Appendix A
Simulation $\mathcal{S}_{CMC}$

The simulator $\mathcal{S}_{CMC}$ interacts with the adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ while accessing $\mathcal{F}_{CMC}$, $\mathcal{F}_{WT}$, $\{\mathcal{F}^{\theta}_{MC}\}_{\theta\in NUM}$, and $\mathcal{L}_{CMC}$. The ledger is honest, whereas users, supervisors, and the Watchtower WT may be statically corrupted.

Internal state. For each payment identifier pid, the simulator stores a tuple $(NUM, ID, m, t, \mathsf{HoldOK}, \mathsf{UpdOK}, \mathsf{status})$ and a deviation log $\mathsf{Dev}[pid]$. Possible values of status are init, holding, ready, wt_done, settled, and abort.

Stage I: Transaction type identification.

- *Case I1 (honest payer, honest payer-channel supervisors).* Upon receiving $(\mathsf{pay}, pid, pr, pe, m, t)$, the simulator forwards it to $\mathcal{F}_{CMC}$. If the payment is identified as cross-channel, the simulator initializes the internal record for pid and outputs an acceptance transcript on behalf of the payer-channel supervisor.
- *Case I2 (corrupt payer, honest payer-channel supervisors).* The simulator forwards the request to $\mathcal{F}_{CMC}$ and outputs either acceptance or rejection transcripts according to the ideal decision.
- *Case I3 (honest payer, corrupt payer-channel supervisor).* The simulator forwards the request to $\mathcal{F}_{CMC}$. Any externally inconsistent acceptance or rejection transcript generated by the adversary is recorded as a deviation for potential reverse supervision.
- *Case I4 (corrupt payer and corrupt payer-channel supervisor).* The simulator lets the adversary determine external transcripts while maintaining internal consistency with the ideal functionality.

Stage II: Fund holding and pre-deduction.

- *Case H1 (honest supervisors in all involved channels).* For each channel in NUM, the simulator delivers a holding request on behalf of the payer-channel supervisor. The other supervisor in the channel performs tentative pre-deduction. Upon success, the simulator outputs a holding confirmation and records the channel as successfully held.
- *Case H2 (some channel supervisors corrupt, honest payer).* For honest channels, the simulator behaves as in Case H1. For corrupted channels, the adversary may refuse holding, apply incorrect pre-deduction, or output inconsistent confirmations. All such behavior is recorded in the deviation log. If any honest channel rejects holding, the simulator aborts the payment.
- *Case H3 (corrupt payer, honest supervisors).* If the ideal functionality reaches the holding stage, the simulator proceeds as in Case H1. Concurrent reuse of funds by the payer is prevented by the tentative balance updates in the ideal world, and subsequent requests are simulated as rejected.
- *Case H4 (corrupt payer and corrupt supervisors).* The simulator combines the behaviors of Case H2 and Case H3 and records all detected inconsistencies.

Stage III: Encrypted lookup-table updates.

- *Case U1 (honest supervisors, arbitrary Watchtower).* For each successfully held channel, the simulator constructs a well-formed encrypted update consistent with the ideal state and delivers it to $\mathcal{F}_{WT}$ on behalf of the channel supervisors.
- *Case U2 (corrupt supervisors in some channels, honest Watchtower).* Encrypted updates from honest channels are delivered as in Case U1. Malformed or missing updates from corrupted channels prevent Watchtower from satisfying its execution condition.
- *Case U3 (corrupt Watchtower).* The adversary may drop, reorder, or falsely acknowledge updates. Any discrepancy between submitted updates and Watchtower behavior is recorded as a deviation.

Stage IV: Watchtower assistance.

- *Case W1 (honest Watchtower).* If encrypted updates from all channels are available, the simulator outputs a Watchtower confirmation transcript and marks the payment as completed by Watchtower.
- *Case W2 (corrupt Watchtower, stalling or rejecting).* If the watchtower rejects or stalls despite all conditions being satisfied, the simulator aborts the payment and records the deviation.
- *Case W3 (corrupt Watchtower, premature confirmation).* If Watchtower confirms execution without receiving all required updates, the simulator records a critical inconsistency for later slashing.

Stage V: Local ledger finalization.

- *Case F1 (honest supervisors in all channels).* After Watchtower confirmation, the simulator instructs each involved channel to finalize its local balances and outputs a successful cross-channel payment completion transcript.
- *Case F2 (corrupt supervisors in some channels).* If corrupted supervisors refuse to finalize or output inconsistent local states, the simulator records the deviation and reduces the execution to dispute resolution.

Stage VI: Dispute resolution and slashing.

- *Reverse supervision.* When an inconsistency is detectable by honest users, the simulator constructs abstract evidence and submits it to $\mathcal{L}_{CMC}$ on behalf of the user.
- *Mutual supervision.* When contradictory transcripts from Watchtower or supervisors exist, the simulator submits corresponding evidence on behalf of an honest supervisor.

Fig. 1: Simulation $\mathcal{S}_{CMC}$

Appendix B
Ledger functionality $\mathcal{L}_{CMC}$

The ledger functionality $\mathcal{L}_{CMC}$ interacts with $\mathcal{F}_{CMC}$, $\mathcal{F}_{WT}$

Deposit locking. Upon receiving $(\mathsf{LockWT}, WT, Dep_{WT})$, lock Dep_{WT} as the collateral of WT.

Deposit release. Upon receiving $(\mathsf{ReleaseWT}, pid, WT)$, if pid is marked as successfully settled and no valid evidence against WT has been accepted, release Dep_{WT} to WT.

Evidence submission. Upon receiving $(\mathsf{SubmitEvidence}, pid, WT, \mathsf{etype}, \pi)$ from any party, where $\mathsf{etype} \in \{\mathsf{reverse}, \mathsf{mutual}\}$:

1) If pid is already resolved, ignore the submission.
2) Otherwise, evaluate the abstract predicate $\mathsf{Verify}_{\mathsf{etype}}(pid, WT, \pi)$.

Slashing. If $\mathsf{Verify}_{\mathsf{etype}}(pid, WT, \pi) = 1$, slash Dep_{WT}, transfer it to the reporter, and mark pid as resolved.

Rejection. If $\mathsf{Verify}_{\mathsf{etype}}(pid, WT, \pi) = 0$, ignore the evidence and keep Dep_{WT} locked.

Finalization marker. Upon receiving $(\mathsf{Finalize}, pid)$ from $\mathcal{F}_{CMC}$, mark pid as successfully settled unless it has already been resolved by slashing.

Fig. 2: Ledger functionality $\mathcal{L}_{CMC}$