Supplementary material: Bitcoin-Compatible Virtual Channels

A. Rooted Transactions

UTXO based blockchains can be viewed as a directed acyclic graph, where each node represents a transaction. Nodes corresponding to transactions tx_i and tx_j are connected with an edge if at least one of the outputs of tx_i is an input of tx_j, i.e, tx_i is (partially) funding tx_j. We denote the transitive reachability relation between nodes, which constitutes a partial order, as $\leq$. We say that a transaction tx is *rooted* in the set of transactions R if (1) $\forall \mathsf{tx}_i \leq \mathsf{tx}.\exists \mathsf{tx}_j \in R.\mathsf{tx}_j \leq \mathsf{tx}_i \vee \mathsf{tx}_i \leq \mathsf{tx}_j$, (2) $\forall \mathsf{tx}_i, \mathsf{tx}_j \in R.\mathsf{tx}_i \neq \mathsf{tx}_j, \mathsf{tx}_i \not\leq \mathsf{tx}_j$, (3) $\mathsf{tx} \notin R$.

Let us now see an example of rooted transactions to clarify this concept.

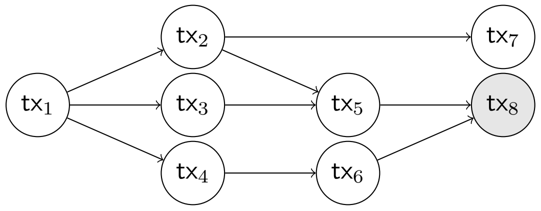

Fig. 1: The root sets of transaction tx_8 are $\{\mathsf{tx}_1\}$, $\{\mathsf{tx}_2, \mathsf{tx}_3, \mathsf{tx}_4\}$, $\{\mathsf{tx}_5, \mathsf{tx}_6\}$, $\{\mathsf{tx}_4, \mathsf{tx}_5\}$ and $\{\mathsf{tx}_2, \mathsf{tx}_3, \mathsf{tx}_6\}$.

B. On the usage of the UC-Framework

To formally model the security of our construction, we use a synchronous version of the global UC framework (GUC) [4] which extends the standard UC framework [3] by allowing for a global setup. Since our model is essentially the same as in [1], which in turn follows [5, 6], parts of this section are taken verbatim from there.

a) Protocols and adversarial model: We consider a protocol π that runs between parties from the set $\mathcal{P} = \{P_1, \ldots, P_n\}$. A protocol is executed in the presence of an *adversary* $\mathcal{A}$ that takes as input a security parameter 1^λ (with $\lambda \in \mathbb{N}$) and an auxiliary input $z \in \{0,1\}^*$, and who can *corrupt* any party P_i at the beginning of the protocol execution (so-called static corruption). By corruption we mean that $\mathcal{A}$ takes full control over P_i and learns its internal state. Parties and the adversary $\mathcal{A}$ receive their inputs from a special entity – called the *environment* $\mathcal{E}$ – which represents anything "external" to the current protocol execution. The environment also observes all outputs returned by the parties of the protocol. In addition to the above entities, the parties can have access to ideal functionalities $\mathcal{H}_1, \ldots, \mathcal{H}_m$. In this case we say that the protocol π *works in the* $(\mathcal{H}_1, \ldots, \mathcal{H}_m)$*-hybrid model* and write $\pi^{\mathcal{H}_1, \ldots, \mathcal{H}_m}$.

b) Modeling time and communication: We assume a synchronous communication network, which means that the execution of the protocol happens in rounds. Let us emphasize that the notion of rounds is just an abstraction which simplifies our model and allows us to argue about the time complexity of our protocols in a natural way. We follow [6], which in turn follows [8], and formalize the notion of rounds via an ideal functionality $\widehat{\mathcal{F}}_{clock}$ representing "the clock". On a high level, the ideal functionality requires all honest parties to indicate that they are prepared to proceed to the next round before the clock is "ticked". We treat the clock functionality as a *global* ideal functionality using the GUC model. This means that all entities are always aware of the given round.

We assume that parties of a protocol are connected via authenticated communication channels with guaranteed delivery of exactly one round. This means that if a party P sends a message m to party Q in round t, party Q receives this message in beginning of round $t+1$. In addition, Q is sure that the message was sent by party P. The adversary can see the content of the message and can reorder messages that were sent in the same round. However, it can not modify, delay or drop messages sent between parties, or insert new messages. The assumptions on the communication channels are formalized as an ideal functionality $\mathcal{F}_{GDC}$. We refer the reader to [6] its formal description.

While the communication between two parties of a protocol takes exactly one round, all other communication – for example, between the adversary $\mathcal{A}$ and the environment $\mathcal{E}$ – takes zero rounds. For simplicity, we assume that any computation made by any entity takes zero rounds as well.

c) Handling coins: We model the money mechanics offered by UTXO cryptocurrencies, such as Bitcoin, via a *global* ideal functionality $\widehat{\mathcal{L}}$ using the GUC model. Our functionality is parameterized by a *delay parameter* Δ which upper bounded in the maximal number of rounds it takes to publish a valid transaction, and a signature scheme Σ. The functionality accepts messages from a fixed set of parties $\mathcal{P}$.

The ledger functionality $\widehat{\mathcal{L}}$ is initiated by the environment $\mathcal{E}$ via the following steps: (1) $\mathcal{E}$ instructs the ledger functionality to generate public parameter of the signature scheme pp; (2) $\mathcal{E}$ instructs every party $P \in \mathcal{P}$ to generate a key pair (sk_P, pk_P) and submit the public key pk_P to the ledger via the message $(\mathrm{register}, pk_P)$; (3) sets the initial state of the ledger meaning that it initialize a set $\mathtt{TX}$ defining all published transactions.

Once initialized, the state of $\widehat{\mathcal{L}}$ is public and can be accessed by all parties of the protocol, the adversary $\mathcal{A}$ and the environment $\mathcal{E}$. Any party $P \in \mathcal{P}$ can at any time post a transaction on the ledger via the message $(\text{post}, \mathsf{tx})$. The ledger functionality waits for at most Δ rounds (the exact number of rounds is determined by the adversary). Thereafter, the ledger verifies the validity of the transaction and adds it to the transaction set TX. The formal description of the ledger functionality follows.

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

The functionality accepts messages from all parties that are in the set $\mathcal{P}$ and maintains a PKI for those parties. The functionality maintains the set of all accepted transactions TX and all unspent transaction outputs UTXO. The set $\mathcal{V}$ defines valid output conditions.

Initialize public keys: Upon $(\text{register}, pk_P) \overset{\tau_0}{\hookleftarrow} P$ and it is the first time P sends a registration message, add (pk_P, P) to PKI.

Post transaction: Upon $(\text{post}, \mathsf{tx}) \overset{\tau_0}{\hookleftarrow} P$, check that $|\mathtt{PKI}| = |\mathcal{P}|$. If not, drop the message, else wait until round $\tau_1 \leq \tau_0 + \Delta$ (the exact value of τ_1 is determined by the adversary). Then check if:

1) The id is unique, i.e. for all $(t, \mathsf{tx}') \in \mathtt{TX}$, $\mathsf{tx}'.\mathsf{txid} \neq \mathsf{tx}.\mathsf{txid}$.
2) All the inputs are unspent and the witness satisfies all the output conditions, i.e. for each $(tid, i) \in \mathsf{tx}.\mathsf{Input}$, there exists $(t, tid, i, \theta) \in \mathtt{UTXO}$ and $\theta.\varphi(\mathsf{tx}, t, \tau_1) = 1$.
3) All outputs are valid, i.e. for each $\theta \in \mathsf{tx}.\mathsf{Output}$ it holds that $\theta.\mathsf{cash} > 0$ and $\theta.\varphi \in \mathcal{V}$.
4) The value of the outputs is not larger than the value of the inputs. More formally, let $I := \{\mathtt{utxo} := (t, tid, i, \theta) \mid \mathtt{utxo} \in \mathtt{UTXO} \wedge (tid, i) \in \mathsf{tx}.\mathsf{Input}\}$, then it must hold that $\sum_{\theta' \in \mathsf{tx}.\mathsf{Output}} \theta'.\mathsf{cash} \leq \sum_{\mathtt{utxo} \in I} \mathtt{utxo}.\theta.\mathsf{cash}$
5) The absolute time-lock of the transaction has expired, i.e. it must hold that $\mathsf{tx}.\mathsf{TimeLock} \leq \mathsf{now}$.

If all the above checks return true, add (τ_1, tx) to TX, remove the spent outputs from UTXO, i.e., $\mathtt{UTXO} := \mathtt{UTXO} \setminus I$ and add the outputs of tx to UTXO, i.e., $\mathtt{UTXO} := \mathtt{UTXO} \cup \{(\tau_1, \mathsf{tx}.\mathsf{txid}, i, \theta_i)\}_{i \in [n]}$ for $(\theta_1, \dots, \theta_n) := \mathsf{tx}.\mathsf{Output}$. Else, ignore the message.

Let us emphasize that our ledger functionality is fairly simplified. In reality, parties can join and leave the blockchain system dynamically. Moreover, we completely abstract from the fact that transactions are published in blocks which are proposed by parties and the adversary. Those and other features are captured by prior works, such as [2], that provide a more accurate formalization of the Bitcoin ledger in the UC framework [3]. However, interaction with such ledger functionality is fairly complex. To increase the readability of our channel protocols and ideal functionality, which is the main focus on our work, we decided for this simpler ledger.

d) The GUC-security definition: Let π be a protocol with access to the global ledger $\widehat{\mathcal{L}}(\Delta, \Sigma)$, the global clock $\widehat{\mathcal{F}}_{clock}$ and ideal functionalities $\mathcal{H}_1, \dots, \mathcal{H}_m$. The output of an environment $\mathcal{E}$ interacting with a protocol π and an adversary $\mathcal{A}$ on input 1^λ and auxiliary input z is denoted as

$$\mathrm{EXE}_{\pi,\mathcal{A},\mathcal{E}}^{\widehat{\mathcal{L}}(\Delta,\Sigma),\widehat{\mathcal{F}}_{clock},\mathcal{H}_1,\dots,\mathcal{H}_m}(\lambda, z).$$

Let $\phi_{\mathcal{F}}$ be the ideal protocol for an ideal functionality $\mathcal{F}$ with access to the global ledger $\widehat{\mathcal{L}}(\Delta, \Sigma)$ and the global clock $\widehat{\mathcal{F}}_{clock}$. This means that $\phi_{\mathcal{F}}$ is a trivial protocol in which the parties simply forward their inputs to the ideal functionality $\mathcal{F}$. The output of an environment $\mathcal{E}$ interacting with a protocol $\phi_{\mathcal{F}}$ and a adversary $\mathcal{S}$ (sometimes also call *simulator*) on input 1^λ and auxiliary input z is denoted as

$$\mathrm{EXE}_{\phi_{\mathcal{F}},\mathcal{S},\mathcal{E}}^{\widehat{\mathcal{L}}(\Delta,\Sigma),\widehat{\mathcal{F}}_{clock}}(\lambda, z).$$

We are now ready to state our main security definition which, informally, says that if a protocol π UC-realizes an ideal functionality $\mathcal{F}$, then any attack that can be carried out against the real-world protocol π can also be carried out against the ideal protocol $\phi_{\mathcal{F}}$.

Definition 1: A protocol π working in a $(\mathcal{H}_1, \dots, \mathcal{H}_m)$-hybrid model *UC-realizes an ideal functionality $\mathcal{F}$ with respect to a global ledger* $\widehat{\mathcal{L}} := \widehat{\mathcal{L}}(\Delta, \Sigma)$ and a global clock $\widehat{\mathcal{F}}_{clock}$ if for every adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that we have

$$\left\{\mathrm{EXE}_{\pi,\mathcal{A},\mathcal{E}}^{\widehat{\mathcal{L}},\widehat{\mathcal{F}}_{clock},\mathcal{H}_1,\dots,\mathcal{H}_m}(\lambda, z)\right\}_{\substack{\lambda\in\mathbb{N},\\ z\in\{0,1\}^*}} \overset{c}{\approx} \left\{\mathrm{EXE}_{\phi_{\mathcal{F}},\mathcal{S},\mathcal{E}}^{\widehat{\mathcal{L}},\widehat{\mathcal{F}}_{clock}}(\lambda, z)\right\}_{\substack{\lambda\in\mathbb{N},\\ z\in\{0,1\}^*}}$$

(where "$\overset{c}{\approx}$" denotes computational indistinguishability of distribution ensembles, see, e.g., [7]).
To simplify exposition, we omit the session identifiers *sid* and the sub-session identifiers *ssid*. Instead, we will use expressions like "message m is a reply to message m'". We believe that this approach improves readability.

I. Additional material to ledger channels

A. Ledger channel functionality

For completeness, we recall the ledger channel ideal functionality from [1].

Ideal Functionality $\mathcal{F}_L(T, k)$

We abbreviate $Q := \gamma.\mathsf{otherParty}(P)$ for $P \in \gamma.\mathsf{endUsers}$.

Create

Upon $(\mathtt{CREATE}, \gamma, tid_P) \overset{\tau_0}{\hookleftarrow} P$, let $\mathcal{S}$ define $T_1 \leq T$ and:

Both agreed: If already received $(\mathtt{CREATE}, \gamma, tid_Q) \overset{\tau}{\hookleftarrow} Q$, where $\tau_0 - \tau \leq T_1$, wait if in round $\tau_1 \leq \tau + \Delta + T_1$ a transaction tx, with $\mathsf{tx}.\mathsf{Input} = (tid_P, tid_Q)$ and $\mathsf{tx}.\mathsf{Output} = (\gamma.\mathsf{cash}, \varphi)$, appears on the ledger $\widehat{\mathcal{L}}$. If yes, set $\Gamma(\gamma.\mathsf{id}) := (\gamma, \mathsf{tx})$ and $(\mathtt{CREATED}, \gamma.\mathsf{id}) \overset{\tau_1}{\hookrightarrow} \gamma.\mathsf{endUsers}$. Else stop.
Wait for Q: Else store the message and stop.

Update

Upon $(\mathtt{UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \overset{\tau_0}{\hookleftarrow} P$, let $\mathcal{S}$ define $T_1, T_2 \leq T$, parse $(\gamma, \mathsf{tx}) := \Gamma(id)$ and proceed as follows:

1) In round $\tau_1 \leq \tau_0 + T$, let $\mathcal{S}$ set $|\vec{tid}| = k$. Then $(\mathtt{UPDATE\text{-}REQ}, id, \vec{\theta}, t_{\mathsf{stp}}, \vec{tid}) \overset{\tau_1}{\hookrightarrow} Q$ and $(\mathtt{SETUP}, id, \vec{tid}) \overset{\tau_1}{\hookrightarrow} P$.
2) If $(\mathtt{SETUP\text{-}OK}, id) \overset{\tau_2 \leq \tau_1 + t_{\mathsf{stp}}}{\hookleftarrow} P$, then $(\mathtt{SETUP\text{-}OK}, id) \overset{\tau_2 + T_1}{\hookrightarrow} Q$. Else stop.
3) If $(\mathtt{UPDATE\text{-}OK}, id) \overset{\tau_2 + T_1}{\hookleftarrow} Q$, then $(\mathtt{UPDATE\text{-}OK}, id) \overset{\tau_2 + 2T_1}{\hookrightarrow} P$. Else distinguish:
   - If Q honest or if instructed by $\mathcal{S}$, stop (update rejected).
   - Else execute $\mathtt{L\text{-}ForceClose}(id)$ and stop.

4) If $(\texttt{REVOKE}, id) \xhookleftarrow{\tau_2+2T_1} P$, $(\texttt{REVOKE-REQ}, id) \xhookrightarrow{\tau_2+2T_1+T_2} Q$. Else execute $\texttt{L-ForceClose}(id)$ and stop.
5) If $(\texttt{REVOKE}, id) \xhookleftarrow{\tau_2+2T_1+T_2} Q$, set $\gamma.\mathsf{st} = \vec{\theta}$ and $\Gamma(id) := (\gamma, \mathsf{tx})$. Then $(\texttt{UPDATED}, id, \vec{\theta}) \xhookrightarrow{\tau_2+2T_1+2T_2} \gamma.\mathsf{endUsers}$ and stop. Else distinguish:
   - If Q honest, execute $\texttt{L-ForceClose}(id)$ and stop.
   - If Q corrupt, and wait for Δ rounds. If tx still unspent, then set $\vec{\theta}_{old} := \gamma.\mathsf{st}$, $\gamma.\mathsf{st} := \{\vec{\theta}_{old}, \vec{\theta}\}$ and $\Gamma(id) := (\gamma, \mathsf{tx})$. Execute $\texttt{L-ForceClose}(id)$ and stop.

$\boxed{\text{Close}}$

Upon $(\texttt{CLOSE}, id) \xhookleftarrow{\tau_0} P$, let $\mathcal{S}$ define $T_1 \leq T$ and distinguish:

Both agreed: If you received $(\texttt{CLOSE}, id) \xhookleftarrow{\tau} Q$, where $\tau_0 - \tau \leq T_1$, let $(\gamma, \mathsf{tx}) := \Gamma(id)$ and distinguish:

- If in round $\tau_1 \leq \tau + T_1 + \Delta$ a transaction tx', with $\mathsf{tx}'.\mathsf{Output} = \gamma.\mathsf{st}$ and $\mathsf{tx}'.\mathsf{Input} = \mathsf{tx.txid}$, appears on $\widehat{\mathcal{L}}$, set $\Gamma(id) := (\bot, \mathsf{tx})$, $(\texttt{CLOSED}, id) \xhookrightarrow{\tau_1} \gamma.\mathsf{endUsers}$ and stop.
- If tx is still unspent in round $\tau + T_1 + \Delta$, output $(\texttt{ERROR}) \xhookrightarrow{\tau+T_1+\Delta} \gamma.\mathsf{endUsers}$ and stop.

Wait for Q: Else wait for at most T_1 rounds to receive $(\texttt{CLOSE}, id) \xhookleftarrow{\tau \leq \tau_0+T_1} Q$ (in that case option "Both agreed" is executed). If such message is not received, execute $\texttt{L-ForceClose}(id)$ in round $\tau_0 + T_1$.

$\boxed{\text{Punish (executed at the end of every round } \tau_0)}$

For each $(\gamma, \mathsf{tx}) \in \Gamma$ check if $\widehat{\mathcal{L}}$ contains tx' with $\mathsf{tx}'.\mathsf{Input} = \mathsf{tx.txid}$. If yes, then distinguish:

Punish: For $P \in \gamma.\mathsf{endUsers}$ honest, the following must hold: in round $\tau_1 \leq \tau_0 + \Delta$, a transaction tx'' with $\mathsf{tx}''.\mathsf{Input} = \mathsf{tx}'.\mathsf{txid}$ and $\mathsf{tx}''.\mathsf{Output} = (\gamma.\mathsf{cash}, \texttt{One-Sig}_{pk_P})$ appears on $\widehat{\mathcal{L}}$. Then send $(\texttt{PUNISHED}, id) \xhookrightarrow{\tau_1} P$, set $\Gamma(id) := \bot$ and stop.
Close: Either $\Gamma(id) = (\bot, \mathsf{tx})$ before round $\tau_0 + \Delta$ (channels was peacefully closed) or in round $\tau_1 \leq \tau_0 + 2\Delta$ a transaction tx'', with $\mathsf{tx}''.\mathsf{Output} \in \gamma.\mathsf{st}$ and $\mathsf{tx}''.\mathsf{Input} = \mathsf{tx}'.\mathsf{txid}$, appears on $\widehat{\mathcal{L}}$ (channel is forcefully closed). In the latter case, set $\Gamma(id) := (\bot, \mathsf{tx})$ and $(\texttt{CLOSED}, id) \xhookrightarrow{\tau_1} \gamma.\mathsf{endUsers}$.
Error: Otherwise $(\texttt{ERROR}) \xhookrightarrow{\tau_0+2\Delta} \gamma.\mathsf{endUsers}$.

$\boxed{\text{Subprocedure } \texttt{L-ForceClose}(id)}$

Let τ_0 be the current round and $(\gamma, \mathsf{tx}) := \Gamma(id)$. If within Δ rounds tx is still an unspent transaction on $\widehat{\mathcal{L}}$, then $(\texttt{ERROR}) \xhookrightarrow{\tau_0+\Delta} \gamma.\mathsf{endUsers}$ and stop. Else, latest in round $\tau_0 + 3\Delta$, $m \in \{\texttt{CLOSED}, \texttt{PUNISHED}, \texttt{ERROR}\}$ is output via Punish.

B. Wrapped ledger channel functionality

As discussed already in **??**, for technical reason we cannot use the ledger channel functionality ($\mathcal{F}_L(T,k)$) for building virtual channels in a black-box way. The main problem comes from the offloading feature of virtual channel. In order to overcome these issues, we present $\mathcal{F}_{preL}(T,k)$, an ideal functionality that extends $\mathcal{F}_L(T,k)$ to supports the preparation generalized channels ahead of time and later registration of such prepared generalized channels. Technically, the functionality extension is done by *wrapping* the original functionality. Before we present the functionality wrapper $\mathcal{F}_{preL}(T,k)$ formally, let us explain each of its parts on high level.

a) Generalized channels: The functionality treats messages about standard generalized channels exactly as the functionality $\mathcal{F}_L(T,k)$ presented in Section I-A.

b) Creation: In order to pre-create a generalized channel γ, both end-users of the channel must the message $(\texttt{PRE-CREATE}, \gamma, \texttt{TX}_\texttt{f}, i, t_{ofl})$ to the ideal functionality. Here $\texttt{TX}_\texttt{f}||i$ identifies the funding of the channel and $t_{ofl} \in \mathbb{N}$ represents the maximal number of round it should take to publish the channel funding transaction on-chain. If the functionality receives such a message from both parties within T rounds, it stores the channel γ, the funding identifier and the waiting time to a special channel set Γ_{pre}, and informs both parties about the successful pre-creation.

c) Update: The update process works similarly as for standard ledger channel with one difference. If the update process fails at some point (e.g., one of the parties does not revoke), the functionality does not call $\texttt{L-ForceClose}$ since there is no ledger channel to be forcefully closed. Instead, it calls a subprocedure called $\texttt{Wait-if-Register}$ which add a flag "$\texttt{in-dispute}$" to the channel and waits for at most t_{ofl} rounds if the prepared channel is turned into a standard generalized channel (i.e., the corresponding funding transaction is added to the blockchain). If not, then it adds the new channel state back into the set of prepared (not yet full-fledged) channel states.

d) Register: The ideal functionality constantly monitors the ledger. Once the funding transaction of one of the channels in preparation appears on-chain, the functionality moves the information about the channel from the channel space Γ_{pre} to the channel space Γ. Moreover, if the channel in preparation was marked as "$\texttt{in-dispute}$", then it immediately calls $\texttt{L-ForceClose}$.

The formal functionality description on the functionality wrapper $\mathcal{F}_{preL}(T,k)$ follows.

Wrapped Ledger Channel Functionality $\mathcal{F}_{preL}(T,k)$

We abbreviate $Q := \gamma.\mathsf{otherParty}(P)$ for $P \in \gamma.\mathsf{endUsers}$.

$\boxed{\text{Ledger Channels}}$

Upon receiving a $\texttt{CREATE}$, $\texttt{UPDATE}$, $\texttt{SETUP-OK}$, $\texttt{UPDATE-OK}$, $\texttt{REVOKE}$ or $\texttt{CLOSE}$ message, then behave exactly as the functionality $\mathcal{F}_L(T,k)$.

$\boxed{\text{Pre-Create}}$

Upon $(\texttt{PRE-CREATE}, \gamma, \texttt{TX}_\texttt{f}, i, t_{ofl}) \xhookleftarrow{\tau_0} P$, let $\mathcal{S}$ define $T_1 \leq T$ and:

Both agreed: If already received $(\texttt{PRE-CREATE}, \gamma, \texttt{TX}_\texttt{f}, i, t_{ofl}) \xhookleftarrow{\tau} Q$, where $\tau_0 - \tau \leq T_1$, check that $\texttt{TX}_\texttt{f}.\mathsf{Output}[i].\mathsf{cash} = \gamma.\mathsf{cash}$. If yes, set $\Gamma_{pre}(\gamma.\mathsf{id}) := (\gamma, \texttt{TX}_\texttt{f}, t_{ofl})$ and $(\texttt{PRE-CREATED}, \gamma.\mathsf{id}) \xhookrightarrow{\tau_0} \gamma.\mathsf{endUsers}$. Else stop.
Wait for Q: Else store the message and stop.

$\boxed{\text{Pre-Update}}$

Upon $(\texttt{PRE-UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \xhookleftarrow{\tau_0} P$, let $\mathcal{S}$ define $T_1, T_2 \leq T$, parse $(\gamma, \texttt{TX}_\texttt{f}, t) := \Gamma_{pre}(id)$ and proceed as follows:

1) In round $\tau_1 \leq \tau_0 + T$, let $\mathcal{S}$ set $|\vec{tid}| = k$. Then $(\texttt{PRE-UPDATE-REQ}, id, \vec{\theta}, t_{\mathsf{stp}}, \vec{tid}) \xhookrightarrow{\tau_1} Q$ and $(\texttt{PRE-SETUP}, id, \vec{tid}) \xhookrightarrow{\tau_1} P$.
2) If $(\texttt{PRE-SETUP-OK}, id) \xhookleftarrow{\tau_2 \leq \tau_1 + t_{\mathsf{stp}}} P$, then $(\texttt{PRE-SETUP-OK}, id) \xhookrightarrow{\tau_2 + T_1} Q$. Else stop.
3) In round $\tau_2 + T_1$ distinguish:
   - If $(\texttt{PRE-UPDATE-OK}, id) \xhookleftarrow{\tau_2 + T_1} Q$, then $(\texttt{PRE-UPDATE-OK}, id) \xhookrightarrow{\tau_2 + 2T_1} P$.
   - If not and Q honest or if instructed by $\mathcal{S}$, $(\texttt{PRE-UPDATE-REJECT}, id) \xhookrightarrow{\tau_2 + 2T_1} P$.
   - Else execute $\texttt{Wait-if-Register}(id)$ and stop.
4) If $(\texttt{PRE-REVOKE}, id) \xhookleftarrow{\tau_2 + 2T_1} P$, $(\texttt{PRE-REVOKE-REQ}, id) \xhookrightarrow{\tau_2 + 2T_1 + T_2} Q$. Else execute $\texttt{Wait-if-Register}(id)$ and stop.
5) If $(\texttt{PRE-REVOKE}, id) \xhookleftarrow{\tau_2 + 2T_1 + T_2} Q$, set $\gamma.\mathsf{st} = \vec{\theta}$ and $\Gamma_V(id) := (\gamma, \texttt{TX}_\texttt{f})$. Then $(\texttt{PRE-UPDATED}, id, \vec{\theta}) \xhookrightarrow{\tau_2 + 2T_1 + 2T_2} \gamma.\mathsf{endUsers}$ and stop. Else $\texttt{Wait-if-Register}(id)$ and stop.

$\boxed{\text{Register – executed in every round}}$

Let t_0 be the current round. For every $(\gamma, \texttt{TX}_\texttt{f}) \in \Gamma_{pre}$ check if $\texttt{TX}_\texttt{f}$ appears on the ledger $\widehat{\mathcal{L}}$. If yes, then $\Gamma_{pre}(\gamma.id) = \bot$ and $\Gamma(\gamma.id) = (\gamma, \texttt{TX}_\texttt{f})$.

$\boxed{\text{Subprocedure } \texttt{Wait-if-Register}(id)}$

Let τ_0 be the current round and $(\gamma, \texttt{TX}_\texttt{f}, t_{ofl}) := \Gamma_{pre}(id)$.
1) Set $\Gamma_{pre}(id) := (\gamma, \texttt{TX}_\texttt{f}, t_{ofl}, \texttt{in-dispute})$.
2) Wait for t_{ofl} rounds. If after this time, $\Gamma_{pre}(id) \neq \bot$, then set $\vec{\theta}_{old} := \gamma.\mathsf{st}$, $\gamma.\mathsf{st} := \{\vec{\theta}_{old}, \vec{\theta}\}$ and $\Gamma_{pre}(id) := (\gamma, \texttt{TX}_\texttt{f}, t_{ofl}, \texttt{in-dispute})$.

C. Adaptor Signatures

Adaptor signatures have been introduced and used in the cryptocurrencies community for some time, but have been formalized for the fist time in [1]. These signatures not only allow for authentication as normal signatures schemes do, but also reveal a secret value upon publishing. Here we recall the definition of an adaptor signature scheme from [1]. In a nutshell, an adaptor signature is generated in two phases. First a pre-signature is computed w.r.t. some statement Y of a hard relation R e.g. $Y = g^y$ where g is the generator of the group $\mathbb{G}$ in which computing the discrete logarithm is hard. We define L_R to be the associated language for R defined as $L_R := \{Y \mid \exists y \text{ s.t. } (Y, y) \in R\}$. This pre-signature can be adapted to a full signature given a witness y for the statement Y, i.e. $(Y, y) \in R$. Furthermore, given the pre-signature and the adapted full signature one can extract a witness y. We now recall the definition for adaptor signature schemes from [1].

Definition 2 (Adaptor Signature Scheme): An adaptor signature scheme wrt. a hard relation R and a signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ consists of four algorithms $\Xi_{R,\Sigma} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ defined as:

$\mathsf{pSign}_{sk}(m, Y)$**:** is a PPT algorithm that on input a secret key sk, message $m \in \{0,1\}^*$ and statement $Y \in L_R$, outputs a pre-signature $\tilde{\sigma}$.

$\mathsf{pVrfy}_{pk}(m, Y; \tilde{\sigma})$**:** is a DPT algorithm that on input a public key pk, message $m \in \{0,1\}^*$, statement $Y \in L_R$ and pre-signature $\tilde{\sigma}$, outputs a bit b.

$\mathsf{Adapt}(\tilde{\sigma}, y)$**:** is a DPT algorithm that on input a pre-signature $\tilde{\sigma}$ and witness y, outputs a signature σ.

$\mathsf{Ext}(\sigma, \tilde{\sigma}, Y)$**:** is a DPT algorithm that on input a signature σ, pre-signature $\tilde{\sigma}$ and statement $Y \in L_R$, outputs a witness y such that $(Y, y) \in R$, or $\bot$.

We now briefly recall the properties that an adaptor signature scheme must satisfy and refer the reader to [1] for the formal definitions.

a) Correctness:: An adaptor signature should not only satisfy the standard signature correctness, but it must also satisfy *pre-signature correctness*. This property guarantees that if a pre-signature is generated honestly (wrt. a statement $Y \in L_R$), it can be *adapted* into a valid signature such that a witness for Y can be extracted.

b) Existential unforgeablity under chosen message attack for adaptor signatures:: Unforgeability for adaptor signatures is very similar to the normal definition of existential unforgeability under chosen message attacks for digital signatures, but it additionally requires that producing a forged signature σ for a message m is hard even if the adversary is given a pre-signature on the challenge message m w.r.t. a random statement $Y \in L_R$.

c) Pre-signature adaptability:: Intuitively it is required that any *valid* pre-signature w.r.t. Y (even when produced by a malicious signer) can be completed into a valid signature using the witness y where $(Y, y) \in R$.

d) Witness extractability:: In a nutshell, this property states that given a valid signature/pre-signatue pair $(\sigma, \tilde{\sigma})$ for a message m with respect to a statement Y, one can extract the corresponding witness y.

D. Realizing the wrapped functionality

While [1] presents a protocol Π_L that realizes the ideal functionality $\mathcal{F}_L$, it does not say anything about our wrapped functionality $\mathcal{F}_{preL}$. In order to have such protocol, we design a *protocol wrapper* around the protocol Π_L and prove that such wrapped protocol, which we denote Π_{preL} realizes the ideal functionality $\mathcal{F}_{preL}$.

Let us stress that the protocol wrapper very closely follows the protocol for ledger channel. Below we stress the main difference and thereafter we formally define the protocol for completeness.

a) Pre-Create: The only deference between the pre-create and create is that in pre-create $\texttt{TX}_\texttt{f}$ is neither generated by the parties nor posted on the ledger and is given as an input from the environment. Intuitively this is a funding transactions that might be posted in the future. Hence such channels are called pre-created or prepared channels.

b) Pre-Update: During the pre-update procedure, parties update the state of the pre-created channel as in normal ledger channels, but parties cannot directly force-close the channel since the funding transaction is not posted on the ledger yet. Hence in case of dispute parties first have to post this transaction on the ledger this is captured in calls to `Wait–if–Register` sub-procedure.

c) Register: This is a new procedure in order to capture the situation during which the funding transaction of a pre-created channel is posted on the ledger. In this case the pre-created channel is transformed into a normal ledger channel and is added to the list of ledger channels. Furthermore if this channel was in dispute, it is directly force closed.

To summarize, parties upon receiving one of the `PRE–UPDATE`, `PRE–SETUP–OK`, `PRE–UPDATE–OK` or `PRE–REVOKE` messages, behave as in the protocol Π_L with the following changes:

- Use the channel space Γ^P_{pre} instead of Γ^P.
- Add t_{ofl} rounds to the absolute time lock of new $\mathtt{TX_c}$.
- Replace calls to $\mathtt{L–ForceClose}^P$ by calls to $\mathtt{Wait–if–Register}^P$ which marks a channel to be in dispute.
- In case the reacting party peacefully rejects the update, output `PRE–UPDATE–REJECT` before you stop.
- When the protocol instructs you to output a m-message, where $m \in \{$`UPDATE–REQ`, `SETUP`, `SETUP–OK`, `UPDATE–OK`, `REVOKE–REQ`, `UPDATED`$\}$, then output PRE–m.

Wrapped Ledger Channel Protocol Π_{preL}

Below, we abbreviate $Q := \gamma.\mathsf{otherParty}(P)$ for $P \in \gamma.\mathsf{endUsers}$.

Ledger channels

Upon receiving a `CREATE`, `UPDATE`, `SETUP–OK`, `UPDATE–OK`, `REVOKE` or `CLOSE` message, then behave exactly as in the protocol Π_L.

Pre-Create

Party P upon $(\mathtt{PRE–CREATE}, \gamma, \mathtt{TX_f}, i, t_{ofl}) \xleftarrow{t_0} \mathcal{E}$:

1) If $\mathtt{TX_f}.\mathsf{Output}[i].\mathsf{cash} \neq \gamma.\mathsf{cash}$, then ignore the message.
2) Set $id := \gamma.\mathsf{id}$, generate $(R_P, r_P) \leftarrow \mathsf{GenR}$, $(Y_P, y_P) \leftarrow \mathsf{GenR}$ and send $(\text{createInfo}, id, \mathtt{TX_f}, i, t_{ofl}, R_P, Y_P) \xhookrightarrow{t_0} Q$.
3) If $(\text{createInfo}, id, \mathtt{TX_f}, i, t_{ofl}, R_Q, Y_Q) \xhookleftarrow{t_0+1} Q$, create:
$$[\mathtt{TX_c}] := \mathtt{GenCommit}([\mathtt{TX_f}], I_P, I_Q, 0)$$
$$[\mathtt{TX_s}] := \mathtt{GenSplit}([\mathtt{TX_c}].\mathsf{txid}\|1, \gamma.\mathsf{st})$$
for $I_P := (pk_P, R_P, Y_P)$, $I_Q := (pk_Q, R_Q, Y_Q)$. Else stop.
4) Compute $s^P_{\mathsf{c}} \leftarrow \mathsf{pSign}_{sk_P}([\mathtt{TX_c}], Y_Q)$, $s^P_{\mathsf{s}} \leftarrow \mathsf{Sign}_{sk_P}([\mathtt{TX_s}])$ and send $(\text{createCom}, id, s^P_{\mathsf{c}}, s^P_{\mathsf{s}}) \xhookrightarrow{t_0+1} Q$.
5) If $(\text{createCom}, id, s^Q_{\mathsf{c}}, s^Q_{\mathsf{s}}) \xhookleftarrow{t_0+2} Q$, s.t. $\mathsf{pVrfy}_{pk_Q}([\mathtt{TX_c}], Y_P; s^Q_{\mathsf{c}}) = 1$ and $\mathsf{Vrfy}_{pk_Q}([\mathtt{TX_s}]; s^Q_{\mathsf{s}}) = 1$, set
$$\mathtt{TX_c} := ([\mathtt{TX_c}], \{\mathsf{Sign}_{sk_P}([\mathtt{TX_c}]), \mathsf{Adapt}(s^Q_{\mathsf{c}}, y_P)\})$$
$$\mathtt{TX_s} := ([\mathtt{TX_s}], \{s^P_{\mathsf{s}}, s^Q_{\mathsf{s}}\})$$
$$\Gamma^P_{pre}(\gamma.\mathsf{id}) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_P, R_Q, Y_Q, s^P_{\mathsf{c}}), \mathtt{TX_s}, t_{ofl}).$$
and send $(\mathtt{PRE–CREATED}, id) \xhookrightarrow{t_0+2} \mathcal{E}$.

Pre-Update

Party P upon $(\mathtt{PRE–UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \xhookleftarrow{t^P_0} \mathcal{E}$

1) Generate $(R_P, r_P) \leftarrow \mathsf{GenR}$, $(Y_P, y_P) \leftarrow \mathsf{GenR}$ and send the message $(\text{updateReq}, id, \vec{\theta}, t_{\mathsf{stp}}, R_P, Y_P) \xhookrightarrow{t^P_0} Q$.

Party Q upon $(\text{updateReq}, id, \vec{\theta}, t_{\mathsf{stp}}, R_P, Y_P) \xhookleftarrow{t^Q_0} P$

2) Generate $(R_Q, r_Q) \leftarrow \mathsf{GenR}$ and $(Y_Q, y_Q) \leftarrow \mathsf{GenR}$.
3) Extract $\mathtt{TX_f}$ and t_{ofl} from $\Gamma^P_{pre}(id)$.
4) Set $t_{\mathsf{lock}} := t^Q_0 + t_{\mathsf{stp}} + 4 + \Delta + t_{ofl}$ and
$$[\mathtt{TX_c}] := \mathtt{GenCommit}([\mathtt{TX_f}], I_P, I_Q, t_{\mathsf{lock}})$$
$$[\mathtt{TX_s}] := \mathtt{GenSplit}([\mathtt{TX_c}].\mathsf{txid}\|1, \vec{\theta})$$
where $I_P := (pk_P, R_P, Y_P)$, $I_Q := (pk_Q, R_Q, Y_Q)$.
5) Sign $s^Q_{\mathsf{s}} \leftarrow \mathsf{Sign}_{sk_Q}([\mathtt{TX_s}])$, send $(\text{updateInfo}, id, R_Q, Y_Q, s^Q_{\mathsf{s}}) \xhookrightarrow{t^Q_0} P$, $(\mathtt{PRE–UPDATE–REQ}, id, \vec{\theta}, t_{\mathsf{stp}}, \mathtt{TX_s}.\mathsf{txid}) \xhookrightarrow{t^Q_0+1} \mathcal{E}$.

Party P upon $(\text{updateInfo}, id, h_Q, Y_Q, s^Q_{\mathsf{s}}) \xhookleftarrow{t^P_0+2} Q$

6) Extract $\mathtt{TX_f}$ and t_{ofl} from $\Gamma^Q_{pre}(id)$.
7) Set $t_{\mathsf{lock}} := t^P_0 + t_{\mathsf{stp}} + 5 + \Delta + t_{ofl}$, and
$$[\mathtt{TX_c}] := \mathtt{GenCommit}([\mathtt{TX_f}], I_P, I_Q, t_{\mathsf{lock}})$$
$$[\mathtt{TX_s}] := \mathtt{GenSplit}([\mathtt{TX_c}].\mathsf{txid}\|1, \vec{\theta}),$$
for $I_P := (pk_P, R_P, Y_P)$ and $I_Q := (pk_Q, R_Q, Y_Q)$. If it holds that $\mathsf{Vrfy}_{pk_Q}([\mathtt{TX_s}]; s^Q_{\mathsf{s}}) = 1$, $(\mathtt{PRE–SETUP}, id, \mathtt{TX_s}.\mathsf{txid}) \xhookrightarrow{t^P_0+2} \mathcal{E}$. Else stop.
8) If $(\mathtt{PRE–SETUP–OK}, id) \xhookleftarrow{t^P_1 \leq t^P_0+2+t_{\mathsf{stp}}} \mathcal{E}$, compute the values $s^P_{\mathsf{c}} \leftarrow \mathsf{pSign}_{sk_P}([\mathtt{TX_c}], Y_Q)$, $s^P_{\mathsf{s}} \leftarrow \mathsf{Sign}_{sk_P}([\mathtt{TX_s}])$ and send the message $(\text{updateComP}, id, s^P_{\mathsf{c}}, s^P_{\mathsf{s}}) \xhookrightarrow{t^P_1} Q$. Else stop.

Party Q

9) If $(\text{updateComP}, id, s^P_{\mathsf{c}}, s^P_{\mathsf{s}}) \xhookleftarrow{t^Q_1 \leq t^Q_0+2+t_{\mathsf{stp}}} P$, s.t. $\mathsf{pVrfy}_{pk_P}([\mathtt{TX_c}], Y_Q; s^P_{\mathsf{c}}) = 1$ and $\mathsf{Vrfy}_{pk_P}([\mathtt{TX_s}]; s^P_{\mathsf{s}}) = 1$, output $(\mathtt{PRE–SETUP–OK}, id) \xhookrightarrow{t^Q_1} \mathcal{E}$. Else stop.
10) If $(\mathtt{PRE–UPDATE–OK}, id) \xhookleftarrow{t^Q_1} \mathcal{E}$, pre-sign $s^Q_{\mathsf{c}} \leftarrow \mathsf{pSign}([\mathtt{TX_c}], Y_P)$ and send $(\text{updateComQ}, id, s^Q_{\mathsf{c}}) \xhookrightarrow{t^Q_1} P$. Else send the message $(\text{updateNotOk}, id, r_Q) \xhookrightarrow{t^Q_1} P$ and stop.

Party P

11) In round $t^P_1 + 2$ distinguish the following cases:
    - If $(\text{updateComQ}, id, s^Q_{\mathsf{c}}) \xhookleftarrow{t^P_1+2} Q$, s.t. $\mathsf{pVrfy}_{pk_Q}([\mathtt{TX_c}], Y_P; s^Q_{\mathsf{c}}) = 1$, output $(\mathtt{PRE–UPDATE–OK}, id) \xhookrightarrow{t^P_1+2} \mathcal{E}$.

- If $(\text{updateNotOk}, id, r_Q) \xleftarrow{t_1^P+2} Q$, s.t. $(R_Q, r_Q) \in R$, add $\Theta^P(id) := \Theta^P(id) \cup ([\mathtt{TX_c}], r_Q, Y_Q, s_c^P)$, output the message $(\mathtt{PRE\text{–}UPDATE\text{–}REJECT}) \xhookrightarrow{t_1^P+2} \mathcal{E}$ and stop.
- Else, execute the procedure $\mathtt{Wait\text{–}if\text{–}Register}^P(id)$ and stop.

12) If $(\mathtt{PRE\text{–}REVOKE}, id) \xleftarrow{t_1^P+2} \mathcal{E}$, parse $\Gamma^P_{pre}(id)$ as $(\gamma, \mathtt{TX_f}, (\overline{\mathtt{TX}}_\mathtt{c}, \bar{r}_P, \bar{R}_Q, \bar{Y}_Q, \bar{s}^P_{\mathtt{Com}}), \overline{\mathtt{TX}}_\mathtt{s})$ and update the channel space as $\Gamma^P_{pre}(id) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_P, R_Q, Y_Q, s_c^P), \mathtt{TX_s})$, for $\mathtt{TX_s} := ([\mathtt{TX_s}], \{s_s^P, s_s^Q\})$ and $\mathtt{TX_c} := ([\mathtt{TX_c}], \{\mathsf{Sign}_{sk_P}([\mathtt{TX_c}]), \mathsf{Adapt}(s_c^Q, y_P)\})$., and send $(\text{revokeP}, id, \bar{r}_P) \xhookrightarrow{t_1^P+2} Q$. Else, execute $\mathtt{Wait\text{–}if\text{–}Register}^P(id)$ and stop.

$\boxed{\text{Party } Q}$

13) Parse $\Gamma^Q_{pre}(id)$ as $(\gamma, \mathtt{TX_f}, (\overline{\mathtt{TX}}_\mathtt{c}, \bar{r}_Q, \bar{R}_P, \bar{Y}_P, \bar{s}^Q_{\mathtt{Com}}), \overline{\mathtt{TX}}_\mathtt{s})$. If $(\text{revokeP}, id, \bar{r}_P) \xleftarrow{t_1^Q+2} P$, s.t. $(\bar{R}_P, \bar{r}_P) \in R$, $(\mathtt{PRE\text{–}REVOKE\text{–}REQ}, id) \xhookrightarrow{t_1^Q+2} \mathcal{E}$. Else execute $\mathtt{Wait\text{–}if\text{–}Register}^Q(id)$ and stop.
14) If $(\mathtt{PRE\text{–}REVOKE}, id) \xleftarrow{t_1^Q+2} \mathcal{E}$ as a reply, set

$$\Theta^Q(id) := \Theta^Q(id) \cup ([\overline{\mathtt{TX}}_\mathtt{c}], \bar{r}_P, \bar{Y}_P, \bar{s}^Q_{\mathtt{Com}})$$
$$\Gamma^Q_{pre}(id) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_Q, R_P, Y_P, s_c^Q), \mathtt{TX_s}),$$

for $\mathtt{TX_s} := ([\mathtt{TX_s}], \{s_s^P, s_s^Q\})$, $\mathtt{TX_c} := ([\mathtt{TX_c}], \{\mathsf{Sign}_{sk_Q}([\mathtt{TX_c}]), \mathsf{Adapt}(s_c^P, y_Q)\})$, and send $(\text{revokeQ}, id, \bar{r}_Q) \xhookrightarrow{t_1^Q+2} P$. In the next round $(\mathtt{PRE\text{–}UPDATED}, id) \xhookrightarrow{t_1^Q+3} \mathcal{E}$ and stop. Else, in round $t_1^Q + 2$, execute $\mathtt{Wait\text{–}if\text{–}Register}^Q(id)$ and stop.

$\boxed{\text{Party } P}$

15) If $(\text{revokeQ}, id, \bar{r}_Q) \xleftarrow{t_1^P+4} Q$ s.t. $(\bar{R}_Q, \bar{r}_Q) \in R$, then set $\Theta^P(id) := \Theta^P(id) \cup ([\overline{\mathtt{TX}}_\mathtt{c}], \bar{r}_Q, \bar{Y}_Q, \bar{s}^P_{\mathtt{Com}})$ and $(\mathtt{PRE\text{–}UPDATED}, id) \xhookrightarrow{t_1^P+4} \mathcal{E}$. Else execute $\mathtt{Wait\text{–}if\text{–}Register}^P(id)$ and stop.

Register

Party P in every round t_0: For each $id \in \{0,1\}^*$ s.t. $\Gamma^P_{pre}(id) \neq \perp$:
1) Parse $\Gamma^P_{pre}(id) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_P, R_Q, Y_Q, s_c^P), \mathtt{TX_s}, t_{ofl}, x)$
2) If $\mathtt{TX_f}$ appeared on-chain in this round, then
   a) Set $\Gamma(id) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_P, R_Q, Y_Q, s_c^P), \mathtt{TX_s})$.
   b) Set $\Gamma^P_{pre}(id) := \perp$
   c) If $x = \mathtt{in\text{–}dispute}$, then call $\mathtt{L\text{–}ForceClose}^P(id)$.

Subprocedures

$\mathtt{GenCommit}([\mathtt{TX_f}], (pk_P, R_P, Y_P), (pk_Q, R_Q, Y_Q), t)$:

Let $(c, \mathtt{Multi\text{–}Sig}_{pk_P, pk_Q}) := \mathtt{TX_f}.\mathsf{Output}[1]$ and denote

$$\varphi_1 := \mathtt{Multi\text{–}Sig}_{\mathsf{ToKey}(R_Q), \mathsf{ToKey}(Y_Q), pk_P},$$
$$\varphi_2 := \mathtt{Multi\text{–}Sig}_{\mathsf{ToKey}(R_P), \mathsf{ToKey}(Y_P), pk_Q},$$
$$\varphi_3 := \mathtt{CheckRelative}_\Delta \wedge \mathtt{Multi\text{–}Sig}_{pk_P, pk_Q}.$$

Return [tx], where $\mathsf{tx.Input} = \mathtt{TX_f}.\mathsf{txid}\|1$, $\mathsf{tx.Output} := (c, \varphi_1 \vee \varphi_2 \vee \varphi_3)$ and set $\mathsf{tx.TimeLock}$ to t if $t > \mathsf{now}$ and to 0 otherwise.

$\mathtt{GenSplit}(tid, \vec{\theta})$:
Return [tx], where $\mathsf{tx.Input} := tid$ and $\mathsf{tx.Output} := \vec{\theta}$.

$\mathtt{Wait\text{–}if\text{–}Register}^P(id)$:
Let t_0 be the current round. Let $X := \Gamma^P_{pre}(id)$. Then set $\Gamma^P_{pre}(id) := (X, \mathtt{in\text{–}dispute})$.

Theorem 1: Let Σ be a signature scheme that is existentially unforgeable against chosen message attacks, R a hard relation and $\Xi_{R,\Sigma}$ a secure adaptor signature scheme. Then for any ledger delay $\Delta \in \mathbb{N}$, the protocol Π_{preL} UC-realizes the ideal functionality $\mathcal{F}_{preL}(3, 1)$.

II. Additional material for Virtual channels

We now formally describe the protocol $\Pi_V(T)$ that was discussed on a high level in **??**. Since our goal it to prove that $\Pi_V(T)$ UC-realizes $\mathcal{F}_V(T)$, we need to discuss about parties deal with instruction about ledger channel as well as virtual channel.

A. Ledger Channels

As a first step, we discuss how parties deal with message about ledger channel or prepared ledger channel. On a high level, parties simply forward these instructions to the hybrid ideal functionality $\mathcal{F}_{preL}(T, 1)$. If the functionality sends a reply, the party forwards this reply to the environment. In addition to the message forwarding, parties stores information about the ledger channels in a channel space Γ_L. More precisely, once the a ledger channel is created or pre-created, party adds this channels to Γ_L. Once an existing ledger channel is updated or pre-updated, the party updates the latest state of the channel stored in Γ_L.

There is one technicality that we need to take care of. There are two different situations in which a party of a virtual channel protocol instructs the hybrid ideal functionality $\mathcal{F}_{preL}(T, 1)$ to pre-cerate (resp. pre-update) a channel γ:
1) Party receives a pre-create, resp. pre-update, instruction from the environment. As discussed above, in this case the party acts as a dummy party and forward the message to $\mathcal{F}_{preL}(T, 1)$.
2) Party is creating, resp. updating, a virtual channel and hence is sending pre-create, resp. pre-update, messages to $\mathcal{F}_{preL}(T, 1)$.

Let us stress that while channels pre-created via option (1) exist in both the real and ideal world, channels pre-created via option (2) exist only in the real world. This is because the pre-creation of these channels was not initiated by the environment but by the parties of the virtual channel protocol. Hence, we need to make sure that the environment cannot "accidentally" update a channel pre-created via (2) since this would help the environment distinguish between the real and ideal world.

To this end, party in the case (1) modifies the identifier of the channel by adding a prefix "ledger". More precisely, if the environment makes a request about a channel with identifier id it forwards the instruction to the hybrid functionality but replaces id with ledger$\|id$. Analogously, if the hybrid functionality replies to this message, the party removes the prefix. This ensure that the environment cannot directly make any change on the ledger channels pre-created via option (2).

B. Create

The creation of a virtual channel was described on a high level in **??**. The main idea is to update the two subchannels of the virtual channel and pre-create a new ledger channel corresponding to the virtual channel. Importantly, the update of the subchannel needs to be synchonized in order to ensure that either both updates complete (in which case the virtual channel is created) or both updates are rejected (in which case the virtual channel creation fails).

Since large part of the creation process is the same for channel with and without validity, our formal description is modularized.

Create a virtual channels - modular

Below we abbreviate $\mathcal{F}_{preL} := \mathcal{F}_{preL}(T, 1)$, $A := \gamma.\mathsf{Alice}$, $B := \gamma.\mathsf{Bob}$, $I = \gamma.\mathsf{Ingrid}$. For $P \in \gamma.\mathsf{endUsers}$, we denote $Q := \gamma.\mathsf{otherParty}(P)$.

$\boxed{\text{Party } P \in \{A, B\}}$

Upon receiving $(\mathtt{CREATE}, \gamma) \overset{t_0^P}{\hookleftarrow} \mathcal{E}$ proceed as follows:

1) Let $id_\alpha := \gamma.\mathsf{subchan}(P)$ and compute
$$\theta_P := \mathtt{GenVChannelOutput}(\gamma, P).$$
2) Send $(\mathtt{UPDATE}, id_\alpha, \theta_P, t_{\mathsf{stp}}) \overset{t_0^P}{\hookrightarrow} \mathcal{F}_{preL}$.
3) Upon receiving $(\mathtt{SETUP}, id_\alpha, tid_P) \overset{t_1^P \le t_0^P + T}{\hookleftarrow} \mathcal{F}_{preL}$, engage in the subprotocol $\mathtt{SetupVChannel}$ with input (γ, tid_P).

$\boxed{\text{Party } I}$

Upon receiving $(\mathtt{CREATE}, \gamma) \overset{t_0^I}{\hookleftarrow} \mathcal{E}$ proceed as follows:

1) Set $id_\alpha = \gamma.\mathsf{subchan}(A), id_\beta = \gamma.\mathsf{subchan}(B)$ and generate
$$\theta_A := \mathtt{GenVChannelOutput}(\gamma, A)$$
$$\theta_B := \mathtt{GenVChannelOutput}(\gamma, B)$$
2) If in round $t_1^I \le t_0^I + T$ you have received both $(\mathtt{UPDATE\text{–}REQ}, id_\alpha, \theta_A, t_{\mathsf{stp}}, tid_A) \hookleftarrow \mathcal{F}_{preL}$ and $(\mathtt{UPDATE\text{–}REQ}, id_\beta, \theta_B, t_{\mathsf{stp}}, tid_B) \hookleftarrow \mathcal{F}_{preL}$, then engage in the subprotocol $\mathtt{SetupVChannel}$ with inputs (γ, tid_A, tid_B). Else stop.

$\boxed{\text{Party } P \in \{A, B\}}$

Wait until $t_2^P := t_1^P + t_{\mathsf{stp}}$. If the subprotocol completed successfully, then send $(\mathtt{SETUP\text{–}OK}, id_\alpha) \overset{t_2^P}{\hookrightarrow} \mathcal{F}_{preL}$. Else stop.

$\boxed{\text{Party } I}$

If in round $t_2^I \le t_1^I + t_{\mathsf{stp}} + T$ you receive both $(\mathtt{SETUP\text{–}OK}, id_\alpha)) \hookleftarrow \mathcal{F}_{preL}$ and $(\mathtt{SETUP\text{–}OK}, id_\beta)) \hookleftarrow \mathcal{F}_{preL}$, send $(\mathtt{UPDATE\text{–}OK}, id_\alpha) \overset{t_2}{\hookrightarrow} \mathcal{F}_{preL}$ and $(\mathtt{UPDATE\text{–}OK}, id_\beta) \overset{t_2}{\hookrightarrow} \mathcal{F}_{preL}$. Otherwise stop.

$\boxed{\text{Party } P \in \{A, B\}}$

1) If you receive $(\mathtt{UPDATE\text{–}OK}, id_\alpha) \overset{t_2^P \le t_1^P + 2T}{\hookleftarrow} \mathcal{F}_{preL}$, reply with $(\mathtt{REVOKE}, id_\alpha) \overset{t_2^P + T}{\hookrightarrow} \mathcal{F}_{preL}$. Otherwise stop.

$\boxed{\text{Party } I}$

If in round $t_3^I \le t_2^I + 4T$ you have received both $(\mathtt{REVOKE\text{–}REQ}, id_\alpha) \hookleftarrow \mathcal{F}_{preL}$ and $(\mathtt{REVOKE\text{–}REQ}, id_\beta) \hookleftarrow \mathcal{F}_{preL}$, reply $(\mathtt{REVOKE}, id_\alpha) \overset{t_3^I}{\hookrightarrow} \mathcal{F}_{preL}$ and $(\mathtt{REVOKE}, id_\beta) \overset{t_3^I}{\hookrightarrow} \mathcal{F}_{preL}$ and update $\Gamma^I(\gamma.\mathsf{id})$ from $(\bot, x)$ to (γ, x). Otherwise stop.

$\boxed{\text{Party } P \in \{A, B\}}$

Upon receiving $(\mathtt{UPDATED}, id_\alpha) \overset{t_3^P \le t_2^P + 3T}{\hookleftarrow} \mathcal{F}_{preL}$, mark γ as created, i.e. update $\Gamma^P(\gamma.\mathsf{id})$ from $(\bot, x)$ to (γ, x), and output $(\mathtt{CREATED}, \gamma.\mathsf{id}) \overset{t_3^P}{\hookrightarrow} \mathcal{E}$.

Function $\mathtt{GenVChannelOutput}(\gamma, P)$

Return θ, where $\theta.\mathsf{cash} = \gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2$ and $\theta.\varphi$ is defined as follows

$$\theta.\varphi = \begin{cases} \mathtt{Multi\text{–}Sig}_{\gamma.\mathsf{users}} \vee (\mathtt{One\text{–}Sig}_P \wedge \mathtt{CheckRelative}_{(T+4\Delta)}), \\ \qquad \text{if } \gamma.\mathsf{val} = \bot \\ \mathtt{Multi\text{–}Sig}_{A,I} \vee (\mathtt{One\text{–}Sig}_I \wedge \mathtt{CheckLockTime}_{\gamma.\mathsf{val}}), \\ \qquad \text{if } \gamma.\mathsf{val} \neq \bot \wedge P = A \\ \mathtt{Multi\text{–}Sig}_{B,I} \vee (\mathtt{One\text{–}Sig}_B \wedge \mathtt{CheckLockTime}_{\gamma.\mathsf{val}+2\Delta}), \\ \qquad \text{if } \gamma.\mathsf{val} \neq \bot \wedge P = B \end{cases}$$

Subprotocol $\mathtt{SetupVChannel}$

Let t_0 be the current round.

Channels without validity

$\boxed{\text{Party } P \in \{A, B\} \text{ on input } (\gamma, tid_P)}$

1) Create the body of the funding transactions:
$$\mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{Input} := (tid_P, tid_Q)$$
$$\mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{Output} := ((\gamma.\mathsf{cash}, \mathtt{Multi\text{–}Sig}_{\{\gamma.\mathsf{endUsers}\}}), (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}, \mathtt{One\text{–}Sig}_{pk_I}))$$
2) Send $(\mathtt{PRE\text{–}CREATE}, \gamma, \mathtt{TX}_{\mathtt{f}}, 1, t_{ofl}) \overset{t_0}{\hookrightarrow} \mathcal{F}_{preL}$, where $t_{ofl} = 2T + 8\Delta$.

3) If $(\texttt{PRE–CREATED}, \gamma.\mathsf{id}) \xleftarrow{t_1 \leq t_0+T} \mathcal{F}_{preL}$, then sign the funding transaction, i.e. $s_{\mathtt{f}}^P \leftarrow \mathsf{Sign}_{sk_P}([\mathtt{TX}_{\mathtt{f}}^{\gamma}])$ and send $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{f}}^P, [\mathtt{TX}_{\mathtt{f}}^{\gamma}]) \overset{t_1}{\hookrightarrow} I$. Else stop.

$\boxed{\text{Party } I \text{ on input } (\gamma, tid_A, tid_B)}$

4) If you receive $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{f}}^A, [\mathtt{TX}_{\mathtt{f}}^{\gamma}]) \xleftarrow{t_2 \leq t_0+T+1} A$ and $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{f}}^B, [\mathtt{TX}_{\mathtt{f}}^{\gamma}]) \overset{t_2}{\hookleftarrow} B$, verify the funding transaction and signatures of A and B, i.e. check:

$$\mathsf{Vrfy}_{pk_A}([\mathtt{TX}_{\mathtt{f}}^{\gamma}]; s_{\mathtt{f}}^A) = 1$$
$$\mathsf{Vrfy}_{pk_B}([\mathtt{TX}_{\mathtt{f}}^{\gamma}], s_{\mathtt{f}}^B) = 1$$
$$(tid_A, tid_B) = \mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{Input}$$
$$(\gamma.\mathsf{cash} + \gamma.\mathsf{fee}, \mathtt{One–Sig}_{pk_I}) \in \mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{Output}.$$

5) If all checks pass, sign the funding transaction, i.e. compute

$$s_{\mathtt{f}}^I := \mathsf{Sign}_{sk_I}([\mathtt{TX}_{\mathtt{f}}^{\gamma}]),$$
$$\mathtt{TX}_{\mathtt{f}}^{\gamma} := \{([\mathtt{TX}_{\mathtt{f}}^{\gamma}], s_{\mathtt{f}}^A, s_{\mathtt{f}}^B, s_{\mathtt{f}}^I)\}.$$

Store $\Gamma^I(\gamma.\mathsf{id}) := (\bot, \mathtt{TX}_{\mathtt{f}}^{\gamma})$. Then send $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{f}}^B, s_{\mathtt{f}}^I) \overset{t_2}{\hookrightarrow} A$ and $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{f}}^A, s_{\mathtt{f}}^I) \overset{t_2}{\hookrightarrow} B$, and consider procedure successfully completed. Else stop.

$\boxed{\text{Party } P \in \{A, B\}}$

6) Upon receiving $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{f}}^Q, s_{\mathtt{f}}^I) \xleftarrow{t_1+1} I$, verify all signatures, i.e. check:

$$\mathsf{Vrfy}_{pk_Q}([\mathtt{TX}_{\mathtt{f}}^{\gamma}]; s_{\mathtt{f}}^Q) = 1$$
$$\mathsf{Vrfy}_{pk_I}([\mathtt{TX}_{\mathtt{f}}^{\gamma}], s_{\mathtt{f}}^I) = 1.$$

If all checks pass define $\mathtt{TX}_{\mathtt{f}}^{\gamma} := \{([\mathtt{TX}_{\mathtt{f}}^{\gamma}], s_{\mathtt{f}}^P, s_{\mathtt{f}}^Q, s_{\mathtt{f}}^I)\}$ and set $\Gamma^P(\gamma.\mathsf{id}) := (\bot, \mathtt{TX}_{\mathtt{f}}^{\gamma}, tid_P)$ and consider procedure successfully completed. Else stop.

Channels with validity

$\boxed{\text{Party } A \text{ on input } (\gamma, tid_A)}$

1) Send $(\text{createInfo}, \gamma.\mathsf{id}, tid_A) \overset{t_0}{\hookrightarrow} B$
2) In round $t_1 := t_0 + 1$, create the body of the funding transaction:

$$\mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{Input} := (tid_A)$$
$$\mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{Output} := ((\gamma.\mathsf{cash}, \mathtt{Multi–Sig}_{\{\gamma.\mathsf{endUsers}\}}), (\gamma.\mathsf{fee}/2, \mathtt{One–Sig}_{pk_I}))$$

3) Send $(\texttt{PRE–CREATE}, \gamma, \mathtt{TX}_{\mathtt{f}}, 1, t_{ofl}) \overset{t_1}{\hookrightarrow} \mathcal{F}_{preL}$, for $t_{ofl} = \gamma.\mathsf{val} + 3\Delta$.
4) If $(\texttt{PRE–CREATED}, \gamma.\mathsf{id}) \xleftarrow{t_2 \leq t_1+T} \mathcal{F}_{preL}$, then goto step (10). Else stop.

$\boxed{\text{Party } B \text{ on input } (\gamma, tid_B)}$

5) If $(\text{createInfo}, \gamma.\mathsf{id}, tid_A) \xleftarrow{t_1 := t_0+1} A$, then create the body of the funding and refund transactions:

$$\mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{Input} := (tid_A)$$
$$\mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{Output} := ((\gamma.\mathsf{cash}, \mathtt{Multi–Sig}_{\{\gamma.\mathsf{endUsers}\}}), (\gamma.\mathsf{fee}/2, \mathtt{One–Sig}_{pk_I}))$$
$$\mathtt{TX}_{\mathtt{refund}}^{\gamma}.\mathsf{Input} := (\mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{txid}||2, tid_B)$$
$$\mathtt{TX}_{\mathtt{refund}}^{\gamma}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}, \mathtt{One–Sig}_{pk_I}).$$

Else stop.
6) Send $(\texttt{PRE–CREATE}, \gamma, \mathtt{TX}_{\mathtt{f}}, 1, t_{ofl}) \overset{t_1}{\hookrightarrow} \mathcal{F}_{preL}$, for $t_{ofl} = \gamma.\mathsf{val} + 3\Delta$.
7) If $(\texttt{PRE–CREATED}, \gamma.\mathsf{id}) \xleftarrow{t_2 \leq t_1+T} \mathcal{F}_{preL}$, then compute a signature on the refund transaction, i.e., $s_{\mathtt{Ref}}^B \leftarrow \mathsf{Sign}_{sk_B}([\mathtt{TX}_{\mathtt{refund}}^{\gamma}])$ and define $\Gamma^B(\gamma.id) := (\bot, [\mathtt{TX}_{\mathtt{f}}^{\gamma}], tid_B)$. Then, send $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{Ref}}^B, [\mathtt{TX}_{\mathtt{refund}}^{\gamma}], [\mathtt{TX}_{\mathtt{f}}^{\gamma}]) \overset{t_2}{\hookrightarrow} I$ and consider procedure successfully completed. Else stop.

$\boxed{\text{Party } I \text{ on input } (\gamma, tid_A, tid_B)}$

8) If $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{Ref}}^B, [\mathtt{TX}_{\mathtt{refund}}^{\gamma}], [\mathtt{TX}_{\mathtt{f}}^{\gamma}]) \xleftarrow{t_3 \leq t_0+T+2} B$, verify the fund and refund transactions and signature of B, i.e. check:

$$\mathsf{Vrfy}_{sk_B}([\mathtt{TX}_{\mathtt{refund}}^{\gamma}]; s_{\mathtt{Ref}}^B) = 1.$$
$$[\mathtt{TX}_{\mathtt{refund}}^{\gamma}].\mathsf{Input} = (\mathtt{TX}_{\mathtt{f}}^{\gamma}.\mathsf{txid}||2, tid_B),$$
$$[\mathtt{TX}_{\mathtt{refund}}^{\gamma}].\mathsf{Output} = (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}, \mathtt{One–Sig}_{pk_I}),$$
$$[\mathtt{TX}_{\mathtt{f}}^{\gamma}].\mathsf{Output}[2] = (\gamma.\mathsf{fee}/2, \mathtt{One–Sig}_{pk_I})$$

If all checks pass, then sign the fund and refund transactions, i.e. compute

$$s_{\mathtt{Ref}}^I := \mathsf{Sign}_{sk_I}([\mathtt{TX}_{\mathtt{refund}}^{\gamma}]), s_{\mathtt{f}}^I := \mathsf{Sign}_{sk_I}([\mathtt{TX}_{\mathtt{f}}^{\gamma}]),$$
$$\mathtt{TX}_{\mathtt{refund}}^{\gamma} := \{([\mathtt{TX}_{\mathtt{refund}}^{\gamma}], s_{\mathtt{Ref}}^I, s_{\mathtt{Ref}}^B)\}.$$

Else stop.
9) Store $\Gamma^I(\gamma.\mathsf{id}) := (\bot, [\mathtt{TX}_{\mathtt{f}}^{\gamma}], \mathtt{TX}_{\mathtt{refund}}^{\gamma}, tid_A, tid_B)$, send the message $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{f}}^I) \overset{t_3}{\hookrightarrow} A$, and consider procedure successfully completed.

$\boxed{\text{Party } A}$

10) If you receive $(\text{createFund}, \gamma.\mathsf{id}, s_{\mathtt{f}}^I) \xleftarrow{t_2+2} I$, verify the signature, i.e. check $\mathsf{Vrfy}_{pk_I}([\mathtt{TX}_{\mathtt{f}}^{\gamma}]; s_{\mathtt{f}}^I) = 1$. If the check passes, compute a signature on the fund transaction:

$$s_{\mathtt{f}}^A := \mathsf{Sign}_{sk_A}([\mathtt{TX}_{\mathtt{f}}^{\gamma}]),$$
$$\mathtt{TX}_{\mathtt{f}}^{\gamma,\mathtt{A}} := \{([\mathtt{TX}_{\mathtt{f}}^{\gamma}], s_{\mathtt{f}}^I, s_{\mathtt{f}}^A)\}.$$

and set $\Gamma^A(\gamma.\mathsf{id}) := (\bot, \mathtt{TX}_{\mathtt{f}}^{\gamma,\mathtt{A}}, tid_A)$. Then consider procedure successfully completed. Else stop.

C. Update

As discussed in **??**, in order to update a virtual channel, parties update the corresponding prepared channel. This is does in a black-box way via the hybrid functionality $\mathcal{F}_{preL}$. Hence, parties act as dummy parties as forward update instructions (modified by adding PRE–) to the hybrid functionality $\mathcal{F}_{preL}$ and forward the replies of the functionality (modified by removing PRE–) to the environment. In case the update fails, party offload the channel which allows to resolve disputes on-chain.

Update

Below we abbreviate $\mathcal{F}_{preL} := \mathcal{F}_{preL}(T, 1)$.

Initiating party P:

1) Upon $(\texttt{UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \xleftarrow{t_0} \mathcal{E}$, $(\texttt{PRE–UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \overset{t_0}{\hookrightarrow} \mathcal{F}_{preL}$.
2) If $(\texttt{PRE–SETUP}, id, tid_P) \xhookleftarrow{t_1 \leq t_0+T} \mathcal{F}_{preL}$, $(\texttt{SETUP}, id, tid_P) \overset{t_1}{\hookrightarrow} \mathcal{E}$. Else stop.
3) If $(\texttt{SETUP–OK}, id) \xhookleftarrow{t_2 \leq t_1+t_{\mathsf{stp}}} \mathcal{E}$, $(\texttt{PRE–SETUP–OK}, id) \overset{t_2}{\hookrightarrow} \mathcal{E}$. Else stop.
4) Distinguish the following three cases:
   - If $(\texttt{PRE–UPDATE–OK}, id) \xhookleftarrow{t_3 \leq t_2+T} \mathcal{F}_{preL}$, $(\texttt{UPDATE–OK}, id) \overset{t_3}{\hookrightarrow} \mathcal{E}$.
   - If $(\texttt{PRE–UPDATE–REJECT}, id) \xhookleftarrow{t_3 \leq t_2+T} \mathcal{F}_{preL}$, then stop.
   - Else execute the procedure $\texttt{Offload}^P(id)$ and stop.
5) If $(\texttt{REVOKE}, id) \xhookleftarrow{t_3} \mathcal{E}$, $(\texttt{PRE–REVOKE}, id) \overset{t_3}{\hookrightarrow} \mathcal{F}_{preL}$. Else execute $\texttt{Offload}^P(id)$ and stop.
6) If $(\texttt{PRE–UPDATED}, id) \xhookleftarrow{t_4 \leq t_3+T} \mathcal{F}_{preL}$, update the channel space, i.e., let $\gamma := \Gamma^P(id)$, set $\gamma.\mathsf{st} := \vec{\theta}$ and $\Gamma(id) := \gamma$. Then $(\texttt{UPDATED}, id) \overset{t_4}{\hookrightarrow} \mathcal{F}_{preL}$. Else execute $\texttt{Offload}^P(id)$ and stop.

Reacting party Q

1) Upon $(\texttt{PRE–UPDATE–REQ}, id, \vec{\theta}, t_{\mathsf{stp}}, tid) \xhookleftarrow{\tau_0} \mathcal{F}_{preL}$, $(\texttt{UPDATE–REQ}, id, \vec{\theta}, t_{\mathsf{stp}}, tid) \overset{\tau_0}{\hookrightarrow} \mathcal{E}$.
2) If $(\texttt{PRE–SETUP–OK}, id) \xhookleftarrow{\tau_1 \leq \tau_0+t_{\mathsf{stp}}+T} \mathcal{F}_{preL}$, $(\texttt{SETUP–OK}, id) \overset{\tau_1}{\hookrightarrow} \mathcal{E}$. Else stop.
3) If $(\texttt{UPDATE–OK}, id) \xhookleftarrow{\tau_1} \mathcal{E}$, $(\texttt{PRE–UPDATE–OK}, id) \overset{\tau_1}{\hookrightarrow} \mathcal{F}_{preL}$. Else stop.
4) If $(\texttt{PRE–REVOKE–REQ}, id) \xhookleftarrow{\tau_2 \leq \tau_1+T} \mathcal{F}_{preL}$, $(\texttt{REVOKE–REQ}, id) \overset{\tau_2}{\hookrightarrow} \mathcal{E}$. Else execute $\texttt{Offload}^Q(id)$ and stop.
5) If $(\texttt{REVOKE}, id) \xhookleftarrow{\tau_2} \mathcal{E}$, $(\texttt{PRE–REVOKE}, id) \overset{\tau_2}{\hookrightarrow} \mathcal{F}_{preL}$. Else execute $\texttt{Offload}^Q(id)$ and stop.
6) Upon $(\texttt{PRE–UPDATED}, id) \xhookleftarrow{\tau_3 \leq \tau_2+T} \mathcal{F}_{preL}$, update the channel space, i.e., let $\gamma := \Gamma^Q(id)$, set $\gamma.\mathsf{st} := \vec{\theta}$ and $\Gamma(id) := \gamma$. Then $(\texttt{UPDATED}, id) \overset{\tau_3}{\hookrightarrow} \mathcal{E}$.

D. Offload

As a next step, we define the offloading process which transforms a virtual channel into a ledger channel. Let us stress that offloading can be triggered either by the environment via a message $\texttt{OFFLOAD}$ or internally by parties when executing an update or close. To avoid code repetition, we define a procedure $\texttt{Offload}^P(id)$ and instruct parties upon receiving $(\texttt{OFFLOAD}, id) \xhookleftarrow{t_0} \mathcal{E}$ to simply call $\texttt{Offload}^P(id)$.

Since channels with validity are constructed in a different way than channel without validity, the procedure is defined for the two cases separately.

Subprocedure $\texttt{Offload}^P(id)$

Below we abbreviate $\mathcal{F}_{preL} := \mathcal{F}_{preL}(T, 1)$, $A := \gamma.\mathsf{Alice}$ and $B := \gamma.\mathsf{Bob}$ and $I = \gamma.\mathsf{Ingrid}$. For $P \in \gamma.\mathsf{endUsers}$, we denote $Q := \gamma.\mathsf{otherParty}(P)$. Let t_0 be the current round.

Channels without validity

$P \in \{A, B\}$

1) Extract γ and $\texttt{TX}_{\mathtt{f}}^{\gamma}$ from $\Gamma^P(id)$ and tid_P, tid_Q from $\texttt{TX}_{\mathtt{f}}^{\gamma}$. Then define $id_\alpha := \gamma.\mathsf{subchan}(P)$ and send $(\texttt{CLOSE}, id_\alpha) \overset{t_0}{\hookrightarrow} \mathcal{F}_{preL}$.
2) If you receive $(\texttt{CLOSED}, id_\alpha) \xhookleftarrow{t_1 \leq t_0+T+3\Delta} \mathcal{F}_{preL}$, then continue. Else set $\Gamma^P(\gamma.\mathsf{id}) = \perp$ and stop.
3) Let $T_2 := t_1 + T + 3\Delta$ and distinguish:
   - If in round $t_2 \leq T_2$ a transaction with tid_Q appeared on $\widehat{\mathcal{L}}$, then $(\mathrm{post}, \texttt{TX}_{\mathtt{f}}^{\gamma}) \overset{t_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
   - Else in round T_2 create the punishment transaction $\texttt{TX}_{\mathsf{pun}}$ as $\texttt{TX}_{\mathsf{pun}}.\mathsf{Input} := tid_P$, $\texttt{TX}_{\mathsf{pun}}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \texttt{One–Sig}_{pk_P})$ and $\texttt{TX}_{\mathsf{pun}}.\mathsf{Witness} := \mathsf{Sign}_{sk_P}([\texttt{TX}_{\mathsf{pun}}])$. Then $(\mathrm{post}, \texttt{TX}_{\mathsf{pun}}) \overset{T_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
4) Let $T_3 := t_2 + \Delta$ and distinguish the following two cases:
   - The transaction $\texttt{TX}_{\mathtt{f}}^{\gamma}$ was accepted by $\widehat{\mathcal{L}}$ in $t_3 \leq T_3$, then update $\Gamma_L^P(id) := \Gamma^P(id)$ and set $m :=$ offloaded.
   - The transaction $\texttt{TX}_{\mathsf{pun}}$ was accepted by $\widehat{\mathcal{L}}$ in $t_3 \leq T_3$, then set $m :=$ punished.
5) Set $\Gamma^P(id) = \perp$ and return m in round t_3.

Party I

1) Extract γ and $\texttt{TX}_{\mathtt{f}}^{\gamma}$ from $\Gamma^I(id)$ and tid_A, tid_B from $\texttt{TX}_{\mathtt{f}}^{\gamma}$. Then define $id_\alpha := \gamma.\mathsf{subchan}(A)$, $id_\beta := \gamma.\mathsf{subchan}(B)$ and send the messages $(\texttt{CLOSE}, id_\alpha) \overset{t_0}{\hookrightarrow} \mathcal{F}_{preL}$ and $(\texttt{CLOSE}, id_\beta) \overset{t_0}{\hookrightarrow} \mathcal{F}_{preL}$.
2) If you receive both messages $(\texttt{CLOSED}, id_\alpha) \xhookleftarrow{t_1^A \leq t_0+T+3\Delta} \mathcal{F}_{preL}$ and $(\texttt{CLOSED}, id_\beta) \xhookleftarrow{t_1^B \leq t_0+T+3\Delta} \mathcal{F}_{preL}$, publish $(\mathrm{post}, \texttt{TX}_{\mathtt{f}}^{\gamma}) \overset{t_1}{\hookrightarrow} \widehat{\mathcal{L}}$, where $t_1 := \max\{t_1^A, t_1^B\}$. Otherwise set $\Gamma^I(id) = \perp$ and stop.
3) Once $\texttt{TX}_{\mathtt{f}}^{\gamma}$ is accepted by $\widehat{\mathcal{L}}$ in round $t_2 \leq t_1 + \Delta$, then $\Gamma^I(id) = \perp$ and return "offloaded".

Channels with validity

Party A

1) Extract γ, tid_A and $\texttt{TX}_{\mathtt{f}}^{\gamma}$ from $\Gamma^A(id)$. Let $id_\alpha := \gamma.\mathsf{subchan}(A)$ and send $(\texttt{CLOSE}, id_\alpha) \overset{t_0}{\hookrightarrow} \mathcal{F}_{preL}$.
2) If you receive $(\texttt{CLOSED}, id_\alpha) \xhookleftarrow{t_1 \leq t_0+T+3\Delta} \mathcal{F}_{preL}$, then post $(\mathrm{post}, \texttt{TX}_{\mathtt{f}}^{\gamma,A}) \overset{t_2}{\hookrightarrow} \widehat{\mathcal{L}}$. Otherwise, set $\Gamma^A(\gamma.\mathsf{id}) = \perp$ and stop.
3) Once $\texttt{TX}_{\mathtt{f}}^{\gamma}$ is accepted by $\widehat{\mathcal{L}}$ in round $t_2 \leq t_1 + \Delta$, then update $\Gamma_L^A(id) := \Gamma^A(id)$, $\Gamma^A(id) := \perp$ and return "offloaded".

Party B

1) Extract γ, tid_B and $[\texttt{TX}_{\mathtt{f}}^{\gamma}]$ from $\Gamma^B(id)$. Let $id_\beta := \gamma.\mathsf{subchan}(B)$ and send $(\texttt{CLOSE}, id_\beta) \overset{t_0}{\hookrightarrow} \mathcal{F}_{preL}$.

2) If you receive $(\texttt{CLOSED}, id_\beta) \overset{t_1 \leq t_0+T+3\Delta}{\hookleftarrow} \mathcal{F}_{preL}$, then continue. Otherwise, set $\Gamma^B(\gamma.\mathsf{id}) = \bot$ and stop.
3) Create the punishment transaction $\texttt{TX}_{\text{pun}}$ as $\texttt{TX}_{\text{pun}}.\mathsf{Input} := tid_B$, $\texttt{TX}_{\text{pun}}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \texttt{One–Sig}_{pk_B})$ and set the value $\texttt{TX}_{\text{pun}}.\mathsf{Witness} := \mathsf{Sign}_{sk_B}([\texttt{TX}_{\text{pun}}])$. Then wait until round $t_2 := \max\{t_1, \gamma.\mathsf{val} + 2\Delta\}$ and send $(\text{post}, \texttt{TX}_{\text{pun}}) \overset{t_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
4) Let $T_3 := t_2 + \Delta$ and distinguish the following two cases:
   - A transaction with identifier $\texttt{TX}^\gamma_{\texttt{f}}.\mathsf{txid}$ was accepted by $\widehat{\mathcal{L}}$ in $t_3 \leq T_3$, then define $\Gamma^B_L(id) := \Gamma^B(id)$ and set $m :=$ offloaded.
   - The transaction $\texttt{TX}_{\text{pun}}$ was accepted by $\widehat{\mathcal{L}}$ in $t_3 \leq T_3$, set $m :=$ punished.
5) Set $\Gamma^B(id) := \bot$ and return m in round t_3.

Party I

1) Extract γ, tid_A, tid_B, $\texttt{TX}^\gamma_{\texttt{refund}}$ and $[\texttt{TX}^\gamma_{\texttt{f}}]$ from $\Gamma^I(id)$. Then define $id_\alpha := \gamma.\mathsf{subchan}(A)$, $id_\beta := \gamma.\mathsf{subchan}(B)$ and send $(\texttt{CLOSE}, id_\alpha) \overset{t_0}{\hookrightarrow} \mathcal{F}_{preL}$ and $(\texttt{CLOSE}, id_\beta) \overset{t_0}{\hookrightarrow} \mathcal{F}_{preL}$.
2) If you receive both messages $(\texttt{CLOSED}, id_\alpha) \overset{t_1^A \leq t_0+T+3\Delta}{\hookleftarrow} \mathcal{F}_{preL}$ and $(\texttt{CLOSED}, id_\beta) \overset{t_1^B \leq t_0+T+3\Delta}{\hookleftarrow} \mathcal{F}_{preL}$, then continue. Otherwise, set $\Gamma^I(\gamma.\mathsf{id}) = \bot$ and stop.
3) Create the punishment transaction $\texttt{TX}_{\text{pun}}$ as $\texttt{TX}_{\text{pun}}.\mathsf{Input} := tid_A$, $\texttt{TX}_{\text{pun}}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \texttt{One–Sig}_{pk_I})$ and set the value $\texttt{TX}_{\text{pun}}.\mathsf{Witness} := \mathsf{Sign}_{sk_I}([\texttt{TX}_{\text{pun}}])$. Then wait until round $t_2 := \max\{t_1^A, \gamma.\mathsf{val}\}$ and send $(\text{post}, \texttt{TX}_{\text{pun}}) \overset{t_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
4) Let $T_3 := t_2 + \Delta$ and distinguish the following two cases:
   - A transaction with identifier $\texttt{TX}^\gamma_{\texttt{f}}.\mathsf{txid}$ was accepted by $\widehat{\mathcal{L}}$ in $t_3' \leq T_3$, send $(\text{post}, \texttt{TX}^\gamma_{\texttt{refund}}) \overset{t_4}{\hookrightarrow} \widehat{\mathcal{L}}$ where $t_4 := \max\{t_1^B, t_3'\}$. Once $\texttt{TX}^\gamma_{\texttt{refund}}$ is accepted by $\widehat{\mathcal{L}}$ in round $t_5 \leq t_4 + \Delta$, then define $m :=$ offloaded and $\Gamma^I(\gamma.\mathsf{id}) = \bot$.
   - The transaction $\texttt{TX}_{\text{pun}}$ was accepted by $\widehat{\mathcal{L}}$ in $t_3'' \leq T_3$, then define $m :=$ punished and $\Gamma^I(\gamma.\mathsf{id}) = \bot$.
5) Return m in round t_6 where $t_6 := \max\{t_5, t_3''\}$.

E. Close

In order to close a virtual channel, parties first try to adjust the balances in the sunchannel according to the latest valid state of the virtual channel. This is done by updating the subchannels in a synchonous way, as was done during virtual channel creation. In case this process fails, parties close the channel forcefully. This means that parties first offload the channel and then immediately close the offloaded ledger channel.

Close a virtual channel

Below we abbreviate $\mathcal{F}_{preL} := \mathcal{F}_{preL}(T, 1)$, $A := \gamma.\mathsf{Alice}$ and $B := \gamma.\mathsf{Bob}$ and $I = \gamma.\mathsf{Ingrid}$. For $P \in \gamma.\mathsf{endUsers}$, we denote $Q := \gamma.\mathsf{otherParty}(P)$.

Party $P \in \{A, B\}$

Upon receiving $(\texttt{CLOSE}, id) \overset{t_0^P}{\hookleftarrow} \mathcal{E}$ or in round $t_0^P := \gamma.\mathsf{val} - (4\Delta + 7T)$ if $\gamma.\mathsf{val} \neq \bot$, proceed as follows:
1) Extract γ, $\texttt{TX}^\gamma_{\texttt{f}}$ from $\Gamma^P(id)$.
2) Parse $\gamma.\mathsf{st} = \Big((c_P, \texttt{One–Sig}_{pk_P}), (c_Q, \texttt{One–Sig}_{pk_Q})\Big)$.
3) Compute the new state of the channel $id_\alpha := \gamma.\mathsf{subchan}(P)$ as
$$\vec{\theta}_P := \{(c_P, \texttt{One–Sig}_{pk_P}), (c_Q + \frac{\gamma.\mathsf{fee}}{2}, \texttt{One–Sig}_{pk_I})\}$$
Then, send $(\texttt{UPDATE}, id_\alpha, \vec{\theta}_P, 0) \overset{t_0^P}{\hookrightarrow} \mathcal{F}_{preL}$.
4) Upon $(\texttt{SETUP}, id_\alpha, tid_P) \overset{t_1^P \leq t_0^P + T}{\hookleftarrow} \mathcal{F}_{preL}$, send $(\texttt{SETUP–OK}, id_\alpha) \overset{t_1^P}{\hookrightarrow} \mathcal{F}_{preL}$.

Party I

Upon receiving $(\texttt{CLOSE}, id) \overset{t_0^I}{\hookleftarrow} \mathcal{E}$ or in round $t_0^I := \gamma.\mathsf{val} - (4\Delta + 7T)$, proceed as follows:
1) Extract γ, $\texttt{TX}^\gamma_{\texttt{f}}$ from $\Gamma^I(id)$.
2) Let $id_\alpha = \gamma.\mathsf{subchan}(A)$, $id_\beta = \gamma.\mathsf{subchan}(B)$ and $c := \gamma.\mathsf{cash}$
3) If in round $t_1^I \leq t_0^I + T$ you received both $(\texttt{UPDATE–REQ}, id_\alpha, tid_A, \vec{\theta}_A, 0) \hookleftarrow \mathcal{F}_{preL}$ and $(\texttt{UPDATE–REQ}, id_\beta, tid_B, \vec{\theta}_B, 0) \hookleftarrow \mathcal{F}_{preL}$ check that for some c_A, c_B s.t. $c_A + c_B = c$ it holds
$$\vec{\theta}_A = \{(c_A, \texttt{One–Sig}_{pk_A}), (c_B + \gamma.\mathsf{fee}/2, \texttt{One–Sig}_{pk_I})\}$$
$$\vec{\theta}_B = \{(c_B, \texttt{One–Sig}_{pk_B}), (c_A + \gamma.\mathsf{fee}/2, \texttt{One–Sig}_{pk_I})\}$$
If not, then stop.
4) If in round $t_2^I \leq t_1^I + T$ you receive both $(\texttt{SETUP–OK}, id_\alpha))$ $\hookleftarrow \mathcal{F}_{preL}$ and $(\texttt{SETUP–OK}, id_\beta)) \hookleftarrow \mathcal{F}_{preL}$, send $(\texttt{UPDATE–OK}, id_\alpha) \overset{t_2^I}{\hookrightarrow} \mathcal{F}_{preL}$ and $(\texttt{UPDATE–OK}, id_\beta) \overset{t_2^I}{\hookrightarrow} \mathcal{F}_{preL}$. If not, then stop.

Party $P \in \{A, B\}$

If you receive $(\texttt{UPDATE–OK}, id_\alpha) \overset{t_2^P \leq t_1^P + 2T}{\hookleftarrow} \mathcal{F}_{preL}$, reply with $(\texttt{REVOKE}, id_\alpha) \overset{t_2^P}{\hookrightarrow} \mathcal{F}_{preL}$. Otherwise execute $\texttt{Offload}^P(id)$ and stop.

Party I

If in round $t_3^I \leq t_2^I + 2T$ you received both $(\texttt{REVOKE–REQ}, id_\alpha)$ $\hookleftarrow \mathcal{F}_{preL}$ and $(\texttt{REVOKE–REQ}, id_\beta) \hookleftarrow \mathcal{F}_{preL}$, reply $(\texttt{REVOKE}, id_\alpha) \overset{t_3^I}{\hookrightarrow} \mathcal{F}_{preL}$ and $(\texttt{REVOKE}, id_\beta) \overset{t_3^I}{\hookrightarrow} \mathcal{F}_{preL}$ and set $\Gamma^I(id) := \bot$.

Party $P \in \{A, B\}$

If you receive $(\texttt{UPDATED}, id_\alpha) \overset{t_3^P \leq t_2^P + 2T}{\hookleftarrow} \mathcal{F}_{preL}$, set $\Gamma^P(id) := \bot$. Then output $(\texttt{CLOSED}, id) \overset{t_3^P}{\hookrightarrow} \mathcal{E}$ and stop. Else execute $\texttt{Offload}^P(id)$ and stop.

F. Punish

Finally, we formalize the actions taken by parties in every round. On a high level, in addition to triggering the hybrid ideal functionality to take the every-round actions for ledger channel (which include blockchain monitoring for outdated commit transactions), parties also need to make several check for virtual channel. Namely, channel users that tried to offload the virtual channel by closing their subchannel) monitor whether the other subchannel was closed as well. If yes, then they can publish the funding transaction and complete the offload and otherwise apply the punishment mechanism.

Punish virtual channel

Below we abbreviate $\mathcal{F}_{preL} := \mathcal{F}_{preL}(T, 1)$, $A := \gamma.\mathsf{Alice}$ and $B := \gamma.\mathsf{Bob}$ and $I = \gamma.\mathsf{Ingrid}$. For $P \in \gamma.\mathsf{endUsers}$, we denote $Q := \gamma.\mathsf{otherParty}(P)$.

Upon receiving $(\mathtt{PUNISH}) \xleftarrow{\tau_0} \mathcal{E}$, do the following:

- Forward this message to the hybrid ideal functionality $(\mathtt{PUNISH}) \xrightarrow{\tau_0} \mathcal{F}_{preL}$. If $(\mathtt{PUNISHED}, id) \xleftarrow{\tau_1} \mathcal{F}_{preL}$, then $(\mathtt{PUNISHED}, id) \xrightarrow{\tau_1} \mathcal{E}$.
- Execute both subprotocols `Punish` and `Punish–Validity`.

<u>`Punish`</u>

$\boxed{\text{Party } P \in \{A, B\}}$

For every $id \in \{0,1\}^*$, such that γ which $\gamma.\mathsf{val} = \perp$ can be extracted from $\Gamma^P(id)$ do the following:

1) Extract $\mathtt{TX}_\mathtt{f}^\gamma$ from $\Gamma^P(id)$ and tid_P, tid_Q from $\mathtt{TX}_\mathtt{f}^\gamma$. Check if tid_P appeared on $\widehat{\mathcal{L}}$. If not, then stop.
2) Denote $T_2 := t_1 + T + 3\Delta$ and distinguish:
   - If in round $t_2 \leq T_2$ the transaction with tid_Q appeared on $\widehat{\mathcal{L}}$, then $(\text{post}, \mathtt{TX}_\mathtt{f}^\gamma) \xrightarrow{t_2} \widehat{\mathcal{L}}$.
   - Else in round T_2 create the punishment transaction $\mathtt{TX}_\mathsf{pun}$ as
   $$\mathtt{TX}_\mathsf{pun}.\mathsf{Input} := tid_P$$
   $$\mathtt{TX}_\mathsf{pun}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \mathtt{One\text{–}Sig}_{pk_P})$$
   $$\mathtt{TX}_\mathsf{pun}.\mathsf{Witness} := \mathsf{Sign}_{sk_P}([\mathtt{TX}_\mathsf{pun}]),$$
   and $(\text{post}, \mathtt{TX}_\mathsf{pun}) \xrightarrow{T_2} \widehat{\mathcal{L}}$.
3) Let $T_3 := t_2 + \Delta$ and distinguish the following two cases:
   - The transaction $\mathtt{TX}_\mathtt{f}^\gamma$ was accepted by $\widehat{\mathcal{L}}$ in $t_3 \leq T_3$, then $\Gamma_L^P(id) := \Gamma^P(id)$, $\Gamma^P(id) = \perp$ and $m := \mathtt{OFFLOADED}$.
   - The transaction $\mathtt{TX}_\mathsf{pun}$ was accepted by $\widehat{\mathcal{L}}$ in $t_3 \leq T_3$, then define $\Gamma^P(\gamma.\mathsf{id}) = \perp$ and set $m := \mathtt{PUNISHED}$.
4) Output $(m, id) \xrightarrow{t_3} \mathcal{E}$.

$\boxed{\text{Party } I}$

For every $id \in \{0,1\}^*$, such that γ with $\gamma.\mathsf{val} = \perp$ can be extracted from $\Gamma^I(id)$ do the following:

1) Extract $\mathtt{TX}_\mathtt{f}^\gamma$ from $\Gamma^I(id)$ and tid_A, tid_B from $\mathtt{TX}_\mathtt{f}^\gamma$. Check if for some $P \in \{A, B\}$ a transaction with identifier tid_P appeared on $\widehat{\mathcal{L}}$. If not, then stop.
2) Denote $id_\alpha := \gamma.\mathsf{subchan}(Q)$ and send $(\mathtt{CLOSE}, id_\alpha) \xrightarrow{t_0} \mathcal{F}_{preL}$.
3) If you receive $(\mathtt{CLOSED}, id_\alpha) \xleftarrow{t_1 \leq t_0 + T + 3\Delta} \mathcal{F}_{preL}$ and tid_Q appeared on $\widehat{\mathcal{L}}$, $(\text{post}, \mathtt{TX}_\mathtt{f}^\gamma) \xrightarrow{t_1} \widehat{\mathcal{L}}$. Otherwise set $\Gamma^I(id) = \perp$ and stop.
4) Once $\mathtt{TX}_\mathtt{f}^\gamma$ is accepted by $\widehat{\mathcal{L}}$ in round t_2, such that $t_2 \leq t_1 + \Delta$, set $\Gamma^I(id) = \perp$ and output $(\mathtt{OFFLOADED}, id) \xrightarrow{t_2} \mathcal{E}$.

<u>`Punish–Validity`</u>

$\boxed{\text{Party } A}$

For every $id \in \{0,1\}^*$, such that γ with $\gamma.\mathsf{val} \neq \perp$ can be extracted from $\Gamma^A(id)$ do the following:

1) Extract $\mathtt{TX}_\mathtt{f}^\gamma$ from $\Gamma^A(id)$ and tid_A from $\mathtt{TX}_\mathtt{f}^\gamma$. If tid_A appeared on $\widehat{\mathcal{L}}$, then send $(\text{post}, \mathtt{TX}_\mathtt{f}^\gamma) \xrightarrow{t_1} \widehat{\mathcal{L}}$. Else stop.
2) Once $\mathtt{TX}_\mathtt{f}^\gamma$ is accepted by $\widehat{\mathcal{L}}$ in round $t_2 \leq t_1 + \Delta$, set $\Gamma_L^A(id) := \Gamma^A(id)$, $\Gamma^A(id) := \perp$ and output $(\mathtt{OFFLOADED}, id) \xrightarrow{t_2} \mathcal{E}$.

$\boxed{\text{Party } B}$

For every $id \in \{0,1\}^*$, such that γ which $\gamma.\mathsf{val} = \perp$ can be extracted from $\Gamma^B(id)$ do the following:

1) Extract tid_B and $[\mathtt{TX}_\mathtt{f}^\gamma]$ from $\Gamma^B(id)$. Check if tid_B or $[\mathtt{TX}_\mathtt{f}^\gamma].\mathsf{txid}$ appeared on $\widehat{\mathcal{L}}$. If not, then stop.
2) If a transaction $\mathtt{TX}_\mathtt{f}^\gamma$ appeared on $\widehat{\mathcal{L}}$, update set $\Gamma_L^B(id) := \Gamma^B(id)$ and $\Gamma^B(id) := \perp$. Then output $(\mathtt{OFFLOADED}, id) \xrightarrow{t_1} \mathcal{E}$ and stop.
3) If tid_B appeared on $\widehat{\mathcal{L}}$, create the punishment transaction $\mathtt{TX}_\mathsf{pun}$ as
$$\mathtt{TX}_\mathsf{pun}.\mathsf{Input} := tid_B$$
$$\mathtt{TX}_\mathsf{pun}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \mathtt{One\text{–}Sig}_{pk_B})$$
$$\mathtt{TX}_\mathsf{pun}.\mathsf{Witness} := \mathsf{Sign}_{sk_B}([\mathtt{TX}_\mathsf{pun}]).$$
Then wait until round $t_2 := \max\{t_1, \gamma.\mathsf{val} + 2\Delta\}$ and send $(\text{post}, \mathtt{TX}_\mathsf{pun}) \xrightarrow{t_2} \widehat{\mathcal{L}}$.
4) If transaction $\mathtt{TX}_\mathsf{pun}$ was accepted by $\widehat{\mathcal{L}}$ in $t_3 \leq t_2 + \Delta$, then define $\Gamma^B(\gamma.\mathsf{id}) = \perp$ and output $(\mathtt{PUNISHED}, id) \xrightarrow{t_3} \mathcal{E}$.

$\boxed{\text{Party } I}$

For every $id \in \{0,1\}^*$, such that γ which $\gamma.\mathsf{val} = \perp$ can be extracted from $\Gamma^I(id)$ do the following:

1) Extract tid_A, tid_B, $\mathtt{TX}_\mathtt{refund}^\gamma$ and $[\mathtt{TX}_\mathtt{f}^\gamma]$ from $\Gamma^I(id)$. Check if tid_A or tid_B appeared on $\widehat{\mathcal{L}}$ or $t_1 = \gamma.\mathsf{val} - (3\Delta + T)$. If not, then stop.
2) Distinguish the following cases:
   - If $t_1 = \gamma.\mathsf{val} - (3\Delta + T)$, define $id_\alpha := \gamma.\mathsf{subchan}(A)$, $id_\beta := \gamma.\mathsf{subchan}(B)$ and send $(\mathtt{CLOSE}, id_\alpha) \xrightarrow{t_1} \mathcal{F}_{preL}$ and $(\mathtt{CLOSE}, id_\beta) \xrightarrow{t_1} \mathcal{F}_{preL}$.
   - If tid_B appeared on $\widehat{\mathcal{L}}$, send $(\mathtt{CLOSE}, id_\alpha) \xrightarrow{t_1} \mathcal{F}_{preL}$.
3) If a transaction with identifier tid_A appeared on $\widehat{\mathcal{L}}$ in round $t_2 \leq t_1 + T + 3\Delta$, create the punishment transaction $\mathtt{TX}_\mathsf{pun}$ as
$$\mathtt{TX}_\mathsf{pun}.\mathsf{Input} := tid_A$$
$$\mathtt{TX}_\mathsf{pun}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \mathtt{One\text{–}Sig}_{pk_I})$$
$$\mathtt{TX}_\mathsf{pun}.\mathsf{Witness} := \mathsf{Sign}_{sk_I}([\mathtt{TX}_\mathsf{pun}]).$$
Then wait until round $t_3 := \max\{t_2, \gamma.\mathsf{val}\}$ and send $(\text{post}, \mathtt{TX}_\mathsf{pun}) \xrightarrow{t_3} \widehat{\mathcal{L}}$.
4) Distinguish the following two cases:
   - The transaction $\mathtt{TX}_\mathtt{f}^\gamma.\mathsf{txid}$ was accepted by $\widehat{\mathcal{L}}$ in $t_4 \leq t_3 + \Delta$, send $(\text{post}, \mathtt{TX}_\mathtt{refund}^\gamma) \xrightarrow{t_5} \widehat{\mathcal{L}}$ where $t_5 := \max\{\gamma.\mathsf{val} + \Delta, t_4\}$. Once $\mathtt{TX}_\mathtt{refund}^\gamma$ is accepted by $\widehat{\mathcal{L}}$ in round $t_6 \leq t_5 + \Delta$, set $\Gamma^I(\gamma.\mathsf{id}) = \perp$ and output $(\mathtt{OFFLOADED}, id) \xrightarrow{t_6} \mathcal{E}$ and stop.

- The transaction $\mathtt{TX}_{\mathsf{pun}}$ was accepted by $\widehat{\mathcal{L}}$ in $t_4 \leq t_3 + \Delta$, then set $\Gamma^I(\gamma.\mathsf{id}) = \perp$ and output $(\mathtt{PUNISHED}, id) \overset{t_4}{\hookrightarrow} \mathcal{E}$.

III. Simplifying functionality description

In order to simplify the exposition, the formal descriptions of the channel ideal functionalities $\mathcal{F}_L$, $\mathcal{F}_{preL}$ and $\mathcal{F}_V$ are simplified. Namely, they exclude several natural checks that one would expect an ideal functionality to make when it receives a message from a party. The purpose of the checks is to avoid the functionality from accepting malformed messages. To provide some intuition, we present several examples of such restrictions:

- A party sends a malformed message (e.g. missing or additional parameters)
- A party request creation of a virtual channel but one of the two subchannels does not exists or does not have enough funds for virtual channel creation.
- Parties try to update the same channel twice in parallel.

We now list all check formally in the wrapper below which can be seen as an extension to the wrapper provided by [1] for $\mathcal{F}_L$.

Functionality wrapper: $\mathcal{W}_{\text{checks}}(T)$

The wrapper is defined for $\mathcal{F} \in \{\mathcal{F}_V(T), \mathcal{F}_{preL}(T), \mathcal{F}_L(T)\}$. Below, we abbreviate $A := \gamma.\mathsf{Alice}$, $B := \gamma.\mathsf{Bob}$ and $I := \gamma.\mathsf{Ingrid}$.

Create: Upon $(\mathtt{CREATE}, \gamma, tid) \overset{\tau_0}{\hookleftarrow} P$, where $P \in \gamma.\mathsf{users}$, check if: $\Gamma(\gamma.\mathsf{id}) = \perp$, $\mathcal{F}.\Gamma_{pre}(\gamma.\mathsf{id}) = \perp$ and there is no channel γ' with $\gamma.\mathsf{id} = \gamma'.\mathsf{id}$ being created or pre-created; γ is valid according to the definition given in **??**; $\gamma.\mathsf{st} = \{(c_P, \mathtt{One\text{-}Sig}_{pk_P}), (c_Q, \mathtt{One\text{-}Sig}_{pk_Q})\}$ for $c_P, c_Q \in \mathbb{R}^{\geq 0}$. Depending on the type of channel, make the following additional checks:

ledger channel: There exists $(t, id, i, \theta) \in \widehat{\mathcal{L}}.\mathtt{UTXO}$ such that $\theta = (c_P, \mathtt{One\text{-}Sig}_P)$ for $(id, i) := tid$;[a]

virtual channel:

- If $P \in \gamma.\mathsf{endUsers}$, then $\alpha := \mathcal{F}.\Gamma(id_P) \neq \perp$ for $id_P := \gamma.\mathsf{subchan}(P)$; $\alpha.\mathsf{endUsers} = \{P, I\}$; there is no other virtual channel being created over α and α is currently not being updated; both P and I have enough funds in α.
- If $P = I$, then $\alpha := \mathcal{F}.\Gamma(id_A) \neq \perp$ for $id_A := \gamma.\mathsf{subchan}(A)$; $\beta := \mathcal{F}.\Gamma(id_B) \neq \perp$ for $id_B := \gamma.\mathsf{subchan}(B)$; $\alpha.\mathsf{endUsers} = \{A, I\}$; $\beta.\mathsf{endUsers} = \{B, I\}$; there is no other virtual channel being created over α or β; A and I have enough funds in α and B and I have enough funds in β.
- If $\gamma.\mathsf{val} \neq \perp$, then $\gamma.\mathsf{val} \geq \tau_0 + 4\Delta + 15T$.

If one of the above checks fails, drop the message. Else proceed as $\mathcal{F}$.

Pre-Create: Upon $(\mathtt{PRE\text{-}CREATE}, \gamma, \mathtt{TX_f}, i, t_{ofl}) \overset{\tau_0}{\hookleftarrow} P$, check if: $P \in \gamma.\mathsf{users}$, $\mathcal{F}.\Gamma_{pre}(\gamma.\mathsf{id}) = \perp$, $\mathcal{F}.\Gamma_{pre}(\gamma.\mathsf{id}) = \perp$ and there is no channel γ' with $\gamma.\mathsf{id} = \gamma'.\mathsf{id}$ being created or pre-created; γ is valid according to the definition given in **??**; $\gamma.\mathsf{st} = \{(c_P, \mathtt{One\text{-}Sig}_{pk_P}), (c_Q, \mathtt{One\text{-}Sig}_{pk_Q})\}$ for $c_P, c_Q \in \mathbb{R}^{\geq 0}$ and $\mathtt{TX_f}$ is not a published transaction on $\widehat{\mathcal{L}}$. If one of the above checks fails, drop the message. Else proceed as $\mathcal{F}$.

(Pre)-Update: Upon $(m, id, \vec{\theta}, t_{\mathsf{stp}}) \overset{\tau_0}{\hookleftarrow} P$, check if: $\gamma := \Gamma(id) \neq \perp$ if $m = \mathtt{UPDATE}$ and $\gamma := \Gamma_{pre}(id) \neq \perp$ if $m = \mathtt{PRE\text{-}UPDATE}$. In both cases additionally check: $P \in \gamma.\mathsf{endUsers}$; there is no other update being preformed on γ; let $\vec{\theta} = (\theta_1, \dots \theta_\ell) = ((c_1, \varphi_1), \dots, (c_\ell, \varphi_\ell))$, then $\sum_{j \in [\ell]} c_i = \gamma.\mathsf{cash}$ and $\varphi_j \in \widehat{\mathcal{L}}.\mathcal{V}$ for each $j \in [\ell]$. If not, drop the message. Else proceed as $\mathcal{F}$.

Upon $((\mathtt{PRE\text{-}})\mathtt{SETUP\text{-}OK}, id) \overset{\tau_2}{\hookleftarrow} P$ check if: you accepted a message $((\mathtt{PRE\text{-}})\mathtt{UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \overset{\tau_0}{\hookleftarrow} P$, where $t_2 - t_0 \leq t_{\mathsf{stp}} + T$ and the message is a reply to the message $((\mathtt{PRE\text{-}})\mathtt{SETUP}, id, tid)$ sent to P in round τ_1 such that $\tau_2 - \tau_1 \leq t_{\mathsf{stp}}$[b]. If not, drop the message. Else proceed as $\mathcal{F}$.

Upon $((\mathtt{PRE\text{-}})\mathtt{UPDATE\text{-}OK}, id) \overset{\tau_0}{\hookleftarrow} P$, check if the message is a reply to the message $((\mathtt{PRE\text{-}})\mathtt{SETUP\text{-}OK}, id)$ sent to P in round τ_0. If not, drop the message. Else proceed as $\mathcal{F}$.

Upon $((\mathtt{PRE\text{-}})\mathtt{REVOKE}, id) \overset{\tau_0}{\hookleftarrow} P$, check if the message is a reply to either the message $((\mathtt{PRE\text{-}})\mathtt{UPDATE\text{-}OK}, id)$ sent to P in round τ_0 or the message $((\mathtt{PRE\text{-}})\mathtt{REVOKE\text{-}REQ}, id)$ sent to P in round τ_0. If not, drop the message. Else proceed as $\mathcal{F}$.

Offload: Upon receiving $(\mathtt{OFFLOAD}, id) \overset{\tau_0}{\hookleftarrow} P$ make the following checks: $\gamma := \Gamma(id) \neq \perp$ is a virtual channel and $P \in \gamma.\mathsf{users}$. If one of the checks fails, then drop the message. Otherwise proceed as the functionality $\mathcal{F}$.

Close: Upon $(\mathtt{CLOSE}, id) \overset{\tau_0}{\hookleftarrow} P$, check if $\gamma := \Gamma(id) \neq \perp$ and $P \in \gamma.\mathsf{endUsers}$. If γ is a virtual channel, additionally check that $\gamma.\mathsf{val} = \perp$. If not, drop the message. Else proceed as $\mathcal{F}$.

All other messages are dropped.

[a]In case more channels are being created at the same time, then none of the other creation requests can use of the tid.

[b]See Section -B what we formally meant by "reply".

IV. Simplifying the protocol descriptions

Similarly as the descriptions of our ideal functionality, the description our channel protocols, the protocol Π_{preL} presented in Section I-D and the protocol Π_V presented in Section II, exclude many natural checks that we would want an honest party to make. Let us give a few examples of requests which an honest party drops if received from the environment:

- The environment sends a malformed message to a party P (e.g. missing or additional parameters);
- A party P receives an instruction to create a channel γ but $P \notin \gamma.\mathsf{endUsers}$;
- A party P receives an instruction to create a virtual channel on top of a ledger channel that does not exist, does not belong to part P or is not sufficiently funded.
- Parties request to create a channel with validity whose validity time already expired (or is about to expire).

We define all these check as a wrapper $\mathcal{W}_{\text{checksP}}$ that can be seen as a extension of the wrapper provided by [1] for their ledger channel protocol.

Protocol wrapper: $\mathcal{W}_{\text{checksP}}$

The wrapper is defined for $\Pi \in \{\Pi_V(T), \Pi_{preL}(T)\}$. Below, we abbreviate $A := \gamma.\mathsf{Alice}$, $B := \gamma.\mathsf{Bob}$ and $I := \gamma.\mathsf{Ingrid}$.

$\boxed{\text{Party } P}$

Create: Upon $(\mathtt{CREATE}, \gamma, tid) \overset{\tau_0}{\hookleftarrow} \mathcal{E}$ check if: $P \in$

γ.endUsers; $\Gamma^P(\gamma.\mathsf{id}) = \perp$, $\Gamma^P_{pre}(\gamma.\mathsf{id}) = \perp$ and there is no channel γ' with $\gamma.\mathsf{id} = \gamma'.\mathsf{id}$ being created or pre-created; γ is valid according to the definition given in **??**; $\gamma.\mathsf{st} = \{(c_P, \texttt{One-Sig}_{pk_P}), (c_Q, \texttt{One-Sig}_{pk_Q})\}$ for $c_P, c_Q \in \mathbb{R}^{\geq 0}$. Depending on the type of channel, make the following additional checks:

ledger channel: There exists $(t, id, i, \theta) \in \widehat{\mathcal{L}}.\texttt{UTXO}$ such that $\theta = (c_P, \texttt{One-Sig}_P)$ for $(id, i) := tid$;[a]

virtual channel:

- If $P \in \gamma$.endUsers, then $\alpha := \Gamma^P(id_P) \neq \perp$ for $id_P := \gamma.\mathsf{subchan}(P)$; $\alpha.\mathsf{endUsers} = \{P, I\}$; there is no other virtual channel being created over α and α is currently not being updated; both P and I have enough funds in α.
- If $P = I$, then $\alpha := \Gamma^P(id_A) \neq \perp$ for $id_A := \gamma.\mathsf{subchan}(A)$; $\beta := \Gamma^P(id_B) \neq \perp$ for $id_B := \gamma.\mathsf{subchan}(B)$; $\alpha.\mathsf{endUsers} = \{A, I\}$; $\beta.\mathsf{endUsers} = \{B, I\}$; there is no other virtual channel being created over α or β; A and I have enough funds in α and B and I have enough funds in β.
- If $\gamma.\mathsf{val} \neq \perp$, then $\gamma.\mathsf{val} \geq \tau_0 + 4\Delta + 15T$.

If one of the checks fails, drop the message. Else proceed as in Π.

Pre-Create: Upon $(\texttt{PRE-CREATE}, \gamma, \texttt{TX}_\texttt{f}, i, t_{ofl}) \overset{\tau_0}{\hookleftarrow} \mathcal{E}$, check if: $P \in \gamma$.users, $\Gamma^P_{pre}(\gamma.\mathsf{id}) = \perp$, $\Gamma^P_{pre}(\gamma.\mathsf{id}) = \perp$ and there is no channel γ' with $\gamma.\mathsf{id} = \gamma'.\mathsf{id}$ being created or pre-created; γ is valid according to the definition given in **??**; $\gamma.\mathsf{st} = \{(c_P, \texttt{One-Sig}_{pk_P}), (c_Q, \texttt{One-Sig}_{pk_Q})\}$ for $c_P, c_Q \in \mathbb{R}^{\geq 0}$ and $\texttt{TX}_\texttt{f}$ is not a published transaction on $\widehat{\mathcal{L}}$. If one of the above checks fails, drop the message. Else proceed as in Π.

(Pre)-Update: Upon $(m, id, \vec{\theta}, t_{\mathsf{stp}}) \overset{\tau_0}{\hookleftarrow} \mathcal{E}$ check if: $\gamma := \Gamma^P(id) \neq \perp$ if $m = \texttt{UPDATE}$ and $\gamma := \Gamma^P_{pre}(id) \neq \perp$ if $m = \texttt{PRE-UPDATE}$. In both cases, check that there is no other update being preformed on γ; let $\vec{\theta} = (\theta_1, \ldots \theta_\ell) = ((c_1, \varphi_1), \ldots, (c_\ell, \varphi_\ell))$, then $\sum_{j \in [\ell]} c_i = \gamma.\mathsf{cash}$ and $\varphi_j \in \widehat{\mathcal{L}}.\mathcal{V}$ for each $j \in [\ell]$. If on of the checks fails, drop the message. Else proceed as in Π.

Upon $((\texttt{PRE-})\texttt{SETUP-OK}, id) \overset{\tau_2}{\hookleftarrow} \mathcal{E}$ check if: you accepted a message $((\texttt{PRE-})\texttt{UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \overset{\tau_0}{\hookleftarrow} \mathcal{E}$, where $t_2 - t_0 \leq t_{\mathsf{stp}} + T$ and the message is a reply to the message $((\texttt{PRE-})\texttt{SETUP}, id, \vec{tid})$ you sent in round τ_1 such that $\tau_2 - \tau_1 \leq t_{\mathsf{stp}}$[b]. If not, drop the message. Else proceed as in Π.

Upon $((\texttt{PRE-})\texttt{UPDATE-OK}, id) \overset{\tau_0}{\hookleftarrow} \mathcal{E}$, check if the message is a reply to the message $((\texttt{PRE-})\texttt{SETUP-OK}, id)$ you sent in round τ_0. If not, drop the message. Else proceed as in Π.

Upon $((\texttt{PRE-})\texttt{REVOKE}, id) \overset{\tau_0}{\hookleftarrow} \mathcal{E}$, check if the message is a reply to either $((\texttt{PRE-})\texttt{UPDATE-OK}, id)$ or $((\texttt{PRE-})\texttt{REVOKE-REQ}, id)$ you sent in round τ_0. If not, drop the message. Else proceed as in Π.

Offload: Upon receiving $(\texttt{OFFLOAD}, id) \overset{\tau_0}{\hookleftarrow} \mathcal{E}$ make the following checks: $\gamma := \Gamma(id) \neq \perp$ is a virtual channel and $P \in \gamma$.users. If one of the checks fails, then drop the message. Else proceed as in Π.

Close: Upon $(\texttt{CLOSE}, id) \overset{\tau_0}{\hookleftarrow} \mathcal{E}$, check if $\gamma := \Gamma^P(id) \neq \perp$, $P \in \gamma$.endUsers. If γ is a virtual channel, additionally check that $\gamma.\mathsf{val} = \perp$. If not, drop the message. Else proceed as in Π. All other messages are dropped.

[a]In case more channels are being created at the same time, then none of the other creation requests can use of the tid.

[b]See Section -B what we formally meant by "reply".

V. Simulation Wrapper

In this section we provide a proof for Theorem 1. In our proof, we provide the code for a simulator, that simulates the protocol Π_{preL} in the ideal world, having access to the functionalities $\widehat{\mathcal{L}}$ and $\mathcal{F}_{preL}$. In UC proofs it is required to provide a simulation of the real protocol in the ideal world even without knowledge of the secret inputs of the honest protocol participants. The main challenge is that this transcript of the simulation has to be indistinguishable to the environment $\mathcal{E}$ from the transcript of the real protocol execution. Yet, in our protocols, parties do not receive secret inputs, but are only instructed by the environment to take certain protocol actions, e.g. updating a channel. Hence the only challenge that arises during simulation is handling different behavior of malicious parties. Due to this, we only provide the simulator code for the protocol without arguing about indistinguishability of simulation and real protocol execution, since it naturally holds due to the reasons given above. In our simulation, we omit the case where all parties are honest, since the simulator simply has to follow the protocol description. In case of three protocol participants, we provide a simulation for all cases where two parties are corrupted and one party is honest, because these cases cover also all cases where just one party is corrupted. In other words, the case where two parties are honest is a combination of cases where each of these parties are honest individually.

Since the functionality $\mathcal{F}_{preL}$ incorporates, $\mathcal{F}_L$, we refer at some point of our simulation to the simulator code for ledger channels.

We note that the indistinguishability of the simulated transcript and the transcript of the real protocol can only hold if the security properties of the underlying adaptor signature scheme holds. Namely, we require the adaptor signature scheme to fulfill the unforgeability, witness extractability and adaptability properties.

Simulator for Wrapper protocol

Pre-Creat

$\boxed{\text{Case } A \text{ honest and } B \text{ corrupted}}$

1) Upon A sending $(\texttt{PRE-CREATE}, \gamma, \texttt{TX}_\texttt{f}, i, t_{ofl}) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_{preL}$ set $T_1 = 2$ and do the following:
2) If $\texttt{TX}_\texttt{f}.\mathsf{Output}[i].\mathsf{cash} \neq \gamma.\mathsf{cash}$, then ignore the message.
3) Set $id := \gamma.\mathsf{id}$, generate $(R_A, r_A) \leftarrow \mathsf{GenR}$, $(Y_A, y_A) \leftarrow \mathsf{GenR}$ and send $(\text{createInfo}, id, \texttt{TX}_\texttt{f}, i, t_{ofl}, R_A, Y_A) \overset{\tau_0}{\hookrightarrow} B$.
4) If $(\text{createInfo}, id, \texttt{TX}_\texttt{f}, i, t_{ofl}, R_B, Y_B) \overset{\tau_0+1}{\longleftarrow} B$, create:
$$[\texttt{TX}_\mathsf{c}] := \texttt{GenCommit}([\texttt{TX}_\texttt{f}], I_A, I_B, 0)$$
$$[\texttt{TX}_\mathsf{s}] := \texttt{GenSplit}([\texttt{TX}_\mathsf{c}].\mathsf{txid}\|1, \gamma.\mathsf{st})$$
for $I_A := (pk_A, R_B, Y_A)$, $I_B := (pk_B, R_B, Y_B)$. Else stop.
5) Compute $s^A_\mathsf{c} \leftarrow \mathsf{pSign}_{sk_A}([\texttt{TX}_\mathsf{c}], Y_B)$, $s^A_\mathsf{s} \leftarrow \mathsf{Sign}_{sk_A}([\texttt{TX}_\mathsf{s}])$ and send $(\text{createCom}, id, s^A_\mathsf{c}, s^A_\mathsf{s}) \overset{\tau_0+1}{\longrightarrow} B$.

6) If $(\text{createCom}, id, s_{\mathsf{c}}^B, s_{\mathsf{s}}^B) \xhookleftarrow{\tau_0+2} B$, s.t. $\mathsf{pVrfy}_{pk_B}([\mathtt{TX_c}], Y_A; s_{\mathsf{c}}^B) = 1$ and $\mathsf{Vrfy}_{pk_B}([\mathtt{TX_s}]; s_{\mathsf{s}}^B) = 1$, set

$$\mathtt{TX_c} := ([\mathtt{TX_c}], \{\mathsf{Sign}_{sk_A}([\mathtt{TX_c}]), \mathsf{Adapt}(s_{\mathsf{c}}^B, y_A)\})$$
$$\mathtt{TX_s} := ([\mathtt{TX_s}], \{s_{\mathsf{s}}^A, s_{\mathsf{s}}^B\})$$
$$\Gamma_{pre}^A(\gamma.\mathsf{id}) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_A, R_B, Y_B, s_{\mathsf{c}}^A), \mathtt{TX_s}, t_{ofl}).$$

and if B has not sent $(\mathtt{PRE\text{–}CREATE}, \gamma, \mathtt{TX_f}, i, t_{ofl})$ to $\mathcal{F}_{preL}$ send this message on behalf of B.

Pre-Update

Let $T_1 = 2$ and $T_2 = 1$ and let $|\vec{tid}| = 1$.

$\boxed{\text{Case } A \text{ is honest and } B \text{ is corrupted}}$

Upon A sending $(\mathtt{PRE\text{–}UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \xhookrightarrow{\tau_0} \mathcal{F}_{preL}$, proceed as follows:

1) Generate new revocation public/secret pair $(R_P, r_P) \leftarrow \mathsf{GenR}$ and a new publishing public/secret pair $(Y_P, y_P) \leftarrow \mathsf{GenR}$ and send $(\text{updateReq}, id, \vec{\theta}, t_{\mathsf{stp}}, R_A, Y_A) \xhookrightarrow{\tau_0^A} B$.
2) Upon $(\text{updateInfo}, id, h_B, Y_B, s_{\mathsf{s}}^B) \xhookleftarrow{\tau_0^A+2} B$, set $t_{\mathsf{lock}} := \tau_0^A + t_{\mathsf{stp}} + 5 + \Delta + t_{ofl}$, extract $\mathtt{TX_f}$ from $\Gamma_{pre}^B(id)$ and

$$[\mathtt{TX_c}] := \mathtt{GenCommit}([\mathtt{TX_f}], I_A, I_B, t_{\mathsf{lock}})$$
$$[\mathtt{TX_s}] := \mathtt{GenSplit}([\mathtt{TX_c}].\mathsf{txid}\|1, \vec{\theta}),$$

for $I_A := (pk_A, R_A, Y_A)$ and $I_B := (pk_B, R_B, Y_B)$. If it holds that $\mathsf{Vrfy}_{pk_B}([\mathtt{TX_s}]; s_{\mathsf{s}}^B) = 1$ continue. Else mark this execution as “failed” and stop.
3) If A sends $(\mathtt{PRE\text{–}SETUP\text{–}OK}, id) \xhookrightarrow{\tau_1^A \le \tau_0^A+2+t_{\mathsf{stp}}} \mathcal{F}_{preL}$, compute $s_{\mathsf{c}}^A \leftarrow \mathsf{pSign}_{sk_A}([\mathtt{TX_c}], Y_B), s_{\mathsf{s}}^A \leftarrow \mathsf{Sign}_{sk_A}([\mathtt{TX_s}])$ and send the message $(\text{update–commitA}, id, s_{\mathsf{c}}^A, s_{\mathsf{s}}^A) \xhookrightarrow{\tau_1^A} B$.
4) In round $\tau_1^A + 2$ distinguish the following cases:
   - If A receives $(\text{update–commitB}, id, s_{\mathsf{c}}^B) \xhookleftarrow{\tau_1^A+2} B$ check if B has not sent $(\mathtt{PRE\text{–}UPDATE\text{–}OK}, id) \xhookrightarrow{\tau_1^A+1} \mathcal{F}_{preL}$. If so send the message $(\mathtt{PRE\text{–}UPDATE\text{–}OK}, id) \xhookrightarrow{\tau_1^A+1} \mathcal{F}_{preL}$ on behalf of B. If $\mathsf{pVrfy}_{pk_B}([\mathtt{TX_c}], Y_A; s_{\mathsf{c}}^B) = 0$, then mark this execution as “failed” and stop.
   - If A receives $(\text{updateNotOk}, id, r_B) \xhookleftarrow{\tau_1^A+2} B$, where $(R_B, r_B) \in R$, add $\Theta^A(id) := \Theta^A(id) \cup ([\mathtt{TX_c}], r_B, Y_B, s_{\mathsf{c}}^A)$, instruct $\mathcal{F}_{preL}$ to send $(\mathtt{PRE\text{–}UPDATE\text{–}REJECT}, id) \hookrightarrow A$ and to stop and mark this execution as “failed” and stop.
   - Else, execute the simulator code for the procedure $\mathtt{Wait\text{–}if\text{–}Register}^A(id)$ and stop.
5) If A sends $(\mathtt{PRE\text{–}REVOKE}, id) \xhookrightarrow{\tau_1^A+2} \mathcal{F}_{preL}$, then parse $\Gamma_{pre}^A(id)$ as $(\gamma, \mathtt{TX_f}, (\overline{\mathtt{TX}}_{\mathsf{c}}, \bar{r}_A, \bar{R}_B, \bar{Y}_B, \bar{s}_{\mathtt{Com}}^A), \overline{\mathtt{TX}}_{\mathsf{s}})$ and update the channel space as $\Gamma_{pre}^A(id) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_A, R_B, Y_B, s_{\mathsf{c}}^A), \mathtt{TX_s})$, for $\mathtt{TX_s} := ([\mathtt{TX_s}], \{s_{\mathsf{s}}^A, s_{\mathsf{s}}^B\})$ and $\mathtt{TX_c} := ([\mathtt{TX_c}], \{\mathsf{Sign}_{sk_A}([\mathtt{TX_c}]), \mathsf{Adapt}(s_{\mathsf{c}}^B, y_A)\})$. Then send $(\text{revokeP}, id, \bar{r}_A) \xhookrightarrow{\tau_1^A+2} B$. Else, execute the simulator code for the procedure $\mathtt{Wait\text{–}if\text{–}Register}^A(id)$ and stop.
6) If A receives $(\text{revokeB}, id, \bar{r}_B) \xhookleftarrow{\tau_1^A+4} B$, check if B has not sent $(\mathtt{PRE\text{–}REVOKE}, id) \xhookrightarrow{\tau_1^B+2} \mathcal{F}_{preL}$. If so send $(\mathtt{PRE\text{–}REVOKE}, id) \xhookrightarrow{\tau_1^B+2} \mathcal{F}_L$ on behalf of B. Check if $(\bar{R}_B, \bar{r}_B) \in R$, then set

$$\Theta^B(id) := \Theta^A(id) \cup ([\overline{\mathtt{TX}}_{\mathsf{c}}], \bar{r}_B, \bar{Y}_B, \bar{s}_{\mathtt{Com}}^A)$$

Else execute the simulator code for the procedure $\mathtt{Wait\text{–}if\text{–}Register}^A(id)$ and stop.

$\boxed{\text{Case } B \text{ is honest and } A \text{ is corrupted}}$

Upon A sending $(\text{updateReq}, id, \vec{\theta}, t_{\mathsf{stp}}, h_A) \xhookrightarrow{\tau_0} B$, send the message $(\mathtt{PRE\text{–}UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \xhookrightarrow{\tau_0} \mathcal{F}_{preL}$ on behalf of A, if A has not already sent this message. Proceed as follows:

1) Upon $(\text{updateReq}, id, \vec{\theta}, t_{\mathsf{stp}}, R_A, Y_A) \xhookleftarrow{\tau_0^B} A$, generate $(R_B, r_B) \leftarrow \mathsf{GenR}$ and $(Y_B, y_B) \leftarrow \mathsf{GenR}$.
2) Set $t_{\mathsf{lock}} := \tau_0^B + t_{\mathsf{stp}} + 4 + \Delta + t_{ofl}$, extract $\mathtt{TX_f}$ from $\Gamma_{pre}^A(id)$ and

$$[\mathtt{TX_c}] := \mathtt{GenCommit}([\mathtt{TX_f}], I_A, I_B, t_{\mathsf{lock}})$$
$$[\mathtt{TX_s}] := \mathtt{GenSplit}([\mathtt{TX_c}].\mathsf{txid}\|1, \vec{\theta})$$

where $I_A := (pk_A, R_A, Y_A)$, $I_B := (pk_B, R_B, Y_B)$.
3) Compute $s_{\mathsf{s}}^B \leftarrow \mathsf{Sign}_{sk_B}([\mathtt{TX_s}])$, send $(\text{updateInfo}, id, R_B, Y_B, s_{\mathsf{s}}^B) \xhookrightarrow{\tau_0^B} A$.
4) If B receives $(\text{update–commitA}, id, s_{\mathsf{c}}^A, s_{\mathsf{s}}^A) \xhookleftarrow{\tau_1^B \le \tau_0^B+2+t_{\mathsf{stp}}} A$ then send $(\mathtt{PRE\text{–}SETUP\text{–}OK}, id) \xhookrightarrow{\tau_1^B} \mathcal{F}_{preL}$ on behalf of A, if A has not sent this message.
5) Check if $\mathsf{pVrfy}_{pk_P}([\mathtt{TX_c}], Y_Q; s_{\mathsf{c}}^P) = 1$ and $\mathsf{Vrfy}_{pk_P}([\mathtt{TX_s}]; s_{\mathsf{s}}^P) = 1$. Else mark this execution as “failed” and stop.
6) If B sends $(\mathtt{PRE\text{–}UPDATE\text{–}OK}, id) \xhookrightarrow{\tau_1^B} \mathcal{F}_{preL}$, then compute $s_{\mathsf{c}}^B \leftarrow \mathsf{pSign}([\mathtt{TX_c}], Y_A)$ and send $(\text{update–commitB}, id, s_{\mathsf{c}}^B) \xhookrightarrow{\tau_1^B} A$. Else send $(\text{updateNotOk}, id, r_B) \xhookrightarrow{\tau_1^B} A$, mark this execution as “failed” and stop.
7) Parse $\Gamma_{pre}^B(id)$ as $(\gamma, \mathtt{TX_f}, (\overline{\mathtt{TX}}_{\mathsf{c}}, \bar{r}_B, \bar{R}_A, \bar{Y}_A, \bar{s}_{\mathtt{Com}}^B), \overline{\mathtt{TX}}_{\mathsf{s}})$. If B receives $(\text{revokeA}, id, \bar{r}_A) \xhookleftarrow{\tau_1^B+2} A$, send $(\mathtt{PRE\text{–}REVOKE}, id) \xhookrightarrow{\tau_1^B+2} \mathcal{F}_{preL}$ on behalf of A, if A has not sent this message.

   Else if you do not receive $(\text{revokeA}, id, \bar{r}_A) \xhookleftarrow{\tau_1^B+2} A$ or if $(\bar{R}_A, \bar{r}_A) \notin R$, execute the simulator code of the procedure $\mathtt{Wait\text{–}if\text{–}Register}^B(id)$ and stop.
8) If B sends $(\mathtt{PRE\text{–}REVOKE}, id) \xhookrightarrow{\tau_1^B+2} \mathcal{F}_{preL}$, then set

$$\Theta^B(id) := \Theta^B(id) \cup ([\overline{\mathtt{TX}}_{\mathsf{c}}], \bar{r}_A, \bar{Y}_A, \bar{s}_{\mathtt{Com}}^B)$$
$$\Gamma_{pre}^B(id) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_B, R_A, Y_A, s_{\mathsf{c}}^B), \mathtt{TX_s}),$$

for $\mathtt{TX_s} := ([\mathtt{TX_s}], \{s_{\mathsf{s}}^A, s_{\mathsf{s}}^B\})$ and $\mathtt{TX_c} := ([\mathtt{TX_c}], \{\mathsf{Sign}_{sk_B}([\mathtt{TX_c}]), \mathsf{Adapt}(s_{\mathsf{c}}^A, y_B)\})$. Then $(\text{revokeB}, id, \bar{r}_B) \xhookrightarrow{\tau_1^B+2} A$ and stop. Else, in round $\tau_1^B + 2$, execute the simulator code of the procedure $\mathtt{Wait\text{–}if\text{–}Register}^B(id)$ and stop.

Register

$\boxed{\text{Case } A \text{ honest and } B \text{ corrupted}}$

For party A in every round τ_0 do the following:

1) For each $id \in \{0,1\}^*$ s.t. $\Gamma^A_{pre}(id) \neq \bot$:
2) Parse $\Gamma^A_{pre}(id) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_A, R_B, Y_B, s^A_c), \mathtt{TX_s}, t_{ofl}, x)$
3) If $\mathtt{TX_f}$ appeared on-chain in this round, then
   a) Set $\Gamma(id) := (\gamma, \mathtt{TX_f}, (\mathtt{TX_c}, r_A, R_B, Y_B, s^A_c), \mathtt{TX_s})$.
   b) Set $\Gamma^A_{pre}(id) := \bot$
   c) If $x = \mathtt{in\text{–}dispute}$, then execute the simulator code for $\mathtt{L\text{–}ForceClose}^A(id)$.

$\mathtt{Wait\text{–}if\text{–}Register}(id)$

$\boxed{\text{Case } A \text{ honest and } B \text{ corrupted}}$

Let τ_0 be the current round. Let $X := \Gamma^A_{pre}(id)$. Then set $\Gamma^A_{pre}(id) := (X, \mathtt{in\text{–}dispute})$.

VI. Simulation Virtual Channels

In this section we provide a proof for Theorem **??**. In our proof, we provide the code for a simulator, that simulates the protocol Π_V in the ideal world having access to the functionalities $\widehat{\mathcal{L}}$ and $\mathcal{F}_V$.

We note that since during our simulation, no ERROR messages are produced by the functionality, the protocol satisfies the security properties of the functionality $\mathcal{F}_V$ as mentioned in Section **??**.

Simulator for creating virtual channels

creating virtual channels

$\boxed{\text{Case } A \text{ is honest and } I \text{ are } B \text{ are corrupt}}$

Upon A sending $(\mathtt{CREATE}, \gamma) \xhookrightarrow{\tau_0^A} \mathcal{F}_V$ set $T_1 = 6T + t_{\mathsf{stp}}$ proceed as follows:

1) Let $id_\alpha := \gamma.\mathsf{subchan}(A)$ and compute
$$\theta_A := \mathtt{GenVChannelOutput}(\gamma, A).$$
2) Upon A sending $(\mathtt{UPDATE}, id_\alpha, \theta_A, t_{\mathsf{stp}}) \xhookrightarrow{\tau_0^A} \mathcal{F}_L$ execute the simulator code of the update procedure for the generalized channels until the message $(\mathtt{SETUP}, id_\alpha, tid_A)$ is sent by $\mathcal{F}_L$. If the execution stops send (peaceful–reject, id_α) $\hookrightarrow \mathcal{F}_V$.
3) Upon A receiving $(\mathtt{SETUP}, id_\alpha, tid_A) \xhookleftarrow{\tau_1^A \leq \tau_0^A + T} \mathcal{F}_L$, execute the simulator code for $\mathtt{SetupVChannel}$ with input (γ, tid_A).
4) If this execution of $\mathtt{SetupVChannel}$ is recorded "failed" stop. Otherwise execute the simulator code of the update procedure for the generalized channels until the end. If the execution failed (I does not revoke) instruct $\mathcal{F}_V$ to $\mathtt{L\text{–}ForceClose}(id_\alpha)$.
5) If B or I have not sent $(\mathtt{CREATE}, \gamma) \hookrightarrow \mathcal{F}_V$ send this message on their behalf.
6) Upon A receiving $(\mathtt{CREATED}, \gamma) \xhookleftarrow{\tau_2^A \leq \tau_1^A + 5T} \mathcal{F}_V$, mark γ as created, i.e. update $\Gamma^A(\gamma.\mathsf{id})$ from $(\bot, x)$ to (γ, x).

$\boxed{\text{Case } I \text{ is honest and } A, B \text{ are corrupted}}$

Upon I sending $(\mathtt{CREATE}, \gamma) \xhookrightarrow{\tau_0^I} \mathcal{F}_V$ proceed as follows:

1) Set $id_\alpha = \gamma.\mathsf{subchan}(A), id_\beta = \gamma.\mathsf{subchan}(B)$ and generate
$$\theta_A := \mathtt{GenVChannelOutput}(\gamma, A)$$
$$\theta_B := \mathtt{GenVChannelOutput}(\gamma, B)$$
2) Upon A and B sending $(\mathtt{UPDATE}, id_\alpha, \theta_A, t_{\mathsf{stp}}) \xhookrightarrow{\tau_0^A} \mathcal{F}_L$ and $(\mathtt{UPDATE}, id_\alpha, \theta_B, t_{\mathsf{stp}}) \xhookrightarrow{\tau_0^B} \mathcal{F}_L$, execute the simulator code for the update procedure of the generalized channel functionality until the message $(\mathtt{UPDATE\text{–}REQ}, id_\alpha, \theta_A, t_{\mathsf{stp}}, tid_A)$ and $(\mathtt{UPDATE\text{–}REQ}, id_\alpha, \theta_B, t_{\mathsf{stp}}, tid_B)$ are sent by $\mathcal{F}_L$.
3) If in round $\tau_1^I \leq \tau_0^I + T$, I has received both $(\mathtt{UPDATE\text{–}REQ}, id_\alpha, \theta_A, t_{\mathsf{stp}}, tid_A) \hookleftarrow \mathcal{F}_L$ and $(\mathtt{UPDATE\text{–}REQ}, id_\beta, \theta_B, t_{\mathsf{stp}}, tid_B) \hookleftarrow \mathcal{F}_L$, then execute the simulator code of $\mathtt{SetupVChannel}$ with inputs (γ, tid_A, tid_B). Or send (peaceful–reject, id_α) $\hookrightarrow \mathcal{F}_V$ and (peaceful–reject, id_β) $\hookrightarrow \mathcal{F}_V$ if instructed by $\mathcal{E}$. Else stop.
4) If in round $\tau_2^I \leq \tau_1^I + t_{\mathsf{stp}} + T$, I receives both $(\mathtt{SETUP\text{–}OK}, id_\alpha)) \hookleftarrow \mathcal{F}_L$ and $(\mathtt{SETUP\text{–}OK}, id_\beta)) \hookleftarrow \mathcal{F}_L$, continue executing the simulator code of the update procedure of generalized channels until the messages $(\mathtt{REVOKE\text{–}REQ}, id_\alpha)$ and $(\mathtt{REVOKE\text{–}REQ}, id_\alpha)$ are sent by $\mathcal{F}_L$. Otherwise stop.
5) If in round $\tau_3^I \leq \tau_2^I + 4T$ you have received both $(\mathtt{REVOKE\text{–}REQ}, id_\alpha) \hookleftarrow \mathcal{F}_L$ and $(\mathtt{REVOKE\text{–}REQ}, id_\beta) \hookleftarrow \mathcal{F}_L$, continue executing the simulator code of the update procedure of generalized channels until the end. Otherwise stop.
6) If A or B have not sent $(\mathtt{CREATE}, \gamma) \hookrightarrow \mathcal{F}_V$ send this message on their behalf. Update $\Gamma^I(\gamma.\mathsf{id})$ from $(\bot, x)$ to (γ, x).

$\mathtt{SetupVChannel}$ **for channels without validity**

$\boxed{\text{Case } A \text{ is honest and } I, B \text{ are corrupted}}$

1) Create the body of the funding transaction:
$$\mathtt{TX}^\gamma_\mathtt{f}.\mathsf{Input} := (tid_A, tid_B)$$
$$\mathtt{TX}^\gamma_\mathtt{f}.\mathsf{Output} := ((\gamma.\mathsf{cash}, \mathtt{Multi\text{–}Sig}_{\{\gamma.\mathsf{endUsers}\}}), (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}, \mathtt{One\text{–}Sig}_{pk_I}))$$
2) Upon A sending $(\mathtt{PRE\text{–}CREATE}, \gamma, \mathtt{TX_f}, 1, t_{ofl}) \xhookrightarrow{t_0} \mathcal{F}_{preL}$ where $t_{ofl} = 2T + 8\Delta$, execute the simulator code for the Pre-Create procedure of the $\mathcal{F}_{preL}$ functionality.
3) Upon A receiving $(\mathtt{PRE\text{–}CREATED}, \gamma.id) \xhookleftarrow{\tau_1 \leq \tau_0 + T} \mathcal{F}_{preL}$ then sign the funding transaction, i.e. $s^B_\mathtt{f} \leftarrow \mathsf{Sign}_{sk_B}([\mathtt{TX}^\gamma_\mathtt{f}])$ and send (createFund, $\gamma.\mathsf{id}, s^A_\mathtt{f}, [\mathtt{TX}^\gamma_\mathtt{f}]) \xhookrightarrow{\tau_1} I$. Else record this execution as "failed" and stop.
4) Upon receiving (createFund, $\gamma.\mathsf{id}, s^B_\mathtt{f}, s^I_\mathtt{f}) \xhookleftarrow{\tau_1 + 1} I$, verify all signatures, i.e. check:
$$\mathsf{Vrfy}_{pk_B}([\mathtt{TX}^\gamma_\mathtt{f}]; s^B_\mathtt{f}) = 1$$
$$\mathsf{Vrfy}_{pk_I}([\mathtt{TX}^\gamma_\mathtt{f}], s^I_\mathtt{f}) = 1.$$

If all checks pass define

$$\mathtt{TX}^\gamma_\mathtt{f} := \{([\mathtt{TX}^\gamma_\mathtt{f}], s^A_\mathtt{f}, s^B_\mathtt{f}, s^I_\mathtt{f})\},$$

and set

$$\Gamma^A(\gamma.\mathsf{id}) := (\bot, \mathtt{TX}^\gamma_\mathtt{f}, tid_A)$$

and consider procedure successfully completed. Else record this execution as "failed" and stop.

$\boxed{\text{Case } I \text{ is honest and } A, B \text{ are corrupted}}$

5) If I receives $(\text{createFund}, \gamma.\mathsf{id}, s^A_\mathtt{f}, [\mathtt{TX}^\gamma_\mathtt{f}]) \xleftarrow{\tau_2 \le \tau_0 + T + 1} A$ and $(\text{createFund}, \gamma.\mathsf{id}, s^B_\mathtt{f}, [\mathtt{TX}^\gamma_\mathtt{f}]) \xleftarrow{\tau_2} B$, verify the funding transaction and signatures of A and B, i.e. check:

$$\mathsf{Vrfy}_{pk_A}([\mathtt{TX}^\gamma_\mathtt{f}]; s^A_\mathtt{f}) = 1$$
$$\mathsf{Vrfy}_{pk_B}([\mathtt{TX}^\gamma_\mathtt{f}], s^B_\mathtt{f}) = 1$$
$$(tid_A, tid_B) = \mathtt{TX}^\gamma_\mathtt{f}.\mathsf{Input}$$
$$(\gamma.\mathsf{cash} + \gamma.\mathsf{fee}, \mathtt{One\text{–}Sig}_{pk_I}) \in \mathtt{TX}^\gamma_\mathtt{f}.\mathsf{Output}.$$

6) If all checks pass, sign the funding transaction, i.e. compute

$$s^I_\mathtt{f} := \mathsf{Sign}_{sk_I}([\mathtt{TX}^\gamma_\mathtt{f}]),$$
$$\mathtt{TX}^\gamma_\mathtt{f} := \{([\mathtt{TX}^\gamma_\mathtt{f}], s^A_\mathtt{f}, s^B_\mathtt{f}, s^I_\mathtt{f})\}.$$

Store $\Gamma^I(\gamma.\mathsf{id}) := (\bot, \mathtt{TX}^\gamma_\mathtt{f})$. Then send $(\text{createFund}, \gamma.\mathsf{id}, s^B_\mathtt{f}, s^I_\mathtt{f}) \xhookrightarrow{\tau_2} A$ and $(\text{createFund}, \gamma.\mathsf{id}, s^A_\mathtt{f}, s^I_\mathtt{f}) \xhookrightarrow{\tau_2} B$, and consider procedure successfully completed. Else record this execution as "failed" and stop.

$\mathtt{SetupVChannel}$ for channels With validity

$\boxed{\text{Case } A \text{ is honest and } B, I \text{ are corrupted}}$

1) Send $(\text{createInfo}, \gamma.\mathsf{id}, tid_A) \xhookrightarrow{\tau_0} B$.
2) In round $\tau_1 = \tau_0 + 1$, create the body of the funding transaction:

$$\mathtt{TX}^\gamma_\mathtt{f}.\mathsf{Input} := (tid_A)$$
$$\mathtt{TX}^\gamma_\mathtt{f}.\mathsf{Output} := ((\gamma.\mathsf{cash}, \mathtt{Multi\text{–}Sig}_{\{\gamma.\mathsf{endUsers}\}}), (\gamma.\mathsf{fee}/2, \mathtt{One\text{–}Sig}_{pk_I}))$$

3) Upon A sending $(\mathtt{PRE\text{–}CREATE}, \gamma, \mathtt{TX}_\mathtt{f}, 1, t_{ofl}) \xhookrightarrow{\tau_1} \mathcal{F}_{preL}$ where $t_{ofl} = \gamma.\mathsf{val} + 3\Delta$, execute the simulator code for the Pre-Create procedure of the $\mathcal{F}_{preL}$ functionality. If A does not receive the message $(\mathtt{PRE\text{–}CREATED}, \gamma.id) \xleftarrow{\tau_2 \le \tau_1 + T} \mathcal{F}_{preL}$ then mark this execution as "failed" and stop.
4) If A receives $(\text{createFund}, \gamma.\mathsf{id}, s^I_\mathtt{f}) \xleftarrow{\tau_2 + 2} I$, verify the signature, i.e. check:

$$\mathsf{Vrfy}_{sk_I}([\mathtt{TX}^\gamma_\mathtt{f}]; s^I_\mathtt{f}) = 1.$$

If the check passes, compute a signature on the fund transaction:

$$s^A_\mathtt{f} := \mathsf{Sign}_{sk_A}([\mathtt{TX}^\gamma_\mathtt{f}]),$$
$$\mathtt{TX}^{\gamma,\mathtt{A}}_\mathtt{f} := \{([\mathtt{TX}^\gamma_\mathtt{f}], s^I_\mathtt{f}, s^A_\mathtt{f})\}.$$

Else record this execution as "failed" and stop.
5) Set

$$\Gamma^A(\gamma.\mathsf{id}) := (\bot, \mathtt{TX}^{\gamma,\mathtt{A}}_\mathtt{f}, tid_A)$$

and consider procedure successfully completed.

$\boxed{\text{Case } B \text{ is honest and } A, I \text{ are corrupted}}$

1) If $(\text{createInfo}, \gamma.\mathsf{id}, tid_A) \xleftarrow{\tau_0 + 1} A$, create the body of the funding and the first commit and split transactions:

$$\mathtt{TX}^\gamma_\mathtt{f}.\mathsf{Input} := (tid_A)$$
$$\mathtt{TX}^\gamma_\mathtt{f}.\mathsf{Output} := ((\gamma.\mathsf{cash}, \mathtt{Multi\text{–}Sig}_{\{\gamma.\mathsf{endUsers}\}}), (\gamma.\mathsf{fee}/2, \mathtt{One\text{–}Sig}_{pk_I}))$$
$$\mathtt{TX}^\gamma_\mathtt{refund}.\mathsf{Input} := (\mathtt{TX}^\gamma_\mathtt{f}.\mathsf{txid}||2, tid_B)$$
$$\mathtt{TX}^\gamma_\mathtt{refund}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}, \mathtt{One\text{–}Sig}_{pk_I}).$$

Else record this execution as "failed" and stop.
2) Upon B sending $(\mathtt{PRE\text{–}CREATE}, \gamma, \mathtt{TX}_\mathtt{f}, 1, t_{ofl}) \xhookrightarrow{\tau_1 = \tau_0 + 1} \mathcal{F}_{preL}$ where $t_{ofl} = \gamma.\mathsf{val} + 3\Delta$, execute the simulator code for the Pre-Create procedure of the $\mathcal{F}_{preL}$ functionality. If B does not receive $(\mathtt{PRE\text{–}CREATED}, \gamma.id) \xleftarrow{\tau_2 \le \tau_1 + T} \mathcal{F}_{preL}$ then mark this execution as "failed" and stop.
3) Compute a signature on the refund transaction, i.e., $s^B_\mathtt{Ref} \leftarrow \mathsf{Sign}_{sk_B}([\mathtt{TX}^\gamma_\mathtt{refund}])$ and define $\mathtt{TX}^{\gamma,\mathtt{B}}_\mathtt{f} := \{([\mathtt{TX}^\gamma_\mathtt{f}])\}$. Then, send $(\text{createFund}, \gamma.\mathsf{id}, s^B_\mathtt{Ref}, [\mathtt{TX}^\gamma_\mathtt{refund}], [\mathtt{TX}^\gamma_\mathtt{f}]) \xhookrightarrow{\tau_2} I$, set

$$\Gamma^B(\gamma.\mathsf{id}) := (\bot, \mathtt{TX}^{\gamma,\mathtt{B}}_\mathtt{f}, tid_B)$$

and consider procedure successfully completed. Else record this execution as "failed" and stop.

$\boxed{\text{Case } I \text{ is honest and } A, B \text{ are corrupted}}$

4) If I receives the message $(\text{createFund}, \gamma.\mathsf{id}, s^B_\mathtt{Ref}, [\mathtt{TX}^\gamma_\mathtt{refund}], [\mathtt{TX}^\gamma_\mathtt{f}]) \xleftarrow{\tau_3 \le \tau_0 + T + 2} B$, verify the fund and refund transactions and signature of B, i.e. check:

$$\mathsf{Vrfy}_{sk_B}([\mathtt{TX}^\gamma_\mathtt{refund}]; s^B_\mathtt{Ref}) = 1.$$
$$[\mathtt{TX}^\gamma_\mathtt{refund}].\mathsf{Input} = (\mathtt{TX}^\gamma_\mathtt{f}.\mathsf{txid}||2, tid_B),$$
$$[\mathtt{TX}^\gamma_\mathtt{refund}].\mathsf{Output} = (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}, \mathtt{One\text{–}Sig}_{pk_I}),$$
$$[\mathtt{TX}^\gamma_\mathtt{f}].\mathsf{Output}[2] = (\gamma.\mathsf{fee}/2, \mathtt{One\text{–}Sig}_{pk_I})$$

If all checks pass, sign the fund and refund transactions, i.e. compute

$$s^I_\mathtt{Ref} := \mathsf{Sign}_{sk_I}([\mathtt{TX}^\gamma_\mathtt{refund}]), s^I_\mathtt{f} := \mathsf{Sign}_{sk_I}([\mathtt{TX}^\gamma_\mathtt{f}]),$$
$$\mathtt{TX}^\gamma_\mathtt{refund} := \{([\mathtt{TX}^\gamma_\mathtt{refund}], s^I_\mathtt{Ref}, s^B_\mathtt{Ref})\}.$$

Store $\Gamma^I(\gamma.\mathsf{id}) := (\bot, [\mathtt{TX}^\gamma_\mathtt{f}], \mathtt{TX}^\gamma_\mathtt{refund}, tid_A, tid_B)$. Then send $(\text{createFund}, \gamma.\mathsf{id}, s^I_\mathtt{f}) \xhookrightarrow{\tau_3} A$, and consider procedure successfully completed. Else record this execution as "failed" and stop.

Function $\mathtt{GenVChannelOutput}(\gamma, P)$

Return θ, where $\theta.\mathsf{cash} = \gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2$ and $\theta.\varphi$ is defined as follows

$$\theta.\varphi = \begin{cases} \texttt{Multi-Sig}_{\gamma.\mathsf{users}} \vee (\texttt{One-Sig}_P \wedge \texttt{CheckRelative}_{(T+4\Delta)}), \\ \qquad\qquad \text{if } \gamma.\mathsf{val} = \bot \\ \texttt{Multi-Sig}_{A,I} \vee (\texttt{One-Sig}_I \wedge \texttt{CheckLockTime}_{\gamma.\mathsf{val}}), \\ \qquad\qquad \text{if } \gamma.\mathsf{val} \neq \bot \wedge P = A \\ \texttt{Multi-Sig}_{B,I} \vee (\texttt{One-Sig}_B \wedge \texttt{CheckLockTime}_{\gamma.\mathsf{val}+2\Delta}), \\ \qquad\qquad \text{if } \gamma.\mathsf{val} \neq \bot \wedge P = B \end{cases}$$

Simulator for updating virtual channels

Update virtual channels

Case A is honest and B is corrupt

Below we abbreviate $\mathcal{F}_{preL} := \mathcal{F}_{preL}(T, 1)$ and assume A is the initiating party.

1) Upon A sending $(\texttt{UPDATE}, id, \vec{\theta}, t_{\mathsf{stp}}) \overset{\tau_0^A}{\hookrightarrow} \mathcal{F}_{preL}$, execute the simulator for the pre-update procedure of the $\mathcal{F}_{preL}$ functionality from beginning until PRE–SETUP is sent. If this execution is marked "failed" stop.
2) Upon A sending $(\texttt{SETUP-OK}, id) \xleftarrow{\tau_2^A \leq \tau_1^A + t_{\mathsf{stp}}} \mathcal{F}_{preL}$, continue executing the simulator code until step 4. If this execution is marked "failed" stop.
3) If A does not receive $(\texttt{PRE-UPDATE-OK}, id) \xleftarrow{\tau_3^A \leq \tau_2^A + T} \mathcal{F}_{preL}$ or $(\texttt{PRE-UPDATE-REJECT}, id) \xleftarrow{\tau_3^A \leq \tau_2^A + T} \mathcal{F}_{preL}$, execute the simulator code for the procedure $\texttt{Offload}^A(id)$ and stop.
4) Upon A sending $(\texttt{PRE-REVOKE}, id) \overset{\tau_3^A}{\hookrightarrow} \mathcal{F}_{preL}$ continue executing the simulator code until the end. If this execution is marked as "failed" execute the simulator code for the procedure $\texttt{Offload}^A(id)$ and stop.
5) Upon A receiving $(\texttt{PRE-UPDATED}, id) \xleftarrow{\tau_4^A \leq \tau_3^A + T} \mathcal{F}_{preL}$, update the channel space, i.e., let $\gamma := \Gamma^A(id)$, set $\gamma.\mathsf{st} := \vec{\theta}$ and $\Gamma(id) := \gamma$. Else if this execution is marked as "failed" execute the simulator code for the procedure $\texttt{Offload}^A(id)$ and stop.

Case B is honest and A is corrupt

1) Let τ_0^B be the round in which B receives $(\texttt{PRE-UPDATE-REQ}, id, \vec{\theta}, t_{\mathsf{stp}}, tid) \overset{\tau_0^B}{\hookleftarrow} \mathcal{F}_{preL}$.
2) Let $\tau_1^B \leq \tau_0 + t_{\mathsf{stp}} + T$ be the round in which B receives the message $(\texttt{PRE-SETUP-OK}, id) \xleftarrow{\tau_1^B \leq \tau_0 + t_{\mathsf{stp}} + T} \mathcal{F}_{preL}$.
3) Upon B sending $(\texttt{UPDATE-OK}, id) \overset{\tau_1^B}{\hookleftarrow} \mathcal{F}_{preL}$ execute the simulator code of the pre-update procedure for the $\mathcal{F}_{preL}$ functionality until the message PRE–REVOKE–REQ is sent by the functionality and let the this round be $\tau_2^B \leq \tau_1^B + T$. If this execution is marked as "failed" execute the simulator code of the procedure $\texttt{Offload}^B(id)$ and stop.
4) Upon B sending $(\texttt{PRE-REVOKE}, id) \overset{\tau_2^B}{\hookrightarrow} \mathcal{F}_{preL}$ continue executing the simulator code until the end. If this execution is marked as "failed" execute the simulator code for the procedure $\texttt{Offload}^B(id)$ and stop.
5) Upon B receiving $(\texttt{PRE-UPDATED}, id) \xleftarrow{\tau_3^B \leq \tau_2^B + T} \mathcal{F}_{preL}$, update the channel space, i.e., let $\gamma := \Gamma^B(id)$, set $\gamma.\mathsf{st} := \vec{\theta}$ and $\Gamma(id) := \gamma$.

Simulator for offloading virtual channels

Offloading virtual channels without validity

Case A honest and I, B corrupted

1) Extract γ and $\texttt{TX}_{\mathsf{f}}^{\gamma}$ from $\Gamma^A(id)$ and tid_A, tid_B from $\texttt{TX}_{\mathsf{f}}^{\gamma}$. Then define $id_\alpha := \gamma.\mathsf{subchan}(A)$. Upon A sending $(\texttt{CLOSE}, id_\alpha) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_L$ execute the simulator code for the close procedure of generalized ledger channels.
2) If A receives $(\texttt{CLOSED}, id_\alpha) \xleftarrow{\tau_1 \leq \tau_0 + T + 3\Delta} \mathcal{F}_L$, check that a transaction with tid_A appeared on $\widehat{\mathcal{L}}$. Else stop.
3) Let $T_2 := \tau_1 + T + 3\Delta$ and distinguish:
   - If in round $\tau_2 \leq T_2$ a transaction with tid_B appeared on $\widehat{\mathcal{L}}$, then $(\mathrm{post}, \texttt{TX}_{\mathsf{f}}^{\gamma}) \overset{\tau_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
   - Else in round T_2 create the punishment transaction $\texttt{TX}_{\mathsf{pun}}$ as $\texttt{TX}_{\mathsf{pun}}.\mathsf{Input} := tid_A$, $\texttt{TX}_{\mathsf{pun}}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \texttt{One-Sig}_{pk_A})$ and $\texttt{TX}_{\mathsf{pun}}.\mathsf{Witness} := \mathsf{Sign}_{sk_A}([\texttt{TX}_{\mathsf{pun}}])$. Then $(\mathrm{post}, \texttt{TX}_{\mathsf{pun}}) \overset{T_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
4) Let $T_3 := \tau_2 + \Delta$ and distinguish the following two cases:
   - The transaction $\texttt{TX}_{\mathsf{f}}^{\gamma}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_3 \leq T_3$, then update $\Gamma^A := \texttt{ToLedgerChannel}(\Gamma^A, \gamma.\mathsf{id})$ and set $m :=$ offloaded.
   - The transaction $\texttt{TX}_{\mathsf{pun}}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_3 \leq T_3$, then define $\Gamma^A(\gamma.\mathsf{id}) = \bot$ and set $m :=$ punished.
5) Return m in round τ_3.

Case I honest and A, B corrupted

1) Extract γ and $\texttt{TX}_{\mathsf{f}}^{\gamma}$ from $\Gamma^I(id)$ and tid_A, tid_B from $\texttt{TX}_{\mathsf{f}}^{\gamma}$. Then define $id_\alpha := \gamma.\mathsf{subchan}(A)$, $id_\beta := \gamma.\mathsf{subchan}(B)$ Upon I sending the messages $(\texttt{CLOSE}, id_\alpha) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_L$ and $(\texttt{CLOSE}, id_\beta) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_L$ execute the simulator code of the close procedure for the generalized channels.
2) If I receives both $(m, id_\alpha) \xleftarrow{\tau_1^A \leq \tau_0 + T + 3\Delta} \mathcal{F}_L$ and $(\texttt{CLOSED}, id_\beta) \xleftarrow{\tau_1^B \leq \tau_0 + T + 3\Delta} \mathcal{F}_L$, check that a transaction with tid_A and a transaction with tid_B appeared on $\widehat{\mathcal{L}}$. Then publish $(\mathrm{post}, \texttt{TX}_{\mathsf{f}}^{\gamma}) \overset{\tau_1}{\hookrightarrow} \widehat{\mathcal{L}}$, where $\tau_1 := \max\{\tau_1^A, \tau_1^B\}$. Otherwise set $\Gamma^I(id) = \bot$ and stop.
3) Once $\texttt{TX}_{\mathsf{f}}^{\gamma}$ is accepted by $\widehat{\mathcal{L}}$ in round $\tau_2 \leq \tau_1 + \Delta$, then $\Gamma^I(\gamma.\mathsf{id}) = \bot$ and return "offloaded".

Offloading virtual channels with validity

Case A honest and B, I corrupted

1) Extract γ, tid_A and $\texttt{TX}_{\mathsf{f}}^{\gamma}$ from $\Gamma^A(id)$. Then define $id_\alpha := \gamma.\mathsf{subchan}(A)$ and Upon A sending $(\texttt{CLOSE}, id_\alpha) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_L$ execute the simulator code of the close procedure of the generalized ledger channel.
2) If A receives $(\texttt{CLOSED}, id_\alpha) \xleftarrow{\tau_1 \leq \tau_0 + T + 3\Delta} \mathcal{F}_L$, then post $(\mathrm{post}, \texttt{TX}_{\mathsf{f}}^{\gamma,\mathsf{A}}) \overset{\tau_2}{\hookrightarrow} \widehat{\mathcal{L}}$. Otherwise, set $\Gamma^A(\gamma.\mathsf{id}) = \bot$ and stop.
3) Once $\texttt{TX}_{\mathsf{f}}^{\gamma}$ is accepted by $\widehat{\mathcal{L}}$ in round $\tau_2 \leq \tau_1 + \Delta$, then set $\Gamma_L^A(id) := \Gamma^A(id)$, $\Gamma^A(id) := \bot$ and return "offloaded".

Case B honest and A, I corrupted

1) Extract γ, tid_B and $[\mathtt{TX}_\mathtt{f}^\gamma]$ from $\Gamma^B(id)$. Then define $id_\beta := \gamma.\mathsf{subchan}(B)$ and Upon B sending $(\mathtt{CLOSE}, id_\beta) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_L$ execute the simulator code of the close procedure of the generalized ledger channel.
2) If B receives $(\mathtt{CLOSED}, id_\beta) \xleftarrow{\tau_1 \le \tau_0 + T + 3\Delta} \mathcal{F}_L$, then continue. Otherwise, set $\Gamma^B(\gamma.\mathsf{id}) = \perp$ and stop.
3) Create the punishment transaction $\mathtt{TX}_\mathsf{pun}$ as $\mathtt{TX}_\mathsf{pun}.\mathsf{Input} := tid_B$, $\mathtt{TX}_\mathsf{pun}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \mathtt{One\text{-}Sig}_{pk_B})$ and $\mathtt{TX}_\mathsf{pun}.\mathsf{Witness} := \mathsf{Sign}_{sk_B}([\mathtt{TX}_\mathsf{pun}])$. Then wait until round $\tau_2 := \max\{\tau_1, \gamma.\mathsf{val} + 2\Delta\}$ and send $(\text{post}, \mathtt{TX}_\mathsf{pun}) \overset{\tau_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
4) Let $T_3 := \tau_2 + \Delta$ and distinguish the following two cases:
   - A transaction with identifier $\mathtt{TX}_\mathtt{f}^\gamma.\mathsf{txid}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_3 \le T_3$, then update $\Gamma_L^B(id) := \Gamma^B(id)$ and set $m :=$ offloaded.
   - The transaction $\mathtt{TX}_\mathsf{pun}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_3 \le T_3$, then define $\Gamma^B(\gamma.\mathsf{id}) = \perp$, and set $m :=$ punished.
5) Return m in round τ_3.

Case I honest and A, B corrupted

1) Extract γ, tid_A, tid_B, $\mathtt{TX}_\mathtt{refund}^\gamma$ and $[\mathtt{TX}_\mathtt{f}^\gamma]$ from $\Gamma^I(id)$. Then define $id_\alpha := \gamma.\mathsf{subchan}(A)$, $id_\beta := \gamma.\mathsf{subchan}(B)$ and upon I sending $(\mathtt{CLOSE}, id_\alpha) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_L$ and $(\mathtt{CLOSE}, id_\beta) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_L$ execute the simulator code of the close procedure of the generalized ledger channel for both id_α and id_β.
2) If I receives both $(\mathtt{CLOSED}, id_\alpha) \xleftarrow{\tau_1^A \le \tau_0 + T + 3\Delta} \mathcal{F}_L$ and $(\mathtt{CLOSED}, id_\beta) \xleftarrow{\tau_1^B \le \tau_0 + T + 3\Delta} \mathcal{F}_L$,then continue. Otherwise, set $\Gamma^I(\gamma.\mathsf{id}) = \perp$ and stop.
3) Create the punishment transaction $\mathtt{TX}_\mathsf{pun}$ as $\mathtt{TX}_\mathsf{pun}.\mathsf{Input} := tid_A$, $\mathtt{TX}_\mathsf{pun}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \mathtt{One\text{-}Sig}_{pk_I})$ and $\mathtt{TX}_\mathsf{pun}.\mathsf{Witness} := \mathsf{Sign}_{sk_I}([\mathtt{TX}_\mathsf{pun}])$. Then wait until round $\tau_2 := \max\{\tau_1^A, \gamma.\mathsf{val}\}$ and send $(\text{post}, \mathtt{TX}_\mathsf{pun}) \overset{\tau_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
4) Let $T_3 := \tau_2 + \Delta$ and distinguish the following two cases:
   - A transaction with identifier $\mathtt{TX}_\mathtt{f}^\gamma.\mathsf{txid}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_3' \le T_3$, send $(\text{post}, \mathtt{TX}_\mathtt{refund}^\gamma) \overset{\tau_4}{\hookrightarrow} \widehat{\mathcal{L}}$ where $\tau_4 := \max\{\tau_1^B, \tau_3'\}$. Once $\mathtt{TX}_\mathtt{refund}^\gamma$ is accepted by $\widehat{\mathcal{L}}$ in round $\tau_5 \le \tau_4 + \Delta$, set $\Gamma^I(\gamma.\mathsf{id}) = \perp$ and $m :=$ offloaded.
   - The transaction $\mathtt{TX}_\mathsf{pun}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_3'' \le T_3$, then set $\Gamma^I(\gamma.\mathsf{id}) = \perp$ and $m :=$ punished.
5) Return m in round τ_6 where $\tau_6 := \max\{\tau_5, \tau_3''\}$.

Simulator for punishing in a virtual channel

- Upon a party sending $(\mathtt{PUNISH}) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_L$, execute the simulator code for the punish procedure of the generalized channels.
- Execute the simulator code for both `Punish` and `Punish–Validity`.

<u>`Punish`</u>

Case A honest and I, B corrupted

For every $id \in \{0,1\}^*$, such that γ which $\gamma.\mathsf{val} = \perp$ can be extracted from $\Gamma^A(id)$ do the following:

1) Extract $\mathtt{TX}_\mathtt{f}^\gamma$ from $\Gamma^A(id)$ and tid_A, tid_B from $\mathtt{TX}_\mathtt{f}^\gamma$. Check if tid_A appeared on $\widehat{\mathcal{L}}$. If not, then stop.
2) Denote $T_2 := \tau_1 + T + 3\Delta$ and distinguish:
   - If in round $\tau_2 \le T_2$ the transaction with tid_B appeared on $\widehat{\mathcal{L}}$, then $(\text{post}, \mathtt{TX}_\mathtt{f}^\gamma) \overset{\tau_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
   - Else in round T_2 create the punishment transaction $\mathtt{TX}_\mathsf{pun}$ as $\mathtt{TX}_\mathsf{pun}.\mathsf{Input} := tid_A$, $\mathtt{TX}_\mathsf{pun}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \mathtt{One\text{-}Sig}_{pk_A})$ and $\mathtt{TX}_\mathsf{pun}.\mathsf{Witness} := \mathsf{Sign}_{sk_A}([\mathtt{TX}_\mathsf{pun}])$ and $(\text{post}, \mathtt{TX}_\mathsf{pun}) \overset{T_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
3) Let $T_3 := \tau_2 + \Delta$ and distinguish the following two cases:
   - The transaction $\mathtt{TX}_\mathtt{f}^\gamma$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_3 \le T_3$, then update $\Gamma^A := \mathtt{ToLedgerChannel}(\Gamma^A, \gamma.\mathsf{id})$.
   - The transaction $\mathtt{TX}_\mathsf{pun}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_3 \le T_3$, then define $\Gamma^A(\gamma.\mathsf{id}) = \perp$.

Case I honest and A, B corrupted

For every $id \in \{0,1\}^*$, such that γ with $\gamma.\mathsf{val} = \perp$ can be extracted from $\Gamma^I(id)$ do the following:

1) Extract $\mathtt{TX}_\mathtt{f}^\gamma$ from $\Gamma^I(id)$ and tid_A, tid_B from $\mathtt{TX}_\mathtt{f}^\gamma$. Check if for some $P \in \{A, B\}$ a transaction with identifier tid_P appeared on $\widehat{\mathcal{L}}$. If not, then stop.
2) Denote $id_\alpha := \gamma.\mathsf{subchan}(Q)$ where $Q = \gamma.\mathsf{otherParty}(P)$ and upon I sending $(\mathtt{CLOSE}, id_\alpha) \overset{\tau_0}{\hookrightarrow} \mathcal{F}_L$ execute simulator the code for the punish procedure of the generalized virtual channels.
3) If I receives $(\mathtt{CLOSED}, id_\alpha) \xleftarrow{\tau_1 \le \tau_0 + T + 3\Delta} \mathcal{F}_L$ and tid_Q appeared on $\widehat{\mathcal{L}}$, $(\text{post}, \mathtt{TX}_\mathtt{f}^\gamma) \overset{\tau_1}{\hookrightarrow} \widehat{\mathcal{L}}$. Otherwise set $\Gamma^I(id) = \perp$ and stop.
4) Once $\mathtt{TX}_\mathtt{f}^\gamma$ is accepted by $\widehat{\mathcal{L}}$ in round τ_2, such that $\tau_2 \le \tau_1 + \Delta$, set $\Gamma^I(id) = \perp$.

<u>`Punish–Validity`</u>

Case A honest and I, B corrupted

For every $id \in \{0,1\}^*$, such that γ which $\gamma.\mathsf{val} \ne \perp$ can be extracted from $\Gamma^A(id)$ do the following:

1) Extract tid_A and $\mathtt{TX}_\mathtt{f}^\gamma$ from $\Gamma^A(id)$. Check if tid_A appeared on $\widehat{\mathcal{L}}$. If not, then stop.
2) Send $(\text{post}, \mathtt{TX}_\mathtt{f}^\gamma) \overset{\tau_1}{\hookrightarrow} \widehat{\mathcal{L}}$.
3) Once $\mathtt{TX}_\mathtt{f}^\gamma$ is accepted by $\widehat{\mathcal{L}}$ in round $\tau_2 \le \tau_1 + \Delta$, set $\Gamma_L^A(id) := \Gamma^A(id)$, $\Gamma^A(id) := \perp$.

Case B honest and A, I corrupted

For every $id \in \{0,1\}^*$, such that γ which $\gamma.\mathsf{val} \ne \perp$ can be extracted from $\Gamma^B(id)$ do the following:

1) Extract tid_B and $[\mathtt{TX}_\mathtt{f}^\gamma]$ from $\Gamma^B(id)$. Check if tid_B or $[\mathtt{TX}_\mathtt{f}^\gamma].\mathsf{txid}$ appeared on $\widehat{\mathcal{L}}$. If not, then stop.
2) If $\mathtt{TX}_\mathtt{f}^\gamma$ appeared on $\widehat{\mathcal{L}}$, set $\Gamma_L^B(id) := \Gamma^B(id)$, $\Gamma^B(id) := \perp$ and stop.
3) If tid_B appeared on $\widehat{\mathcal{L}}$, create the punishment transaction $\mathtt{TX}_\mathsf{pun}$ as $\mathtt{TX}_\mathsf{pun}.\mathsf{Input} := tid_B$, $\mathtt{TX}_\mathsf{pun}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \mathtt{One\text{-}Sig}_{pk_B})$ and $\mathtt{TX}_\mathsf{pun}.\mathsf{Witness} := \mathsf{Sign}_{sk_B}([\mathtt{TX}_\mathsf{pun}])$. Then wait until round $\tau_2 := \max\{\tau_1, \gamma.\mathsf{val} + 2\Delta\}$ and send $(\text{post}, \mathtt{TX}_\mathsf{pun}) \overset{\tau_2}{\hookrightarrow} \widehat{\mathcal{L}}$.
4) If transaction $\mathtt{TX}_\mathsf{pun}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_3 \le \tau_2 + \Delta$, then define $\Gamma^B(\gamma.\mathsf{id}) = \perp$.

$\boxed{\text{Case } I \text{ honest and } A, B \text{ corrupted}}$

For every $id \in \{0,1\}^*$, such that γ which $\gamma.\mathsf{val} \neq \bot$ can be extracted from $\Gamma^I(id)$ do the following:

1) Extract tid_A, tid_B, $\mathtt{TX}^\gamma_{\mathtt{refund}}$ and $[\mathtt{TX}^\gamma_{\mathtt{f}}]$ from $\Gamma^I(id)$. Check if tid_A or tid_B appeared on $\widehat{\mathcal{L}}$ or $\tau_1 = \gamma.\mathsf{val} - (3\Delta + T)$. If not, then stop.
2) Distinguish the following cases:
   - If $\tau_1 = \gamma.\mathsf{val} - (3\Delta + T)$, define $id_\alpha := \gamma.\mathsf{subchan}(A)$, $id_\beta := \gamma.\mathsf{subchan}(B)$ and upon I sending the messages $(\mathtt{CLOSE}, id_\alpha) \xhookrightarrow{\tau_1} \mathcal{F}_L$ and $(\mathtt{CLOSE}, id_\beta) \xhookrightarrow{\tau_1} \mathcal{F}_L$ execute the simulator code for the close procedure of the generalized channels for both channels id_α and id_β.
   - If tid_B appeared on $\widehat{\mathcal{L}}$, send $(\mathtt{CLOSE}, id_\alpha) \xhookrightarrow{\tau_1} \mathcal{F}_L$.
3) If a transaction with identifier tid_A appeared on $\widehat{\mathcal{L}}$ in round $\tau_2 \leq \tau_1 + T + 3\Delta$, create the punishment transaction $\mathtt{TX}_{\mathsf{pun}}$ as $\mathtt{TX}_{\mathsf{pun}}.\mathsf{Input} := tid_A$, $\mathtt{TX}_{\mathsf{pun}}.\mathsf{Output} := (\gamma.\mathsf{cash} + \gamma.\mathsf{fee}/2, \mathtt{One\text{–}Sig}_{pk_I})$ and $\mathtt{TX}_{\mathsf{pun}}.\mathsf{Witness} := \mathsf{Sign}_{sk_I}([\mathtt{TX}_{\mathsf{pun}}])$. Then wait until round $\tau_3 := \max\{\tau_2, \gamma.\mathsf{val}\}$ and send $(\text{post}, \mathtt{TX}_{\mathsf{pun}}) \xhookrightarrow{\tau_3} \widehat{\mathcal{L}}$.
4) Distinguish the following two cases:
   - The transaction $\mathtt{TX}^\gamma_{\mathtt{f}}.\mathsf{txid}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_4 \leq \tau_3 + \Delta$, send $(\text{post}, \mathtt{TX}^\gamma_{\mathtt{refund}}) \xhookrightarrow{\tau_5} \widehat{\mathcal{L}}$ where $\tau_5 := \max\{\gamma.\mathsf{val} + \Delta, \tau_4\}$. Once $\mathtt{TX}^\gamma_{\mathtt{refund}}$ is accepted by $\widehat{\mathcal{L}}$ in round $\tau_6 \leq \tau_5 + \Delta$, set $\Gamma^I(\gamma.\mathsf{id}) = \bot$ and stop.
   - The transaction $\mathtt{TX}_{\mathsf{pun}}$ was accepted by $\widehat{\mathcal{L}}$ in $\tau_4 \leq \tau_3 + \Delta$, then set $\Gamma^I(\gamma.\mathsf{id}) = \bot$.

Simulator for Close in a virtual channel

Closing virtual channels

$\boxed{\text{Case } A \text{ honest and } I, B \text{ corrupted}}$

1) Upon A sending $(\mathtt{CLOSE}, id) \xhookrightarrow{\tau_0^A} \mathcal{F}_V$ or in round $\tau_0^A = \gamma.\mathsf{val} - (4\Delta + 7T)$ if $\gamma.\mathsf{val} \neq \bot$, proceed as follows:
2) Extract γ, $\mathtt{TX}^\gamma_{\mathtt{f}}$, tid_A from $\Gamma^A(id)$. Parse
$$\mathtt{TX}^\gamma_{\mathtt{s}}.\mathsf{Output} = \Big((c_A, \mathtt{One\text{–}Sig}_{pk_A}), (c_B, \mathtt{One\text{–}Sig}_{pk_B})\Big).$$
3) Compute the new state of the channel $id_\alpha := \gamma.\mathsf{subchan}(A)$ as
$$\vec{\theta}_A := \{(c_A, \mathtt{One\text{–}Sig}_{pk_A}), (\gamma.\mathsf{cash} - c_A + \frac{\gamma.\mathsf{fee}}{2}, \mathtt{One\text{–}Sig}_{pk_I})\}$$
Then, upon A sending $(\mathtt{UPDATE}, id_\alpha, \vec{\theta}_A, 0) \xhookrightarrow{\tau_0^A} \mathcal{F}_L$ execute the simulator code for the update procedure of generalized channels until $(\mathtt{SETUP}, id_\alpha, \vec{tid'}_A)$ is sent by $\mathcal{F}_L$. If this execution fails instruct $\mathcal{F}_V$ to execute $\mathtt{V\text{–}ForceClose}(id)$ and stop.
4) Upon A sending $(\mathtt{SETUP\text{–}OK}, id_\alpha) \xhookrightarrow{\tau_1^A \leq \tau_0^A + T} \mathcal{F}_L$ continue executing the simulator code for the update procedure of generalized channels until A receives $(\mathtt{UPDATE\text{–}OK}, id_\alpha) \xhookleftarrow{\tau_2^A \leq \tau_1^A + 2T} \mathcal{F}_L$. If this execution fails instruct $\mathcal{F}_V$ to execute $\mathtt{V\text{–}ForceClose}(id)$ and stop.
5) Upon A sending $(\mathtt{REVOKE}, id_\alpha) \xhookrightarrow{\tau_2^A} \mathcal{F}_L$ continue executing the simulator code for the update procedure of generalized channels until A receives $(\mathtt{UPDATED}, id_\alpha) \xhookleftarrow{\tau_3^A \leq \tau_2^A + 2T} \mathcal{F}_L$, then set $\Gamma^A(id) := \bot$ and stop. If this execution fails instruct $\mathcal{F}_V$ to execute $\mathtt{V\text{–}ForceClose}(id)$ and stop.

$\boxed{\text{Case } I \text{ honest and } A, B \text{ corrupted}}$

1) Upon I sending $(\mathtt{CLOSE}, id) \xhookrightarrow{\tau_0^I} \mathcal{F}_V$ or in round $\tau_0^A = \gamma.\mathsf{val} - (4\Delta + 7T)$ if $\gamma.\mathsf{val} \neq \bot$, proceed as follows:
2) Extract γ, $\mathtt{TX}^\gamma_{\mathtt{f}}$, tid_A and tid_B from $\Gamma^I(id)$.
3) Let $id_\alpha = \gamma.\mathsf{subchan}(A)$, $id_\beta = \gamma.\mathsf{subchan}(B)$ and $c := \gamma.\mathsf{cash}$, execute the simulator code for the update procedure of generalized channels until $(\mathtt{UPDATE\text{–}REQ}, id_\alpha, \vec{tid'}_A, \vec{\theta}_A, 0) \hookleftarrow \mathcal{F}_L$ and $(\mathtt{UPDATE\text{–}REQ}, id_\beta, \vec{tid'}_B, \vec{\theta}_B, 0) \hookleftarrow \mathcal{F}_L$ are sent by $\mathcal{F}_L$ until round $\tau_1^I \leq \tau_0^I + T$.
4) Check that for some c_A, c_B s.t. $c_A + c_B + \gamma.\mathsf{fee} = c$ it holds
$$\vec{\theta}_A = \{(c_A, \mathtt{One\text{–}Sig}_{pk_A}), (c - c_A + \gamma.\mathsf{fee}/2, \mathtt{One\text{–}Sig}_{pk_I})\}$$
$$\vec{\theta}_B = \{(c_B, \mathtt{One\text{–}Sig}_{pk_B}), (c - c_B + \gamma.\mathsf{fee}/2, \mathtt{One\text{–}Sig}_{pk_I})\}$$
If not, then stop.
5) continue executing the simulator code for the update procedure of generalized channels until I receives $(\mathtt{SETUP\text{–}OK}, id_\alpha) \hookleftarrow \mathcal{F}_L$ and $(\mathtt{SETUP\text{–}OK}, id_\beta)) \hookleftarrow \mathcal{F}_L$ until round $\tau_2^I \leq \tau_1^I + T$. Otherwise stop.
6) Upon I sending $(\mathtt{UPDATE\text{–}OK}, id_\beta) \xhookrightarrow{\tau_2^I} \mathcal{F}_L$ continue executing the simulator code for the update procedure of generalized channels until I receives the messages $(\mathtt{REVOKE\text{–}REQ}, id_\alpha) \hookleftarrow \mathcal{F}_L$ and $(\mathtt{REVOKE\text{–}REQ}, id_\beta) \hookleftarrow \mathcal{F}_L$ until round $\tau_3^I \leq \tau_2^I + 2T$.
7) Upon I sending $(\mathtt{REVOKE}, id_\alpha) \xhookrightarrow{\tau_3^I} \mathcal{F}_L$ and $(\mathtt{REVOKE}, id_\beta) \xhookrightarrow{\tau_3^I} \mathcal{F}_L$ executing the simulator code for the update procedure of generalized channels until the end and set $\Gamma^I(id) := \bot$.