Garrison: A Novel Watchtower Scheme for Bitcoin

Arash Mirzaei, Amin Sakzad, Jiangshan Yu, and Ron Steinfeld

Faculty of Information Technology, Monash University, Melbourne, Australia
{arash.mirzaei,amin.sakzad,jiangshan.yu,ron.steinfeld}@monash.edu

Abstract. In this paper, we propose Garrison, which is a payment channel with watchtower for Bitcoin. For this scheme, the storage requirements of both channel parties and their watchtower 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, the new proposal avoids punish-per-output pattern, meaning that all outputs of a revoked state can be claimed using a single revocation transaction. Garrison can be implemented without any update in Bitcoin script.

Keywords: Bitcoin · payment channel · watchtower

1 Introduction

Payment channel is a promising technique to mitigate the scalability issue of blockchains. To establish a payment channel, two parties lock their funds in a 2-of-2 multisignature address on the blockchain. Then, parties privately carry out multiple payments by exchanging off-chain transactions. Finally, parties close the channel by publishing the last channel state on-chain.

Since the channel parties are generally untrusted and blockchain miners are unaware of the off-chain transactions, a mechanism must be adopted to prevent parties from publishing an old state. In Lightning Network [18], as the most widely used Bitcoin payment channel network, with 31,483 nodes, 82,776 channels and total capacity of 159 Million US dollars[1], when a channel party publishes an old channel state on the blockchain, a period called *dispute period* starts. In this period, the other party can publish a *revocation* transaction and penalize the cheating party by claiming all the channel funds.

However, the dispute process works based on the assumption that parties are always online to detect malicious behaviours. This requirement can be practically violated due to crash failures or DoS attacks against the channel party [15, 18]. To relax this assumption, [18] suggests that channel parties delegate the monitoring

[1] https://1ml.com/statistics, data fetched on 06/12/2021

task to a third party called the *watchtower*. The watchtower is an always-online service provider that monitors the blockchain and acts on behalf of its customers to secure their funds. In other words, once channel parties update their channel, each party gives the revocation transaction to the watchtower. Then, once an old state appears on the ledger, the watchtower broadcasts its corresponding revocation transaction.

Monitor [18] and DCWC [4] are two watchtower schemes for the Lightning Network where the storage size of the watchtower in both schemes linearly increases with each channel update and hence the watchtower's storage costs would be $\mathcal{O}(N)$ with N being the number of channel updates. Generalized channel [1], Cerberus [5] and FPPW [16] are also other payment channels that work based on the dispute period idea. However, for all these schemes, the storage size of the watchtower linearly increases with the number of channel updates.

Outpost [13] is a novel payment channel with 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 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 [1]. In other words, to add an application (e.g. *Virtual channel* [2]) 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 [1]. 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 paper is designing a Bitcoin payment channel with watchtower scheme which is storage-efficient for channel parties and the watchtower and also avoids state duplication and punish-per-output pattern.

1.1 Our Contributions

The contribution of this paper is to present a new payment channel with 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. Table 1 presents a comparison between Garrison and other Bitcoin payment channels that work based on dispute period. We also prove security of the Garrison channel under security of its underlying cryptographic primitives.

Table 1. Comparison of different dispute period-based payment channels with N channel updates, M 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 [10]	$\mathcal{O}(\log(N))$	$\mathcal{O}(N)$	$\mathcal{O}(M)$	$\mathcal{O}(2^k)$
Generalized [4]	$\mathcal{O}(\log(N))$	$\mathcal{O}(N)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Outpost [13]	$\mathcal{O}(N)$	$\mathcal{O}(\log(N))$	$\mathcal{O}(M)$	$\mathcal{O}(2^k)$
FPPW [16]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Cerberus [5]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(M)$	$\mathcal{O}(2^k)$
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.

1.2 Related Works

The first payment channels were introduced in [21] but they suffered from being unidirectional. DMC [8] and Lightning [18] were the first bidirectional payment channels where the former uses decrementing timelocks to replace the current channel state with a newer one and the latter revokes the current state upon authorizing a new state. Generalized channels [1] use adaptor signatures to avoid state duplication. Then, both parties would store the same copy of the channel transactions.

Lightning and generalized channels require parties to be always online to prevent their counter-parties from finalizing the channel with an old state. Since this requirement could be difficult to achieve, parties might delegate it to watchtowers. Monitor [18] is a privacy preserving watchtower scheme for Lightning Network. DCWC [4] proposes using a network of watchtowers to minimize the chance of malicious channel closure. In the above mentioned watchtower schemes, the watchtower is unaccountable, i.e. watchtowers do not guarantee their clients' funds. Cerberus [5] and FPPW [16] are two payment channel with watchtower schemes that focus on fairness with respect to the watchtowers' clients. Outpost [13] presents a payment channel with watchtower that reduces the storage costs of the watchtower from $\mathcal{O}(N)$ to $\mathcal{O}(\log(N))$ where N denotes the number of channel updates.

The payment channels eltoo [7] and Daric [17] use a new Bitcoin signature type called `ANYPREVOUT` [6] (also known as `NOINPUT`) that reduces the storage costs of channel parties and their watchtowers to $\mathcal{O}(1)$. However, deployment of the `ANYPREVOUT` signature type requires a soft fork in Bitcoin. Sleepy channel [3] is a payment channel without watchtower where parties are allowed to go offline for a long time period.

2 Preliminaries and Notations

In this section, we closely follow [16] and [1] to introduce the underlying cryptographic primitives of Garrison and notations.

2.1 Preliminaries

Digital Signature A digital signature scheme $\mathit{\Pi}$ includes three algorithms as following:

- **Key Generation.** $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$ on input 1^κ (κ is the security parameter), 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 and outputs a bit b.

In this work, we assume that the utilized signature schemes are existentially unforgeable under an chosen-message attack ($\mathsf{EUF-CMA}$). It guarantees that it is of negligible probability that an adversary, who has access to a signing oracle, outputs a valid signature on any new message. In this paper, we call such signature schemes secure. ECDSA [12] is a secure signature scheme that is currently being used in Bitcoin. Schnorr [20] is another important secure signature scheme that has been proposed to be introduced in Bitcoin due to its key aggregation and signature aggregation properties.

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

Adaptor Signature Given a hard relation $\mathcal{R}$ and a signature scheme $\mathit{\Pi}$, an adaptor signature protocol $\mathit{\Xi}$ 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 $Y \in L_\mathcal{R}$, 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 $Y \in L_\mathcal{R}$ 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 $Y \in L_{\mathcal{R}}$, outputs $\perp$ or a witness y such that $(Y, y) \in \mathcal{R}$.

An adaptor signature scheme is "secure" if it is existentially unforgeable under chosen message attack ($\mathsf{aEUF-CMA}$ security), pre-signature adaptable and witness extractable. The $\mathsf{aEUF-CMA}$ security guarantees that it is of negligible probability that any PPT adversary with 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 adaptablity guarantees that every pre-signature (possibly generated maliciously) w.r.t. Y can adapt to a valid signature using the witness y with $(Y, y) \in \mathcal{R}$. Witness extractablity guarantees that it is of negligible probability that any PPT adversary with access to signing and pre-signing oracles outputs a valid signature and a statement Y for any new message m s.t. 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 analyzed in [1].

2.2 Notations

Throughout this work, we define different attribute tuples. Let U be a tuple of multiple attributes and one of its attributes is denoted by attr. To refer to this attribute, we use $U.\mathsf{attr}$.

Our focus in this work is on Bitcoin or any other blockchains with UTXO model. In this model, units of value which we call coins are held in *outputs*. Formally, an output θ is a tuple of two attributes, $\theta = (\mathsf{cash}, \varphi)$, where $\theta.\mathsf{cash}$ denotes the amount 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 because satisfying the condition requires a valid signature corresponding with that public key. Satisfying a condition might require authorizations by multiple parties. Such conditions contain public keys of all the involved parties separated by $\wedge$ operation(s). The relative timelock of T rounds in an output condition is denoted by Δ_T. It means the output cannot be spent within T rounds of the blockchain.

A condition might also have several subconditions, one of which must be satisfied to spend the output. Different subconditions of an output are separated by $\vee$ operation(s). The OP_RETURN output is a special output which 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.

A transaction changes ownership of coins, meaning that it takes a list of existing outputs and transfers their coins to a list of new outputs. To distinct between these two lists, we refer to the list of existing outputs as *inputs*. A transaction $\mathtt{TX}$ is formally defined as the tuple $(\mathsf{txid}, \mathsf{Input}, \mathsf{Output}, \mathsf{Witness})$. The identifier $\texttt{TX}.\mathsf{txid} \in \{0,1\}^*$ is computed as $\texttt{TX}.\mathsf{txid} := H([\texttt{TX}])$, where $[\texttt{TX}]$ is called the *body* of the transaction defined as $[\texttt{TX}] := (\texttt{TX}.\mathsf{Input}, \texttt{TX}.\mathsf{Output})$ and H is a hash function which is modeled as a random oracle. The attribute $\texttt{TX}.\mathsf{Input}$ is a list of identifiers for all inputs of $\texttt{TX}$. The attribute $\texttt{TX}.\mathsf{Output} = (\theta_1, \ldots, \theta_n)$ is a list of new outputs. The attribute $\texttt{TX}.\mathsf{Witness} = (\mathsf{W}_1, \ldots, \mathsf{W}_m)$ is a list of tuples where its i^{th} tuple authorizes spending the output that is taken as the i^{th} input of $\texttt{TX}$. The tuple $\mathsf{W}_i = (\eta, \zeta)$ of the witness $\texttt{TX}.\mathsf{Witness}$ contains two attributes where $\mathsf{W}_i.\zeta$ denotes the data, e.g. the signature(s), that is (are) required to meet the $\mathsf{W}_i.\eta^{\text{th}}$ subcondition of the output that is taken as the i^{th} input of $\texttt{TX}$. The signature (pre-signature) of P for $\texttt{TX}.\mathsf{Witness}.\mathsf{W}_i.\zeta$ is denoted by $\sigma_{\texttt{TX}}^{P,i}$ ($\tilde{\sigma}_{\texttt{TX}}^{P,i}$), where i can be removed for single-input transactions. The i^{th} entry of a list L is denoted by $L[i]$ with $i > 0$. Table 2 summarizes the notations.

Table 2. Notations

Notation	Description
$U.\mathsf{attr}$	Attribute attr of the tuple U
$\theta = (\mathsf{cash}, \varphi)$	Output with value cash and script condition φ
$\theta = (0, data)$	OP_RETURN output with *data* as its data
$\texttt{TX}$	Transaction $\texttt{TX} = (\mathsf{txid}, \mathsf{Input}, \mathsf{Output}, \mathsf{Witness})$
$[\texttt{TX}]$	Tuple $(\texttt{TX}.\mathsf{Input}, \texttt{TX}.\mathsf{Output})$
$\texttt{TX}.\mathsf{txid}$	Identifier of the transaction $\texttt{TX}$
$\texttt{TX}.\mathsf{Input}$	List of identifiers for all inputs of $\texttt{TX}$
$\texttt{TX}.\mathsf{Output}$	List of new outputs $(\theta_1, \ldots, \theta_n)$ for $\texttt{TX}$
$\texttt{TX}.\mathsf{Witness}$	List of witnesses $(W_1, \ldots, W_m)$ for $\texttt{TX}$ where W_i corresponds with i^{th} input of $\texttt{TX}$
$W = (\eta, \zeta)$	Witness that fulfills η^{th} subcondition of an output using data ζ
$\sigma_{\texttt{TX}}^{P,i}$	Signature of party P on $\texttt{TX}$ for $\texttt{TX}.\mathsf{Witness}.\mathsf{W}_i.\zeta$
$\tilde{\sigma}_{\texttt{TX}}^{P,i}$	Pre-signature of P on $\texttt{TX}$ for $\texttt{TX}.\mathsf{Witness}.\mathsf{W}_i.\zeta$
Δ_T	Relative timelock of T rounds
$L[i]$	i^{th} entry of a list L

We additionally use charts to illustrate the connections between different transactions. Doubled edge and single edge rectangles respectively illustrate transactions that are already published on-chain or are ready to be published. Dotted edge rectangles show transactions that still lack the required witness for at least one input and hence are unprepared to be propagated in the blockchain network. Directional arrows from i^{th} output of transaction $\texttt{TX}$ to j^{th} input of transaction $\texttt{TX}'$ shows that the transaction $\texttt{TX}'$ takes i^{th} output of the transaction $\texttt{TX}$ as its j^{th} input. If an output has multiple subconditions, it is shown by a diamond shape with multiple arrows where each arrow corresponds with one subcondition. OP_RETURN outputs are illustrated by blocked lines (instead of directional arrows). As an example, Fig. 1 shows that $\texttt{TX}_\texttt{i}$ and $\texttt{TX}_\texttt{j}$ are published and unpublished, respectively. The transaction $\texttt{TX}_\texttt{k}$ is still unprepared to be published on the ledger. The transaction $\texttt{TX}_\texttt{i}$ has two subconditions, where one of the subconditions is owned by both A and B and is relatively timelocked by T rounds and another subcondition is owned by C. The second output of $\texttt{TX}_\texttt{k}$ is an OP_RETURN output.

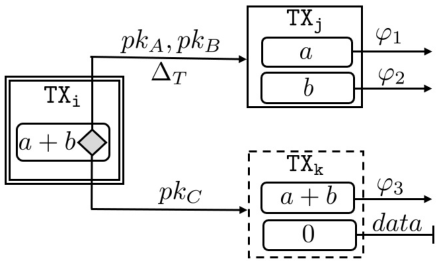


Fig. 1. A sample transaction flow.

3 Garrison Overview

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 [11]. If a valid transaction is propagated in the blockchain network, it is included in the blockchain ledger within τ rounds (i.e. the confirmation delay is τ).

3.2 Garrison Overview

This section overviews the Garrison channel between A (Alice) and B (Bob). We start with a simple payment channel and then modify it step by step to mitigate its limitations.

A Simple Payment Channel Fig. 2 depicts the simple payment channel, introduced in [16]. This channel is created once channel parties publish a funding transaction on the blockchain and hence fund a 2-of-2 multisignature output on the ledger. The i^{th} channel state includes a commit transaction $\texttt{TX}_{\texttt{CM},i}$ as well as a split transaction $\texttt{TX}_{\texttt{SP},i}$. The commit transaction sends the channel funds to a new joint account which is shared between the channel parties. Output of the commit transaction has two subconditions. The first subcondition which is not timelocked, as we will explain later, is used for revocation purposes. The second subcondition is relatively timelocked by T rounds with $T > \tau$ and is met by the corresponding split transaction. Split transaction distributes the channel funds among the channel parties and hence represents the channel state.

The transaction $\mathtt{TX}_{\mathtt{CM},i}$ requires signatures of both parties A and B to be published. To generate $\sigma^B_{\mathtt{TX}_{\mathtt{CM},i}}$, 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 $\tilde{\sigma}^B_{\mathtt{TX}_{\mathtt{CM},i}}$ on $[\mathtt{TX}_{\mathtt{CM},i}]$ and sends the result 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 $\sigma^B_{\mathtt{TX}_{\mathtt{CM},i}}$ and publish $\mathtt{TX}_{\mathtt{CM},i}$ 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.

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 $\mathtt{TX}_{\mathtt{RV},i}$ is created by parties. Unlike the split transaction, the revocation transaction can immediately spend output of the corresponding commit transaction $\mathtt{TX}_{\mathtt{CM},i}$ using its first subcondition which does not contain any timelock. Thus, if the revoked commit transaction $\mathtt{TX}_{\mathtt{CM},i}$ is published by a channel party, let's say A, party B can immediately publish the revocation transaction $\mathtt{TX}_{\mathtt{RV},i}$. Moreover, since commit transactions are signed using the adaptor signature, once $\mathtt{TX}_{\mathtt{CM},i}$ 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 parties have not signed its corresponding revocation transaction yet.

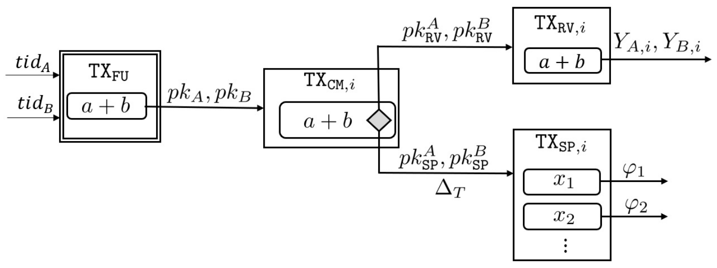

tid_A
tid_B
$\mathtt{TX}_{\mathtt{FU}}$
$a+b$
pk_A, pk_B
$\mathtt{TX}_{\mathtt{CM},i}$
$a+b$
$pk^A_{\mathtt{RV}}, pk^B_{\mathtt{RV}}$
$\mathtt{TX}_{\mathtt{RV},i}$
$a+b$
$Y_{A,i}, Y_{B,i}$
$pk^A_{\mathtt{SP}}, pk^B_{\mathtt{SP}}$
Δ_T
$\mathtt{TX}_{\mathtt{SP},i}$
x_1
x_2
φ_1
φ_2

Fig. 2. A Simple Payment Channel

Reducing the Storage Requirements of the Watchtower All revocation transactions in the introduced scheme 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 [13], our main idea is storing the revocation transaction $\mathtt{TX}_{\mathtt{RV},i}$ inside the commit transaction $\mathtt{TX}_{\mathtt{CM},i}$. Then, once $\mathtt{TX}_{\mathtt{CM},i}$ is published, the watchtower extracts $\mathtt{TX}_{\mathtt{RV},i}$ and records it on the blockchain. However, we have $\mathtt{TX}_{\mathtt{RV},i}.\mathsf{Input} = \mathtt{TX}_{\mathtt{CM},i}.\mathsf{txid}\|1$. Thus, if $\mathtt{TX}_{\mathtt{RV},i}$ is created, signed and finally stored inside $\mathtt{TX}_{\mathtt{CM},i}$, then $[\mathtt{TX}_{\mathtt{CM},i}]$ and hence $\mathtt{TX}_{\mathtt{CM},i}.\mathsf{txid}$ and $\mathtt{TX}_{\mathtt{RV},i}$ change. Thus, there is a self-loop situation [13]. 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 $\mathtt{TX}_{\mathtt{AU},i}$ spends the auxiliary output of the commit transaction. The signatures of party A and party B on $[\mathtt{TX}_{\mathtt{RV},i}]$ are stored in an OP_RETURN output of the auxiliary transaction $\mathtt{TX}_{\mathtt{AU},i}$. The split transaction $\mathtt{TX}_{\mathtt{SP},i}$ spends the main output of $\mathtt{TX}_{\mathtt{CM},i}$ as well as the main output of the auxiliary transaction $\mathtt{TX}_{\mathtt{AU},i}$. Based on this design, parties can be sure that once the revoked commit transaction $\mathtt{TX}_{\mathtt{CM},i}$ is published on the blockchain, its split transaction $\mathtt{TX}_{\mathtt{SP},i}$ cannot be published unless $\mathtt{TX}_{\mathtt{AU},i}$ is also on the blockchain. Furthermore, due to the timelock in the main output of $\mathtt{TX}_{\mathtt{AU},i}$, once this transaction is published on-chain, $\mathtt{TX}_{\mathtt{SP},i}$ cannot be published within $T-1$ rounds. However, the honest party or the watchtower can extract the signatures on $[\mathtt{TX}_{\mathtt{RV},i}]$ from $\mathtt{TX}_{\mathtt{AU},i}$ and publish $\mathtt{TX}_{\mathtt{RV},i}$ immediately. Fig. 3 depicts the transactions flows.

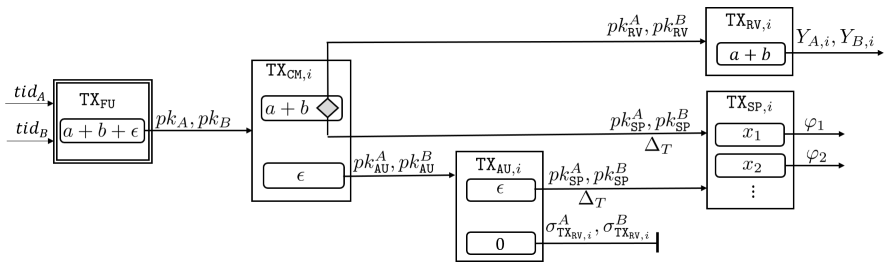

tid_A
tid_B
$\mathtt{TX}_{\mathtt{FU}}$
$a+b+\epsilon$
pk_A, pk_B
$\mathtt{TX}_{\mathtt{CM},i}$
$a+b$
ϵ
$pk^A_{\mathtt{RV}}, pk^B_{\mathtt{RV}}$
$\mathtt{TX}_{\mathtt{RV},i}$
$a+b$
$Y_{A,i}, Y_{B,i}$
$pk^A_{\mathtt{SP}}, pk^B_{\mathtt{SP}}$
Δ_T
$\mathtt{TX}_{\mathtt{SP},i}$
x_1
x_2
$\vdots$
φ_1
φ_2
$pk^A_{\mathtt{AU}}, pk^B_{\mathtt{AU}}$
$\mathtt{TX}_{\mathtt{AU},i}$
ϵ
0
$pk^A_{\mathtt{SP}}, pk^B_{\mathtt{SP}}$
Δ_T
$\sigma^A_{\mathtt{TX}_{\mathtt{RV},i}}, \sigma^B_{\mathtt{TX}_{\mathtt{RV},i}}$

Fig. 3. 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 $\mathtt{TX}_{\mathtt{RV},i}$ is stored in $\mathtt{TX}_{\mathtt{AU},i}$ and hence $\mathtt{TX}_{\mathtt{RV},i}$ 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 counter-party 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 $\texttt{TX}_{\texttt{CM},i}$. To solve the second mentioned issue, we add two statements from the hard relation $\mathcal{R}$, $R_{A,i}$ and $R_{B,i}$, to the first subcondition of the main output of $\texttt{TX}_{\texttt{CM},i}$, 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 counter-party'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 $\texttt{TX}_{\texttt{CM},i+1}$, $\texttt{TX}_{\texttt{AU},i+1}$ and $\texttt{TX}_{\texttt{SP},i+1}$. Thus, $\texttt{TX}_{\texttt{FU}}.\mathsf{txid}$, public keys $pk^A_{\texttt{RV}}$, $pk^B_{\texttt{RV}}$, $pk^A_{\texttt{SP}}$ and $pk^B_{\texttt{SP}}$ as well as r values of both parties are all data needed by the watchtower to watch the channel for both parties. Fig. 4 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 $\texttt{TX}_{\texttt{CM},j}$, party B uses $r_{A,i}$ to compute $r_{A,j}$. Then, B publishes the revocation transaction $\texttt{TX}_{\texttt{RV},j}$ and claims its output. If r values are randomly generated, the mentioned security requirement is met but storage cost of channel parties and the watchtower would be $\mathcal{O}(N)$. To reduce the storage and meeting 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 [9]. In more details, 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 counter-party, will be $\mathcal{O}(\log(N))$ because upon receipt of a node value, its child nodes' values can be removed from the storage.

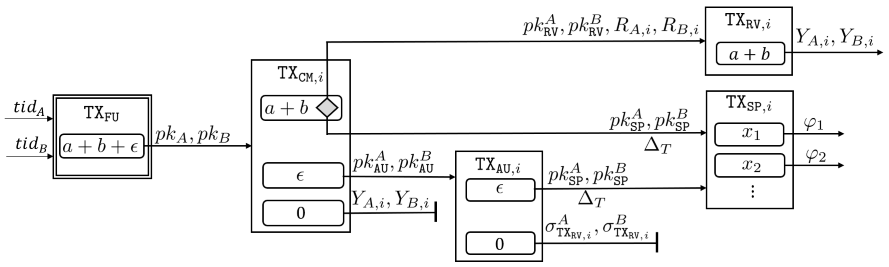

tid_A
tid_B
$\texttt{TX}_{\texttt{FU}}$
$a+b+\epsilon$
pk_A, pk_B
$\texttt{TX}_{\texttt{CM},i}$
$a+b$
ϵ
0
$pk^A_{\texttt{AU}}, pk^B_{\texttt{AU}}$
$Y_{A,i}, Y_{B,i}$
$\texttt{TX}_{\texttt{AU},i}$
ϵ
0
$pk^A_{\texttt{SP}}, pk^B_{\texttt{SP}}$
Δ_T
$\sigma^A_{\texttt{TX}_{\texttt{RV},i}}, \sigma^B_{\texttt{TX}_{\texttt{RV},i}}$
$pk^A_{\texttt{RV}}, pk^B_{\texttt{RV}}, R_{A,i}, R_{B,i}$
$\texttt{TX}_{\texttt{RV},i}$
$a+b$
$Y_{A,i}, Y_{B,i}$
$pk^A_{\texttt{SP}}, pk^B_{\texttt{SP}}$
Δ_T
$\texttt{TX}_{\texttt{SP},i}$
x_1
x_2
φ_1
φ_2

Fig. 4. Adding Y and R Values to Commit Transactions

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 counter-parties on the revocation transactions. Otherwise, the dishonest channel party publishes a revoked commit transaction $\mathtt{TX}_{\mathtt{CM},i}$ without publishing its auxiliary transaction $\mathtt{TX}_{\mathtt{AU},i}$. Then, the channel funds could be locked forever. This raises a hostage situation. The scheme Outpost suffers from this problem which is why storage requirement of channel parties is $\mathcal{O}(N)$. To solve this problem, we add one subcondition, $Y_{A,i} \wedge Y_{B,i} \wedge \Delta_{3T}$, to the main output of the commit transaction $\mathtt{TX}_{\mathtt{CM},i}$. This subcondition 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 $\mathtt{TX}_{\mathtt{CM},i}$, she has $3T$ rounds time to publish $\mathtt{TX}_{\mathtt{AU},i}$ and $\mathtt{TX}_{\mathtt{SP},i}$ before B can claim the channel funds by meeting the subcondition $Y_{A,i} \wedge Y_{B,i} \wedge \Delta_{3T}$. If during this interval, $\mathtt{TX}_{\mathtt{AU},i}$ is published, party B instantly establishes and publishes $\mathtt{TX}_{\mathtt{RV},i}$ 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. 5).

Once party A publishes $\mathtt{TX}_{\mathtt{CM},i}$, 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 $\tilde{\sigma}^{B}_{\mathtt{TX}_{\mathtt{CM},i}}$. 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 $\mathtt{TX}_{\mathtt{CM},i}$ is published by A, party B can regenerate the required random value, recompute the corresponding pre-signature $\tilde{\sigma}^{B}_{\mathtt{TX}_{\mathtt{CM},i}}$ 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 $\mathtt{TX}_{\mathtt{CM},j}$, A computes $y_{B,j}$ and hence derives $y_{B,i}$ with $i < j$ and then try to publish $\mathtt{TX}_{\mathtt{CM},i}$ before $\mathtt{TX}_{\mathtt{CM},j}$ 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 $\mathtt{TX}_{\mathtt{CM},i}$ or by publishing the revocation transaction $\mathtt{TX}_{\mathtt{RV},i}$ and claiming its output. If y values are randomly generated, the mentioned security requirement is met but 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 counter-parties from the root to the deepest leaf nodes in the tree.

4 Garrison Channel

We introduce different transactions of a Garrison channel in section 4.1. Then, in section 4.2, we explain its protocol.

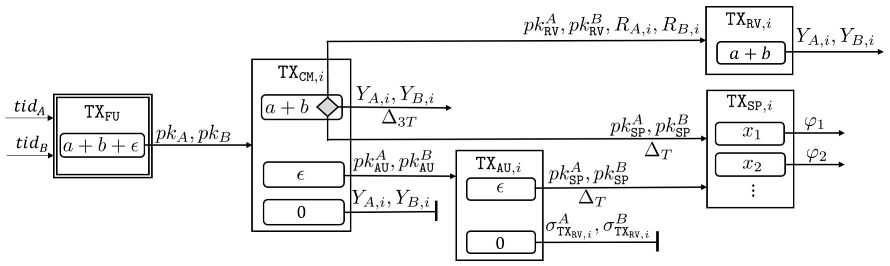

$\mathtt{TX}_{\mathtt{FU}}$
tid_A
tid_B
$a+b+\epsilon$
pk_A, pk_B
$\mathtt{TX}_{\mathtt{CM},i}$
$a+b$
ϵ
0
$Y_{A,i}, Y_{B,i}$
Δ_{3T}
$pk^A_{\mathtt{AU}}, pk^B_{\mathtt{AU}}$
$Y_{A,i}, Y_{B,i}$
$\mathtt{TX}_{\mathtt{AU},i}$
$pk^A_{\mathtt{SP}}, pk^B_{\mathtt{SP}}$
Δ_T
$\sigma^A_{\mathtt{TX}_{\mathtt{RV},i}}, \sigma^B_{\mathtt{TX}_{\mathtt{RV},i}}$
$pk^A_{\mathtt{RV}}, pk^B_{\mathtt{RV}}, R_{A,i}, R_{B,i}$
$\mathtt{TX}_{\mathtt{RV},i}$
$Y_{A,i}, Y_{B,i}$
$pk^A_{\mathtt{SP}}, pk^B_{\mathtt{SP}}$
Δ_T
$\mathtt{TX}_{\mathtt{SP},i}$
x_1
x_2
φ_1
φ_2

Fig. 5. Reducing Storage Requirements of Channel Parties

4.1 Garrison Transactions

Transactions of a Garrison channel are as following:

Funding transaction Parties A and B fund the channel by recording the funding transaction $\mathtt{TX}_{\mathtt{FU}}$ on the blockchain. The output of the funding transaction is a 2-of-2 multisignature address shared between A and B. 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]:

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

Commit transaction The commit transaction for state i is denoted by $\mathtt{TX}_{\mathtt{CM},i}$ and is as follows:

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

with $\varphi_1 := (pk^A_{\mathtt{RV}} \wedge pk^B_{\mathtt{RV}} \wedge R_{A,i} \wedge R_{B,i})$, $\varphi_2 := (Y_{A,i} \wedge Y_{B,i} \wedge \Delta_{3T})$, and $\varphi_3 := (pk^A_{\mathtt{SP}} \wedge pk^B_{\mathtt{SP}} \wedge \Delta_T)$ where $Y_{A,i}$ and $R_{A,i}$ ($Y_{B,i}$ and $R_{B,i}$) are statements of a hard relation $\mathcal{R}$ generated by A (B) for the i^{th} state and T is any number such that $T > \tau$. The first and second output of the transaction are the main and auxiliary outputs. Normally, if $\mathtt{TX}_{\mathtt{CM},i}$ is the last commit transaction and is published on-chain, first its auxiliary output and then its main output are

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

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

Remark 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 33-byte length, are stored.

Revocation transaction The revocation transaction for state i is denoted by $\mathtt{TX}_{\mathtt{RV},i}$ and is as follows:

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

The $\mathtt{TX}_{\mathtt{RV},i}$ spends the main output of $\mathtt{TX}_{\mathtt{CM},i}$ using its non-timelocked subcondition $pk^A_{\mathtt{RV}} \wedge pk^B_{\mathtt{RV}} \wedge R_{A,i} \wedge R_{B,i}$ 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 $\mathtt{TX}_{\mathtt{CM},i}$, she must publish $\mathtt{TX}_{\mathtt{AU},i}$ and then wait for T rounds before being able to publish $\mathtt{TX}_{\mathtt{SP},i}$. However, given that the state i is revoked, B knows the value of $r_{A,i}$ and hence creates the revocation transaction $\mathtt{TX}_{\mathtt{RV},i}$ and instantly publishes it on the blockchain. The output of $\mathtt{TX}_{\mathtt{RV},i}$ 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:

$$\begin{aligned}\mathtt{TX}_{\mathtt{AU},i}.\mathsf{Input} &:= \mathtt{TX}_{\mathtt{CM}}.\mathsf{txid}\|2,\\ \mathtt{TX}_{\mathtt{AU},i}.\mathsf{Output} &:= ((\epsilon, pk^A_{\mathtt{SP}} \wedge pk^B_{\mathtt{SP}} \wedge \Delta_T),\\ &\quad\ (0, (\sigma^A_{\mathtt{TX}_{\mathtt{RV},i}}, \sigma^B_{\mathtt{TX}_{\mathtt{RV},i}})))\\ \mathtt{TX}_{\mathtt{AU},i}.\mathsf{Witness} &:= \{(1, \{\sigma^A_{\mathtt{TX}_{\mathtt{AU},i}}, \sigma^B_{\mathtt{TX}_{\mathtt{AU},i}}\})\}\end{aligned}$$

This transaction spends the auxiliary output of the commit transaction and its output is spent by the split transaction. In other words, split transaction cannot be published unless auxiliary transaction is on the blockchain. The second output of $\mathtt{TX}_{\mathtt{AU},i}$ is an OP_RETURN output containing signatures of both parties on the corresponding revocation transaction.

Remark 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 $pk_{\texttt{RV}}^{A}$ and $pk_{\texttt{RV}}^{B}$ to form an aggregated public key $pk_{\texttt{RV}}$ [14] and change φ_1 in $\texttt{TX}_{\texttt{CM},i}$ to $(pk_{\texttt{RV}} \wedge R_{A,i} \wedge R_{B,i})$. 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 $\texttt{TX}_{\texttt{AU},i}$.

Split transaction $\texttt{TX}_{\texttt{SP},i}$ actually represents the i^{th} channel state and is as follows:

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

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

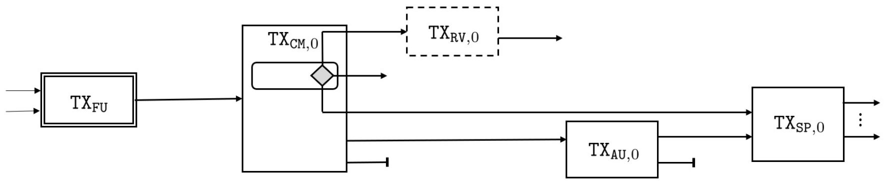
4.2 Garrison Protocol

The lifetime of a Garrison channel can be divided into 4 phases including *create*, *update*, *close*, and *punish*. These phases are explained, hereinafter.

Create The channel creation phase completes once the funding transaction $\texttt{TX}_{\texttt{FU}}$, the commit transactions $\texttt{TX}_{\texttt{CM},0}$, the split transaction $\texttt{TX}_{\texttt{SP},0}$, the auxiliary transaction $\texttt{TX}_{\texttt{AU},0}$ and body of the revocation transaction $[\texttt{TX}_{\texttt{RV},0}]$ are created, and $\texttt{TX}_{\texttt{FU}}$ is published on the blockchain. In this phase, parties do not have access to $\texttt{TX}_{\texttt{RV},0}$ as they have not exchanged $r_{A,i}$ and $r_{B,i}$ yet. At the end of the channel creation phase, the channel would be at state 0. Since 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 summarizes the channel creation phase.

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 $\texttt{TX}_{\texttt{CM},i+1}$, $\texttt{TX}_{\texttt{SP},i+1}$, $\texttt{TX}_{\texttt{AU},i+1}$, and $[\texttt{TX}_{\texttt{RV},i+1}]$ 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. 7 summarizes the channel update phase.

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



1- Create $[\texttt{TX}_{\texttt{FU}}]$

$A \xrightarrow{txid_A} B$

$A \xleftarrow{txid_B} B$

2- Create $[\texttt{TX}_{\texttt{CM},0}]$

$A \xrightarrow{R_{A,0}, Y_{A,0}} B$

$A \xleftarrow{R_{B,0}, Y_{B,0}} B$

3- Create $[\texttt{TX}_{\texttt{RV},0}]$

$A \xrightarrow{\sigma^{A}_{\texttt{TX}_{\texttt{RV},0}}} B$

$A \xleftarrow{\sigma^{B}_{\texttt{TX}_{\texttt{RV},0}}} B$

4- Create $[\texttt{TX}_{\texttt{AU},0}]$ **5- Create** $[\texttt{TX}_{\texttt{SP},0}]$

9- Create $\texttt{TX}_{\texttt{FU}}$

$A \xrightarrow{\sigma^{A}_{\texttt{TX}_{\texttt{FU}}}} B$

$A \xleftarrow{\sigma^{B}_{\texttt{TX}_{\texttt{FU}}}} B$

8- Create $\texttt{TX}_{\texttt{CM},0}$

$A \xrightarrow{\tilde{\sigma}^{A}_{\texttt{TX}_{\texttt{CM},0}}} B$

$A \xleftarrow{\tilde{\sigma}^{B}_{\texttt{TX}_{\texttt{CM},0}}} B$

7- Create $\texttt{TX}_{\texttt{AU},0}$

$A \xrightarrow{\sigma^{A}_{\texttt{TX}_{\texttt{AU},0}}} B$

$A \xleftarrow{\sigma^{B}_{\texttt{TX}_{\texttt{AU},0}}} B$

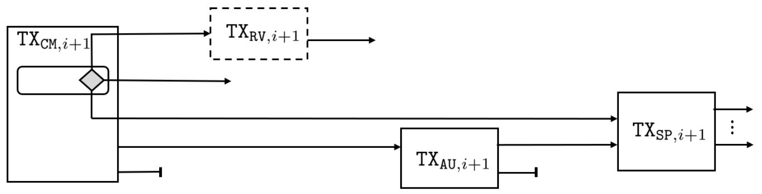
6- Create $\texttt{TX}_{\texttt{SP},0}$

$A \xrightarrow{\sigma^{A}_{\texttt{TX}_{\texttt{SP},0}}} B$

$A \xleftarrow{\sigma^{B}_{\texttt{TX}_{\texttt{SP},0}}} B$

10- Publish $\texttt{TX}_{\texttt{FU}}$

Fig. 6. Summary of Garrison channel creation phase.



1- Create $[\texttt{TX}_{\texttt{CM},i+1}]$

$A \xrightarrow{R_{A,i+1}, Y_{A,i+1}} B$

$A \xleftarrow{R_{B,i+1}, Y_{B,i+1}} B$

2- Create $[\texttt{TX}_{\texttt{RV},i+1}]$

$A \xrightarrow{\sigma^{A}_{\texttt{TX}_{\texttt{RV},i+1}}} B$

$A \xleftarrow{\sigma^{B}_{\texttt{TX}_{\texttt{RV},i+1}}} B$

3- Create $[\texttt{TX}_{\texttt{AU},i+1}]$ **4- Create** $[\texttt{TX}_{\texttt{SP},i+1}]$

7- Create $\texttt{TX}_{\texttt{CM},i+1}$

$A \xrightarrow{\tilde{\sigma}^{A}_{\texttt{TX}_{\texttt{CM},i+1}}} B$

$A \xleftarrow{\tilde{\sigma}^{B}_{\texttt{TX}_{\texttt{CM},i+1}}} B$

6- Create $\texttt{TX}_{\texttt{AU},i+1}$

$A \xrightarrow{\sigma^{A}_{\texttt{TX}_{\texttt{AU},i+1}}} B$

$A \xleftarrow{\sigma^{B}_{\texttt{TX}_{\texttt{AU},i+1}}} B$

5- Create $\texttt{TX}_{\texttt{SP},i+1}$

$A \xrightarrow{\sigma^{A}_{\texttt{TX}_{\texttt{SP},i+1}}} B$

$A \xleftarrow{\sigma^{B}_{\texttt{TX}_{\texttt{SP},i+1}}} B$

8- Revoke $\texttt{TX}_{\texttt{CM},i}$

$A \xrightarrow{r_{A,i}} B$

$A \xleftarrow{r_{B,i}} B$

Fig. 7. Summary of Garrison channel update phase from state i to $i+1$.

cooperatively. To do so, A and B create the below transaction, called modified split transaction $\texttt{TX}_{\overline{\texttt{SP}}}$, and publish it on the blockchain:

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

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 $\mathtt{TX}_{\mathtt{CM},n}$ and $\mathtt{TX}_{\mathtt{AU},n}$ on the ledger. Then, she waits for T rounds, and finally publishes $\mathtt{TX}_{\mathtt{SP},n}$.

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

Remark 3. If the watchtower is non-responsive, the channel might be closed with an old state. The paper [19] proposes a reputation system, called HashCashed, which forces watchtowers to be responsive without requiring them to lock any funds as collateral. We assume that Garrison is used with HashCashed system.

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 1. *Let Π be a $\mathsf{EUF-CMA}$ secure digital signature, $\mathcal{R}$ be a hard relation and Ξ be a secure adaptor digital signature. Then, for a Garrison channel with n channel updates, the broadcast of $\mathtt{TX}_{\mathtt{RV},i}$ 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 $\mathtt{TX}_{\mathtt{RV},i}$ with $i < n$ spends the main output of the revoked $\mathtt{TX}_{\mathtt{CM},i}$ and hence cannot be published unless $\mathtt{TX}_{\mathtt{CM},i}$ is on-chain. The transaction $\mathtt{TX}_{\mathtt{CM},i}$ spends the output of $\mathtt{TX}_{\mathtt{FU}}$. Since the condition in $\mathtt{TX}_{\mathtt{FU}}.\mathsf{Output}$ contains pk_A, this output cannot be spent without A's authorization. Otherwise, security of the underlying digital signature would be violated. Based on the protocol, the honest party A never broadcasts the revoked $\mathtt{TX}_{\mathtt{CM},i}$ on-chain and her pre-signature $\tilde{\sigma}_{\mathtt{TX}_{\mathtt{CM},i}}$ on the transaction $\mathtt{TX}_{\mathtt{CM},i}$ is the only authorization he grants for spending $\mathtt{TX}_{\mathtt{FU}}.\mathsf{Output}$ using $\mathtt{TX}_{\mathtt{CM},i}$. Thus, if $\mathtt{TX}_{\mathtt{CM},i}$ is published, the probability that A fails to obtain $y_{B,i}$ is negligible. Otherwise, $\mathsf{aEUF-CMA}$ security or witness extractability of the used adaptor signature is violated. Furthermore, $\mathtt{TX}_{\mathtt{RV},i}$ 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 $\mathtt{TX}_{\mathtt{RV},i}.\mathsf{Output}$ is negligible. Otherwise, the utilized 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 $\mathtt{TX}_{\mathtt{RV},i}.\mathsf{Output}$ and obtain all the channel funds.

Lemma 2. *Let Π be a $\mathsf{EUF-CMA}$ secure digital signature, $\mathcal{R}$ 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 counter-party publishes $\mathtt{TX}_{\mathtt{CM},i}$, it is with negligible probability that*

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

Proof. Without loss of generality let $P = A$. Similar to the proof of Lemma 1, if B publishes $\mathtt{TX}_{\mathtt{CM},i}$, the probability that A fails to obtain $y_{B,i}$ is negligible. Otherwise, $\mathsf{aEUF-CMA}$ 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 $Y_{A,i} \wedge Y_{B,i} \wedge \Delta_{3T}$. Furthermore, given that A privately preserves its witness value $y_{A,i}$, the probability that any PPT adversary meets this subcondition is negligible. Otherwise, the utilized hard relation would break.

Lemma 3. *Let Π be a* $\mathsf{EUF-CMA}$ *secure digital signature, $\mathcal{R}$ 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* $\mathtt{TX}_{\mathtt{CM},n}$*, P loses funds in the channel with negligible probability.*

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

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

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

Lemma 4. *Let Π be a* $\mathsf{EUF-CMA}$ *secure digital signature, $\mathcal{R}$ 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 $\texttt{TX}_{\texttt{CM},n}$, 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 3. The only difference is that following Lemma 2, it is of negligible probability that A fails to meet the second subcondition of $\texttt{TX}_{\texttt{CM},n}.\mathsf{Output}[1]$. Therefore, A can either publishes both $\texttt{TX}_{\texttt{AU},n}$ and $\texttt{TX}_{\texttt{SP},n}$ or claim $\texttt{TX}_{\texttt{CM},n}.\mathsf{Output}[1]$ by meeting its second subcondition. None of these two cases can cause the honest party A to lose any funds in the channel.

Lemma 5. *Let Π be a $\mathsf{EUF-CMA}$ secure digital signature, $\mathcal{R}$ 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 $\texttt{TX}_{\texttt{CM},i}$ 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 $\texttt{TX}_{\texttt{CM},i}.\mathsf{Output}[1]$ includes 3 subconditions, one of which must be met to cheat A out of its funds. The first subcondition contains $pk^A_{\texttt{RV}}$ and hence it is of negligible probability that this output is spent without A's authorization. Otherwise, the security of the used digital signature is violated. The honest party A grants such an authorization only on the transaction $\texttt{TX}_{\texttt{RV},i}$. However, according to Lemma 1, it is of negligible probability that broadcast of $\texttt{TX}_{\texttt{RV},i}$ causes A to lose any funds. Moreover, according Lemma 2, it is of negligible probability that any PPT adversary can meet the second subcondition. Now, we prove that if the third subcondition is used to cheat A out of her funds, it leads to a contradiction.

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

Theorem 1. *Let Π be a $\mathsf{EUF-CMA}$ secure digital signature, $\mathcal{R}$ 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 $\mathtt{TX}_{\mathtt{FU}}$. Output. It is of negligible probability that any PPT adversary $\mathcal{A}$ spends the output of $\mathtt{TX}_{\mathtt{FU}}$ without the honest party A's authorization. Otherwise, the underlying digital signature would be forgeable. Furthermore, $\mathtt{TX}_{\overline{\mathtt{SP}}}$, $\mathtt{TX}_{\mathtt{CM},i}$ with $i = [0, n-1]$, $\mathtt{TX}_{\mathtt{CM},n}$ are the only transactions in the protocol that spend the output of $\mathtt{TX}_{\mathtt{FU}}$ and A grants authorization for. Thus, these transactions will be discussed further. Since $\mathtt{TX}_{\overline{\mathtt{SP}}}$ represents the final agreed state of the channel, its broadcast cannot cause A to lose any funds. Moreover, according to Lemmas 3 and 4, it is of negligible probability that broadcast of $\mathtt{TX}_{\mathtt{CM},n}$ causes A to be cheated out of her funds. Also, based on the protocol, A never publishes $\mathtt{TX}_{\mathtt{CM},i}$ with $i = [0, n-1]$ and according to Lemma 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.

References

1. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostakova, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized bitcoin-compatible channels. IACR Cryptol. ePrint Arch. **2020**, 476 (2020)
2. Aumayr, L., Maffei, M., Ersoy, O., Erwig, A., Faust, S., Riahi, S., Hostáková, K., Moreno-Sanchez, P.: Bitcoin-compatible virtual channels. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 901–918. IEEE (2021)
3. Aumayr, L., Thyagarajan, S.A., Malavolta, G., Moreno-Sanchez, P., Maffei, M.: Sleepy channels: Bitcoin-compatible bi-directional payment channels without watchtowers. Cryptology ePrint Archive (2021)
4. Avarikioti, G., Laufenberg, F., Sliwinski, J., Wang, Y., Wattenhofer, R.: Towards secure and efficient payment channels. arXiv preprint arXiv:1811.12740 (2018)
5. Avarikioti, G., Litos, O.S.T., Wattenhofer, R.: Cerberus channels: Incentivizing watchtowers for bitcoin. Financial Cryptography and Data Security (FC) (2020)
6. Decker, C.: Bip 118 - sighashnoinput. URl: https://github.com/bitcoin/bips/blob/master/bip-0118.mediawiki (2017)
7. Decker, C., Russell, R., Osuntokun, O.: eltoo: A simple layer2 protocol for bitcoin. White paper: https://blockstream. com/eltoo. pdf (2018)
8. Decker, C., Wattenhofer, R.: A fast and scalable payment network with bitcoin duplex micropayment channels. In: Symposium on Self-Stabilizing Systems. pp. 3–18. Springer (2015)
9. Developers, L.: Bolt# 3: Bitcoin transaction and script formats (2017)
10. Dryja, T., Milano, S.B.: Unlinkable outsourced channel monitoring. Talk transcript) https://diyhpl. us/wiki/transcripts/scalingbitcoin/milan/unlinkable-outsourced-channel-monitoring (2016)
11. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. In: Annual International Cryptology Conference. pp. 291–323. Springer (2017)
12. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ecdsa). International journal of information security **1**(1), 36–63 (2001)
13. Khabbazian, M., Nadahalli, T., Wattenhofer, R.: Outpost: A responsive lightweight watchtower. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies. pp. 31–40 (2019)

14. Lindell, Y.: Fast secure two-party ecdsa signing. In: Annual International Cryptology Conference. pp. 613–644. Springer (2017)
15. McCorry, P., Bakshi, S., Bentov, I., Meiklejohn, S., Miller, A.: Pisa: Arbitration outsourcing for state channels. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies. pp. 16–30 (2019)
16. Mirzaei, A., Sakzad, A., Yu, J., Steinfeld, R.: Fppw: A fair and privacy preserving watchtower for bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 151–169. Springer (2021)
17. Mirzaei, A., Sakzad, A., Yu, J., Steinfeld, R.: Daric: A storage efficient payment channel with penalization mechanism. Cryptology ePrint Archive, Report 2022/1295 (2022), https://eprint.iacr.org/2022/1295
18. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016)
19. Rahimpour, S., Khabbazian, M.: Hashcashed reputation with application in designing watchtowers. In: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 1–9. IEEE (2021)
20. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of cryptology **4**(3), 161–174 (1991)
21. Spilman, J.: [bitcoin-development] anti dos for tx replacement. available at: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/ 2013-April/002433.html (2013)