^{A}_{\mathsf{CM},i}], [\mathsf{TX}^{B}_{\mathsf{CM},i}], [\mathsf{TX}^{B}_{\mathsf{CM},i+1}]\}$ is published, but $\mathsf{TX}_{\mathsf{SP},i}$ or $\mathsf{TX}_{\mathsf{SP},i+1}$ is not accepted by $\mathscr{L}$ within $t + \Delta$ rounds.
- Hybrid $\mathcal{H}_5^{Pu}$: For the honest party A in hybrid $\mathcal{H}_4^{Pu}$, if either of the following cases occurs, the experiment outputs `Error` and fails.
  - While γ.flag $= 1$, a transaction TX with $\mathsf{TX}.\mathsf{Input} = \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid}\|1$ is published s.t. $[\mathsf{TX}] \notin \{[\mathsf{TX}^{A}_{\mathsf{CM},i}], [\mathsf{TX}^{B}_{\mathsf{CM},j}]\}$, $j = [0, i]$ and $\mathsf{TX}.\mathsf{Output} \neq \gamma.\mathsf{st}$.
  - While γ.flag $= 2$, a transaction TX with $\mathsf{TX}.\mathsf{Input} = \mathsf{TX}_{\mathsf{FU}}.\mathsf{txid}\|1$ is published s.t. $[\mathsf{TX}] \notin \{[\mathsf{TX}^{A}_{\mathsf{CM},i}], [\mathsf{TX}^{A}_{\mathsf{CM},i+1}], [\mathsf{TX}^{B}_{\mathsf{CM},j}]\}$, $j = [0, i+1]$ and $\mathsf{TX}.\mathsf{Output} \neq \gamma.\mathsf{st}$.
- Simulator $\mathcal{S}^{Pu}$: This corresponds to the Punish phase of the simulator, as defined at the beginning of this section.

Lemmas 7.9 to 7.14 prove the indistinguishability of the above neighbouring experiments. This concludes the proof of Lemma 7.8. □

Lemma 7.9. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathcal{H}_0^{Pu},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx \{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathcal{H}_1^{Pu},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof. Two hybrids differ if the experiment outputs `Error`. Thus, we must bound the probability that this event occurs. The hybrid $\mathcal{H}_1^{Pu}$ does not output `Error` unless $\mathsf{TX}_{\mathsf{FU}}$.Output is spent by a transaction TX s.t. $[\mathsf{TX}] = [\mathsf{TX}^{B}_{\mathsf{CM},j}]$ with $j \in [0, i-1]$. According to $\mathcal{H}_1^{Pu}$, once this transaction is observed by A at the end of round τ_0, A posts $(\mathrm{post}, \mathsf{TX}^{A}_{\mathsf{RV},i-1}) \overset{\tau_0}{\hookrightarrow} \mathscr{L}$. Since $\mathsf{TX}^{A}_{\mathsf{RV},i-1}$ with $\mathsf{TX}^{A}_{\mathsf{RV},i-1}.\mathsf{Input} = \mathsf{TX}.\mathsf{txid}\|1$ is a valid transaction, it is accepted by $\mathscr{L}$ within Δ rounds unless another valid transaction

TX' is published within this Δ-round interval with $\mathsf{TX}'.\mathsf{Input} = \mathsf{TX}.\mathsf{txid}\|1$. The output $\mathsf{TX}.\mathsf{Output}$ has two sub-conditions, one of which must be satisfied by TX'. The first sub-condition $\varphi_1 = pk^A_{\mathsf{SP}} \wedge pk^B_{\mathsf{SP}} \wedge CLTV_{S_0+j} \wedge CSV_t$ cannot be satisfied within t rounds and since we have $t > \Delta$, φ_1 cannot be met within Δ rounds. Satisfying the second sub-condition $\varphi_2 = pk'^A_{\mathsf{RV}} \wedge pk'^B_{\mathsf{RV}} \wedge CLTV_{S_0+j}$ requires A's signature and according to the protocol π, A does not grant such an authorisation to anyone. Thus, if the experiment outputs $\mathtt{Error}$ with non-negligible probability, we construct a reduction against the existential unforgeability of the underlying signature scheme Σ with non-negligible success probability which contradicts our assumption regarding the security of Σ.

Assume that $\Pr[\mathtt{Error} \mid \mathscr{H}_0^{Pu}] \geq \frac{1}{\mathsf{poly}(\lambda)}$. The reduction receives a public key pk from the challenger as input, and in the channel creation phase sets $pk'^A_{\mathsf{RV}} := pk$. The channel might be updated any arbitrary number of times and might be closed at any time using any method (peacefully or forcefully) selected by the adversary. Assume that the channel has been updated i times. If the output of the funding transaction is spent by a transaction TX s.t. $[\mathsf{TX}] \neq [\mathsf{TX}^B_{\mathsf{CM},j}]$ with $j = [0, i-1]$, the hybrid $\mathscr{H}_1^{Pu}$ does not output $\mathtt{Error}$ and hence the reduction aborts. If a transaction TX with $[\mathsf{TX}] \in [\mathsf{TX}^B_{\mathsf{CM},j}]$ with $j = [0, i-1]$ is published, the reduction calls the signing oracle for the message $[\mathsf{TX}^A_{\mathsf{RV},i-1}]$, creates $\mathsf{TX}^A_{\mathsf{RV},i-1}$ and posts it to the ledger $\mathscr{L}$. If $\mathsf{TX}^A_{\mathsf{RV},i-1}$ is published on $\mathscr{L}$ within Δ rounds, the hybrid $\mathscr{H}_1^{Pu}$ does not output $\mathtt{Error}$ and hence the reduction aborts. Otherwise, since $\mathsf{TX}^A_{\mathsf{RV},i-1}$ is a valid transaction, based on our assumptions on $\mathscr{L}$, another transaction TX' with $[\mathsf{TX}'] \neq [\mathsf{TX}^A_{\mathsf{RV},i-1}]$ and $\mathsf{TX}'.\mathsf{Input} = \mathsf{TX}.\mathsf{txid}\|1$ appears on the ledger within this Δ round interval. This causes $\mathscr{H}_1^{Pu}$ to output $\mathtt{Error}$. As mentioned earlier, $\mathsf{TX}'.\mathsf{Witness}$ satisfies the condition $pk'^A_{\mathsf{RV}} \wedge pk'^B_{\mathsf{RV}} \wedge CLTV_{S_0+j}$ of the output $\mathsf{TX}.\mathsf{Output}$. Now, the reduction outputs (m^*, σ^*) with $m^* = [\mathsf{TX}']$ and $\sigma^* \in \mathsf{TX}'.\mathsf{Witness}.\zeta$. Moreover, the reduction has never called the signing oracle for m^* before, because the signing oracle was called only once for $[\mathsf{TX}^A_{\mathsf{RV},i-1}] \neq m^*$. Therefore, the reduction outputs a valid forgery with probability at least $\frac{1}{\mathsf{poly}(\lambda)}$, which contradicts our assumption regarding the security of Σ. This contradiction proves that $\Pr[\mathtt{Error} \mid \mathscr{H}_0^{Pu}] < \frac{1}{\mathsf{poly}(\lambda)}$.

□

Lemma 7.10. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_1^{Pu},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx$$
$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_2^{Pu},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof. Similar to the proof of Lemma 7.9, we show that if the experiment outputs $\mathtt{Error}$ with non-negligible probability we construct a reduction against the existential unforgeability of the underlying signature scheme Σ with non-negligible success probability. Assume that $\Pr[\mathtt{Error} \mid \mathscr{H}_1^{Pu}] \geq \frac{1}{\mathsf{poly}(\lambda)}$. The reduction receives as input a public key

pk, and in the channel creation phase sets $pk'^{A}_{\mathsf{RV}} := pk$. The channel is updated any arbitrary number of times and might be closed at any time using any method (peacefully or forcefully) selected by the adversary. Assume that the channel has been updated i times. If the output of the funding transaction is spent by a transaction TX with $[\mathsf{TX}] \neq [\mathsf{TX}^{B}_{\mathsf{CM},i}]$ given that $\gamma.\mathsf{flag} = 1$ or $[\mathsf{TX}] \notin \{[\mathsf{TX}^{B}_{\mathsf{CM},i}], [\mathsf{TX}^{B}_{\mathsf{CM},i+1}]\}$ given that $\gamma.\mathsf{flag} = 2$, the experiment does not output $\mathtt{Error}$ and the reduction aborts. Otherwise, the reduction waits for t rounds and then publishes the latest split transaction. If $\mathsf{TX}.\mathsf{Output}$ is spent by a transaction TX' s.t. $\mathsf{TX}'.\mathsf{Witness}.\eta = 1$, the experiment does not output $\mathtt{Error}$ and the reduction aborts. However, if $\mathsf{TX}'.\mathsf{Witness}.\eta = 1$, the experiment outputs $\mathtt{Error}$. Since $\mathsf{TX}'.\mathsf{Witness}$ satisfies the condition $pk'^{A}_{\mathsf{RV}} \wedge pk'^{B}_{\mathsf{RV}} \wedge CLTV_{S_0+i}$ of the output $\mathsf{TX}.\mathsf{Output}$, reduction outputs (m^*, σ^*) with $m^* = [\mathsf{TX}']$ and $\sigma^* \in \mathsf{TX}'.\mathsf{Witness}.\zeta$. Moreover, the reduction has never called the signing oracle. Therefore, the reduction outputs a valid forgery with probability at least $\frac{1}{\cdot \mathsf{poly}(\lambda)}$, which contradicts our assumption regarding the security of Σ. This contradiction proves that $\Pr[\mathtt{Error} \mid \mathscr{H}_1^{Pu}] < \frac{1}{\mathsf{poly}(\lambda)}$. □

Lemma 7.11. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_2^{Pu},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx$$
$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_3^{Pu},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof. We bound the probability that the experiment outputs $\mathtt{Error}$. Assume that $\Pr[\mathtt{Error} \mid \mathscr{H}_2^{Pu}] \geq \frac{1}{\mathsf{poly}(\lambda)}$. The reduction receives as input a public key pk from the challenger and in the channel creation phase sets $pk'^{A}_{\mathsf{RV}} := pk$. The channel is updated any arbitrary number of times and might be closed at any time using any method (peacefully or forcefully) selected by the adversary. For the j^{th} channel update, to generate the signature on $\mathsf{TX}^{B}_{\mathsf{RV},j-1}$, a call to the signing oracle for the message $\overline{[\mathsf{TX}^{B}_{\mathsf{RV},j-1}]}$ is performed. We know that the channel has been updated i times. Assume that $\mathsf{TX}_{\mathsf{FU}}.\mathsf{Output}$ is spent by a transaction TX. If one of the following cases occurs, the experiment does not output $\mathtt{Error}$ and the reduction aborts: (1) $\gamma.\mathsf{flag} = 1$ and $[\mathsf{TX}] \neq [\mathsf{TX}^{A}_{\mathsf{CM},i}]$, (2) $\gamma.\mathsf{flag} = 2$, and $[\mathsf{TX}] \neq [\mathsf{TX}^{A}_{\mathsf{CM},i}]$ given that $\Gamma'^{A}(id)[1] = \perp$ or $[\mathsf{TX}] \neq [\mathsf{TX}^{A}_{\mathsf{CM},i+1}]$ otherwise. Otherwise, the reduction waits for t rounds and then publishes the latest split transaction. If $\mathsf{TX}.\mathsf{Output}$ is spent by a transaction TX' s.t. $\mathsf{TX}'.\mathsf{Witness}.\eta = 1$, the experiment does not output $\mathtt{Error}$ and hence the reduction aborts. If $\mathsf{TX}'.\mathsf{Witness}.\eta = 2$, the experiment outputs $\mathtt{Error}$. Now either of the following cases might have occurred:

- We have $\gamma.\mathsf{flag} = 1$ or $\gamma.\mathsf{flag} = 2$ and $\Gamma'^{A}(id)[1] = \perp$ and hence $[\mathsf{TX}] = [\mathsf{TX}^{A}_{\mathsf{CM},i}]$ holds. Also, since $\mathsf{TX}'.\mathsf{Witness}.\eta = 2$, $\mathsf{TX}'.\mathsf{Witness}.\zeta$ satisfies the condition $pk'^{A}_{\mathsf{RV}} \wedge pk'^{B}_{\mathsf{RV}} \wedge CLTV_{S_0+i}$. The reduction outputs (m^*, σ^*) with $m^* = [\mathsf{TX}']$ and $\sigma^* \in \mathsf{TX}'.\mathsf{Witness}.\zeta$. Moreover, the signing oracle was only called for messages $\overline{[\mathsf{TX}^{B}_{\mathsf{RV},j}]}$ with $j < i$ and

since for these transactions we have $\overline{[\mathsf{TX}^B_{\mathsf{RV},j}].\mathsf{nLT}} = S_0 + j < S_0 + i$, according to $\mathscr{L}$, $\mathsf{TX}^B_{\mathsf{RV},j}$ can not spend $\mathsf{TX}.\mathsf{Output}$ and hence $m^* \neq \overline{[\mathsf{TX}^B_{\mathsf{RV},j}]}$ with $j < i$.

- We have $\gamma.\mathsf{flag} = 2$ and $\Gamma'^A(id)[1] \neq \perp$ and hence $[\mathsf{TX}] = [\mathsf{TX}^A_{\mathsf{CM},i+1}]$ holds. Also, since $\mathsf{TX}'.\mathsf{Witness}.\eta = 2$, $\mathsf{TX}'.\mathsf{Witness}.\zeta$ satisfies the condition $pk'^A_{\mathsf{RV}} \wedge pk'^B_{\mathsf{RV}} \wedge CLTV_{S_0+i+1}$. The reduction outputs (m^*, σ^*) with $m^* = [\mathsf{TX}']$ and $\sigma^* \in \mathsf{TX}'.\mathsf{Witness}.\zeta$. Moreover, the signing oracle was only called for messages $\overline{[\mathsf{TX}^B_{\mathsf{RV},j}]}$ with $j \leq i$ and since for these transactions we have $\overline{[\mathsf{TX}^B_{\mathsf{RV},j}].\mathsf{nLT}} = S_0 + j < S_0 + i + 1$, according to $\mathscr{L}$, $\mathsf{TX}^B_{\mathsf{RV},j}$ cannot spend $\mathsf{TX}.\mathsf{Output}$ and hence $m^* \neq \overline{[\mathsf{TX}^B_{\mathsf{RV},j}]}$ with $j \leq i$.

Therefore, the reduction has never called the signing oracle for the message m^* and hence the reduction outputs a valid forgery with probability at least $\frac{1}{\mathsf{poly}(\lambda)}$, which contradicts with our assumption regarding the security of Σ. This contradiction proves that $\Pr[\mathtt{Error} \mid \mathscr{H}_2^{Pu}] < \frac{1}{\mathsf{poly}(\lambda)}$.

□

Lemma 7.12. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_3^{Pu},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx$$
$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_4^{Pu},\mathscr{A},\mathscr{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof. Similar to previous proofs, assume that $\Pr[\mathtt{Error} \mid \mathscr{H}_3^{Pu}] \geq \frac{1}{\mathsf{poly}(\lambda)}$. The reduction receives as input a public key pk, and in the channel creation phase, sets $pk^A_{\mathsf{SP}} = pk$. The channel is updated any arbitrary number of times and might be closed at any time using any method (peacefully or forcefully) selected by the adversary. Once the channel is created or once it is updated for the j^{th} time, a call to the signing oracle is performed to receive the signature on $\mathsf{TX}_{\mathsf{SP},0}$ and $\mathsf{TX}_{\mathsf{SP},j}$, respectively. Assume that the channel has been updated i times. If the output of the funding transaction is spent by a transaction TX with $[\mathsf{TX}] \in \{[\mathsf{TX}^A_{\mathsf{CM},i}], [\mathsf{TX}^B_{\mathsf{CM},i}]$ given that $\gamma.\mathsf{flag} = 1$ or $[\mathsf{TX}] \in \{[\mathsf{TX}^A_{\mathsf{CM},i}], [\mathsf{TX}^B_{\mathsf{CM},i}], [\mathsf{TX}^A_{\mathsf{CM},i+1}], [\mathsf{TX}^B_{\mathsf{CM},i+1}]$ given that $\gamma.\mathsf{flag} = 2$, the experiment does not output $\mathtt{Error}$ and the reduction aborts. Otherwise, the reduction waits for t rounds and then posts the transaction TX' on the ledger $\mathscr{L}$ with $\mathsf{TX}' = \mathsf{TX}_{\mathsf{SP},i}$ given that $\gamma.\mathsf{flag} = 1$ or $\mathsf{TX}' = \mathsf{TX}_{\mathsf{SP},i+1}$ given that $\gamma.\mathsf{flag} = 2$.

If TX' is accepted by $\mathscr{L}$, the experiment does not output $\mathtt{Error}$ and the reduction aborts. If $\mathsf{TX}.\mathsf{Output}$ is spent by a transaction TX'' with $[\mathsf{TX}''] \neq [\mathsf{TX}']$ we know that either the first or the second sub-condition of $\mathsf{TX}.\mathsf{Output}$ is satisfied by $\mathsf{TX}''.\mathsf{Witness}$. The second sub-condition of $\mathsf{TX}.\mathsf{Output}$ is not satisfied by $\mathsf{TX}''.\mathsf{Witness}$ otherwise $\mathscr{H}_2^{Pu}$ or $\mathscr{H}_3^{Pu}$

would output Error which contradicts with our assumptions. Thus, TX''.Witness satisfies the first sub-condition of TX.Output. Therefore, the reduction outputs (m^*, σ^*) with $m^* = \overline{[\mathsf{TX}'']}$ and $\sigma^* \in \mathsf{TX}''$.Witness.$\zeta$. Moreover, the signing oracle was only called for messages $\overline{[\mathsf{TX}_{\mathsf{SP},j}]}$ with $j = [0,i]$ given that γ.flag $= 1$ or $j = [0,i+1]$ given that γ.flag $= 2$. However, $m^* \notin \{\overline{[\mathsf{TX}_{\mathsf{SP},i}]}, \overline{[\mathsf{TX}_{\mathsf{SP},i+1}]}\}$. Otherwise, $\mathscr{H}_4^{Pu}$ would not output Error. Also, $m^* \notin \{\overline{[\mathsf{TX}_{\mathsf{SP},j}]}$ with $j = [0,i-1]$ because for these transactions we have $\overline{[\mathsf{TX}_{\mathsf{SP},j}]}.\mathsf{nLT} = S_0 + j < S_0 + i$, and hence according to $\mathscr{L}$, $\mathsf{TX}_{\mathsf{SP},j}$ can not spend TX.Output. Therefore, the reduction has never called the signing oracle for the message m^* and hence the reduction outputs a valid forgery with probability at least $\frac{1}{\mathsf{poly}(\lambda)}$, which contradicts with our assumption regarding the security of Σ. This contradiction proves that $\Pr[\mathsf{Error} \mid \mathscr{H}_3^{Pu}] < \frac{1}{\mathsf{poly}(\lambda)}$.

□

Lemma 7.13. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_4^{Pu},\mathscr{A},\mathscr{E}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx \{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}_5^{Pu},\mathscr{A},\mathscr{E}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof. Similar to previous proofs, we construct a reduction against the existential unforgeability of the underlying signature scheme Σ. Assume that $\Pr[\mathsf{Error} \mid \mathscr{H}_4^{Pu}] \geq \frac{1}{\mathsf{poly}(\lambda)}$. The reduction receives as input a public key pk from the challenger and in the channel creation phase sets $pk_A := pk$. The channel is updated any arbitrary number of times and might be closed at any time using any method (peacefully or forcefully) selected by the adversary. All calls to the signing algorithm are redirected to the signing oracle. Assume that the channel has been updated i times. Now assume that the output of the funding transaction is spent by a transaction TX s.t. either of the following two sets of conditions hold:

- γ.flag $= 1$, $[\mathsf{TX}] \notin \{[\mathsf{TX}^A_{\mathsf{CM},i}], [\mathsf{TX}^B_{\mathsf{CM},j}]\}$, $j = [0,i]$ and TX.Output $\neq \gamma$.st.
- γ.flag $= 2$, $[\mathsf{TX}] \notin \{[\mathsf{TX}^A_{\mathsf{CM},i}], [\mathsf{TX}^A_{\mathsf{CM},i+1}], [\mathsf{TX}^B_{\mathsf{CM},j}]\}$, $j = [0,i+1]$ and TX.Output $\neq \gamma$.st.

Then, the hybrid outputs Error. The reduction also outputs (m^*, σ^*) with $m^* = [\mathsf{TX}]$ and $\sigma^* \in \mathsf{TX}.\mathsf{Witness}.\zeta$. The signature σ^* is a valid signature on m^* because TX.Witness satisfies the condition of $\mathsf{TX}_{\mathsf{FU}}$.Output which is $pk_A \wedge pk_B$. Moreover, as we will show in the next paragraph, the signing oracle was never called for the message m^*.

Once the channel is created or each time it is updated, a call to the signing oracle is performed to receive the signature on the funding transaction as well as the new commit transaction held by B, i.e. $[\mathsf{TX}^B_{\mathsf{CM},j}]$ with $j = [0,i]$ if γ.flag $= 1$ or $j = [0,i+1]$ otherwise. Also, if the channel is closed forcefully, A calls the signing oracle to receive the signature

on $[\mathsf{TX}^{A}_{\mathsf{CM},i}]$ if $\gamma.\mathsf{flag} = 1$ or either $[\mathsf{TX}^{A}_{\mathsf{CM},i}]$ or $[\mathsf{TX}^{A}_{\mathsf{CM},i+1}]$ otherwise. If the channel is closed peacefully, A calls the signing oracle to receive the signature on $[\mathsf{TX}]$ with $\mathsf{TX}.\mathsf{Output} = \gamma.\mathsf{st}$. Therefore, the reduction has never called the oracle for the message m^*. □

Lemma 7.14. *For all PPT distinguishers $\mathscr{E}$ it holds that*

$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\mathscr{H}^{Pu}_{5},\mathscr{A},\mathscr{E}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \approx$$
$$\{\mathrm{EXE}^{\mathscr{L}(\Delta,\Sigma),\mathscr{F}_{clock}}_{\varphi_{\mathscr{F}},\mathcal{S}^{Pu},\mathscr{E}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof. The two experiments are identical, and hence, indistinguishability follows. □

7.13 Discussions

Compatibility with P2WSH transactions: Let the output of commit transactions be of type P2WSH, meaning that it can be spent based on the fulfilment of the script whose hash is included in the condition part of the output. Now, assume that while the channel is in state n, party A publishes $\mathsf{TX}^{A}_{\mathsf{CM},i}$ with $i < n$. According to the protocol, party B is supposed to instantly publish the latest revocation transaction $\mathsf{TX}^{B}_{\mathsf{RV},n-1}$. To do so, he must create the original script of the main output of $\mathsf{TX}^{A}_{\mathsf{CM},i}$ and add it to the witness data of $\mathsf{TX}^{B}_{\mathsf{RV},n-1}$. Since the parameter i is a part of the script, B must extract the value of i from the published commit transaction. However, i varies in different commit transactions and its value cannot be directly derived from the hash of the script in the commit transaction output. Therefore, the value of i must be encoded in $\mathsf{TX}^{A}_{\mathsf{CM},i}.\mathsf{nLT}$ or in the parameter *sequence* of $\mathsf{TX}^{A}_{\mathsf{CM},i}.\mathsf{Input}$.

Fee handling: Once a dishonest channel party publishes a revoked commit transaction, her counterparty has T rounds time to publish the revocation transaction on the ledger. However, the time it takes for a transaction to be published depends on two factors: (1) network congestion at the time when the transaction is submitted to the blockchain network, and (2) the transaction fee. Revocation transactions in Daric have a single input and a single output. Since based on BIP 143 [58], the ANYPREVOUT flag may be combined with SINGLE flag, it is possible for a channel party to add a new input and a new output to the latest revocation transaction before submitting it to the blockchain. The difference between the value of the new output and the new input can be collected by miners. A similar approach can also be used for commit transactions.

Compatibility with any digital signature scheme: Generalized and FPPW payment channels leverage adaptor signatures and hence may not work if the current Bitcoin digital signature scheme changes to BLS [20] or a post-quantum digital signature. However, Daric is compatible with any digital signature and can benefit from their properties.

Extending Daric to multi-hop payments: Payment channels typically use HTLC to establish a PCN, where parties who do not have a shared channel can still exchange coins by using other nodes as relays. For Daric, since state duplication is avoided, there is no complications in adding HTLC outputs to split transactions.

Other applications: To have a new application on top of a Daric channel, parties must update the channel state such that the new split transaction has one or multiple outputs for the new application. For example, assume that channel parties want to create multiple channels on top of their existing channel. To do so, they update their channel such that the new split transaction of the channel consists of multiple outputs where each output is a 2-of-2 multisignature address shared between channel parties and acts like the output of a funding transaction for a new Daric channel. The only difference between this new channel and the original one is that since the split transaction for the original channel is floating, its transaction identifier depends on its input and so cannot be determined in advance. Hence, the commit transactions of new established channels must be also floating. The only important criteria for new channels is that each channel must have its own set of public keys. Otherwise, for example, a commit transaction from one channel can spend funding transaction output of another channel.

Channel reset: If the lifetime of a Daric channel is close to its end (which occurs if the channel update rate is more than once per second), channel parties can reset the channel off-chain. To do so, they update the channel such that output of the split transaction in the latest state acts like the output of the funding transaction for a new channel. All the state numbers also reset and the new established channel can be updated at least for about 1 billion times again. Along with required data from the new established channel, each party must also maintain the last commit, split and revocation transactions from the original channel.

7.14 Conclusion and Future Work

In this work, we presented an efficient payment channel with unlimited lifetime for Bitcoin, called Daric, that achieves constant storage. Moreover, the new scheme allows the honest channel party to penalise her dishonest counterparty by taking all the channel funds. Daric also guarantees that channel parties can close the channel within a bounded time. Furthermore, the new scheme is compatible with any digital signature algorithm and simultaneously avoids state duplication. We proved Daric is secure in the Universal Composability model.

An interesting open topic to study is extending Daric to an efficient m-party scheme with $m > 2$. Moreover, one of the main advantages of Daric is that the storage requirements

of the watchtower for each channel could be constant over time. However, there are also other factors for a watchtower (e.g. privacy, fairness, and coverage [54]) that must be carefully taken into account. Designing a watchtower for Daric which can achieve optimality in terms of the above mentioned properties could be another subject for future research.

Chapter 8

Conclusion and future work

Conclusion

In this thesis, we focused on different aspects of the payment channel which is a promising solution to the scalability issue of Bitcoin. Since the deployment of the payment channel does not require any changes in the Bitcoin protocol, this idea is already being used in the Lightning Network. To prevent their counterparties from closing the channel with an old state, each party in the Lightning Network should frequently monitor the blockchain. Alternatively, channel parties might employ a third-party service provider, called the watchtower, to do the monitoring task.

In this thesis, first, we formally defined the watchtower and its desired properties. Moreover, we proved a trade-off between two of those properties, i.e. the fairness towards the channel party and the coverage. This trade-off prevents a watchtower scheme from achieving both optimal fairness and coverage. We also compared all the existing watchtower schemes with respect to all the defined properties. This comparison showed none of the current watchtower schemes for Bitcoin achieve fairness towards the channel party and simultaneously protect the channel privacy against the watchtower. These contributions answer our first research question, RQ1, about the formal definition of different properties of a watchtower and the limitations of the existing watchtower schemes in meeting those properties.

We solved the mentioned fairness-privacy problem by designing a watchtower scheme called FPPW which can achieve both fairness and privacy at the same time. Coverage for this scheme is also the highest possible value for a fully-fair watchtower scheme. Focusing on performance features, we observed that for all current watchtower schemes as well as FPPW, the storage costs of the channel parties or their watchtowers increase linearly with each channel update. Thus, we also focused on improving the efficiency of

storage costs of the channel parties and their watchtower and designed a second watchtower scheme called Garrison. The storage costs of all participants for this new scheme increase logarithmically with the number of channel updates. Both FPPW and Garrison avoid state duplication (i.e. both parties store the same version of transactions) and can be implemented without any update in the Bitcoin blockchain. Designing these two watchtower schemes for existing payment channels answer our second research question, RQ2.

At the next step, rather than designing watchtowers for the existing payment channels, we focused on current Bitcoin payment channels and analysed their limitations in achieving all the properties required for a payment channel. Then, relying on the deployment of an already proposed signature type (called ANYPREVOUT), we designed a new payment channel with an unlimited lifetime called Daric. This new scheme with desired computation and communication complexity is optimal in storage, provides a penalisation mechanism and avoids state duplication without relying on any particular property of the underlying digital signature. We also proved the security of Daric in the UC framework. These contributions answer our third research question, RQ3, about the limitations of the existing payment channels and how to design a provably secure payment channel to mitigate those limitations.

In general, our work benefits payment channel research by providing a rigorous study on different aspects of this important type of solution to cryptocurrency scalability. We defined some properties based on which the future payment channels and watchtower schemes can be evaluated. We also designed two watchtower schemes for current payment channels, each focusing on different aspects and hence applicable in different scenarios. All the desirable features of our proposed payment channel might also urge the Bitcoin community to deploy the ANYPREVOUT signature type and benefit from its applications.

Future Work

Despite their advantages, FPPW and Garrison still suffer from some limitations: (i) the storage costs of channel parties and their watchtowers in FPPW increase linearly with each channel update and (ii) Garrison does now achieve fairness with respect to the channel party. Designing a watchtower scheme that mitigates the mentioned limitations of these two schemes has been left to future works. Furthermore, we have established the existence of a trade-off between fairness and coverage in watchtower schemes. Specifically, in order to ensure fairness towards the channel party, the watchtower is required to lock collateral as compensation for potential losses incurred by the hiring party. However, this collateral locking mechanism can weaken the overall coverage provided by the

watchtower. Exploring alternative methods to guarantee the watchtower's service without relying solely on collateral locking presents an intriguing avenue for future research. By investigating such approaches, we can address this trade-off and further enhance the effectiveness of watchtower schemes.

Our proposed payment channel performs well for payment applications when we have 2 parties involved. Exploring the extension of this idea to encompass more general multi-party applications, such as the channel factory [74, 75], represents an interesting and important open topic. The significance lies in the potential to leverage the concept's advantages, such as scalability and efficiency, in broader multi-party scenarios. For instance, by adapting and applying these principles to the channel factory setting, we can potentially achieve enhanced scalability, improved transaction throughput, and reduced costs. Further research and exploration in this direction are warranted to better understand the implications, challenges, and opportunities that arise when extending the idea to multi-party applications beyond the immediate payment channel context. Moreover, designing a watchtower for Daric with all the required properties is left as a future work.

References

[1] Christian Decker, Rusty Russell, and Olaoluwa Osuntokun. eltoo: A simple layer2 protocol for bitcoin. *URL: https://blockstream.com/eltoo.pdf*, 2018.

[2] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[3] Tobias Bamert, Christian Decker, Lennart Elsen, Roger Wattenhofer, and Samuel Welten. Have a snack, pay with bitcoins. In *IEEE P2P 2013 Proceedings*, pages 1–5. IEEE, 2013.

[4] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[5] Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *Cryptology ePrint Archive*, 2013.

[6] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[7] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International conference on financial cryptography and data security*, pages 106–125. Springer, 2016.

[8] Manny Trillo. Stress test prepares visanet for the most wonderful time of the year. *URL: http://www.visa.com/blogarchives/us/2013/10/10/stress-testprepares-visanet-for-the-most-wonderfultime-of-the-year/index.html*, 2013.

[9] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[10] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, pages 45–59, 2016.

[11] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30, 2016.

[12] Adem Efe Gencer, Robbert van Renesse, and Emin Gün Sirer. Service-oriented sharding with aspen. *arXiv preprint arXiv:1611.06816*, 2016.

[13] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. *URL: http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 72:201–224, 2014.

[14] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 635–664. Springer, 2021.

[15] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. 2016.

[16] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Off the chain transactions. *IACR Cryptology ePrint Archive*, 2019:360, 2019.

[17] Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry. Sprites: Payment channels that go faster than lightning. *CoRR abs/1702.05812*, 306, 2017.

[18] Rami Khalil, Arthur Gervais, and Guillaume Felley. Nocust-a securely scalable commit-chain. *Cryptology ePrint Archive, Report 2018/642*, 2018.

[19] Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. *White paper*, pages 1–47, 2017.

[20] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer, 2001.

[21] Thaddeus Dryja and Scaling Bitcoin Milano. Unlinkable outsourced channel monitoring. *URL: https://diyhpl. us/wiki/transcripts/scalingbitcoin/milan/unlinkable-outsourced-channel-monitoring*, 2016.

[22] Majid Khabbazian, Tejaswi Nadahalli, and Roger Wattenhofer. Outpost: A responsive lightweight watchtower. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 31–40, 2019.

[23] Zeta Avarikioti, Orfeas Stefanos Thyfronitis Litos, and Roger Wattenhofer. Cerberus channels: Incentivizing watchtowers for bitcoin. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*, pages 346–366. Springer, 2020.

[24] Patrick McCorry, Surya Bakshi, Iddo Bentov, Sarah Meiklejohn, and Andrew Miller. Pisa: Arbitration outsourcing for state channels. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 16–30, 2019.

[25] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Sok: Consensus in the age of blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 183–198, 2019.

[26] J Spilman. [bitcoin-development] anti dos for tx replacement. *URL: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/ 2013-April/002433.html*, 2013.

[27] Lukas Aumayr, Matteo Maffei, Oğuzhan Ersoy, Andreas Erwig, Sebastian Faust, Siavash Riahi, Kristina Hostáková, and Pedro Moreno-Sanchez. Bitcoin-compatible virtual channels. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 901–918. IEEE, 2021.

[28] Lukas Aumayr, Sri AravindaKrishnan Thyagarajan, Giulio Malavolta, Pedro Moreno-Sanchez, and Matteo Maffei. Sleepy channels: Bitcoin-compatible bi-directional payment channels without watchtowers. *Cryptology ePrint Archive*, 2021.

[29] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 106–123. IEEE, 2019.

[30] Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 473–489, 2017.

[31] Christian Decker and Anthony Towns. Bip 118 - sighashnoinput. *URL: https://github.com/bitcoin/bips/blob/master/bip-0118.mediawiki*, 2017.

[32] Georgia Avarikioti, Felix Laufenberg, Jakub Sliwinski, Yuyi Wang, and Roger Wattenhofer. Towards secure and efficient payment channels. *arXiv preprint arXiv:1811.12740*, 2018.

[33] Sonbol Rahimpour and Majid Khabbazian. Hashcashed reputation with application in designing watchtowers. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2021.

[34] Adam Back. Hashcash-a denial of service counter-measure. 2002.

[35] Bowen Liu, Pawel Szalachowski, and Siwei Sun. Fail-safe watchtowers and short-lived assertions for payment channels. *arXiv preprint arXiv:2003.06127*, 2020.

[36] Marc Leinweber, Matthias Grundmann, Leonard Schönborn, and Hannes Hartenstein. Tee-based distributed watchtowers for fraud protection in the lightning network. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 177–194. Springer, 2019.

[37] Georgia Avarikioti, Eleftherios Kokoris Kogias, and Roger Wattenhofer. Brick: Asynchronous state channels. *arXiv preprint arXiv:1905.11360*, 2019.

[38] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *Network and Distributed System Security Symposium*, 2017.

[39] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A^2l: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1834–1851. IEEE, 2021.

[40] Noemi Glaeser, Matteo Maffei, Giulio Malavolta, Pedro Moreno-Sanchez, Erkan Tairi, and Sri Aravinda Krishnan Thyagarajan. Foundations of coin mixing services. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1259–1273, 2022.

[41] Bitcoin Wiki. Hashed timelock contracts. *URL: https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts*, 2019.

[42] Raiden network. *URL: http://raiden.network/*, 2017.

[43] Vitalik Buterin. Ethereum white paper: a next generation smart contract & decentralized application platform. *First version*, 53, 2014.

[44] Christoph Egger, Pedro Moreno-Sanchez, and Matteo Maffei. Atomic multi-channel updates with constant collateral in bitcoin-compatible payment-channel networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 801–815, 2019.

[45] Maxim Jourenko, Mario Larangeira, and Keisuke Tanaka. Payment trees: Low collateral payments for payment channel networks. In *International Conference on Financial Cryptography and Data Security*, pages 189–208. Springer, 2021.

[46] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. *Cryptology ePrint Archive*, 2018.

[47] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 455–471, 2017.

[48] Bin Yu, Shabnam Kasra Kermanshahi, Amin Sakzad, and Surya Nepal. Chameleon hash time-lock contract for privacy preserving payment channel networks. In *Provable Security: 13th International Conference, ProvSec 2019, Cairns, QLD, Australia, October 1–4, 2019, Proceedings 13*, pages 303–318. Springer, 2019.

[49] Sri Aravinda Krishnan Thyagarajan and Giulio Malavolta. Lockable signatures for blockchains: Scriptless scripts for all signatures. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 937–954. IEEE, 2021.

[50] Lukas Aumayr, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Blitz: Secure multi-hop payments without two-phase commits. In *30th USENIX Security Symposium*, pages 4043–4060, 2021.

[51] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Bitcoin-compatible virtual channels. *IACR Cryptol. ePrint Arch.*, 2020:554, 2020.

[52] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63, 2001.

[53] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4:161–174, 1991.

[54] Arash Mirzaei, Amin Sakzad, Jiangshan Yu, and Ron Steinfeld. Fppw: A fair and privacy preserving watchtower for bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 151–169. Springer, 2021.

[55] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[56] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[57] libbitcoin. Sighash and tx signings. *URL: https://github.com/libbitcoin/libbitcoin-system/wiki/Sighash-and-TX-Signing*, 2018.

[58] Johnson Lau and Pieter Wuille. Bip 143: Transaction signature verification for version 0 witness program. *URL: https://github.com/bitcoin/bips/blob/master/bip-0143.mediawiki*, 2016.

[59] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies.* " O'Reilly Media, Inc.", 2014.

[60] Arash Mirzaei, Amin Sakzad, Jiangshan Yu, and Ron Steinfeld. Garrison: a novel watchtower scheme for bitcoin. In *Australasian Conference on Information Security and Privacy*, pages 489–508. Springer, 2022.

[61] Lightning Developers. Bolt# 3: Bitcoin transaction and script formats, 2017.

[62] Yehuda Lindell. Fast secure two-party ecdsa signing. In *Annual International Cryptology Conference*, pages 613–644. Springer, 2017.

[63] Arash Mirzaei, Amin Sakzad, Jiangshan Yu, and Ron Steinfeld. Daric: a storage efficient payment channel with punishment mechanism. In *International Conference on Information Security*, pages 229–249. Springer, 2022.

[64] Rene Pickhardt. Does eltoo eliminate the need to watch the blockchain/implement watchtowers. *URL: https://bitcoin.stackexchange.com/questions/84846/does-eltoo-eliminate-the-need-to-watch-the-blockchain-implement-watchtowers*, 2019.

[65] Lightning channels - top capacity. *URL: https://1ml.com/channel?order=capacity*.

[66] Using per-update credential to enable eltoo-penalty. *URL: https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-July/002068.html*, 2019.

[67] eltoo: A simplified update mechanism for lightning and off-chain contracts. *URL: https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-June/001313.html*, 2018.

[68] Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 949–966, 2018.

[69] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, Julia Hesse, and Kristina Hostáková. Multi-party virtual state channels. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 625–656. Springer, 2019.

[70] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In *Theory of Cryptography Conference*, pages 61–85. Springer, 2007.

[71] Peter Todd and David A. Harding. Bip 125: Opt-in full replace-by-fee signaling. *URL: https://github.com/bitcoin/bips/blob/master/bip-0125.mediawiki*, 2015.

[72] Bolt 5: Recommendations for on-chain transaction handling. *URL: https://github.com/lightningnetwork/lightning-rfc/blob/master/05-onchain.md*.

[73] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In *Annual international cryptology conference*, pages 324–356. Springer, 2017.

[74] Conrad Burchert, Christian Decker, and Roger Wattenhofer. Scalable funding of bitcoin micropayment channel networks. *Royal Society open science*, 5(8):180089, 2018.

[75] Alejandro Ranchal Pedrosa, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Scalable lightning factories for bitcoin. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 302–309, 2019.