VeCroToken: An Efficient, Verifiable, and Privacy-Preserving Cross-Chain Model for Consortium Blockchains based on zk-SNARKs (Supplementary Material)

Ke Ding, Xiaoyan Hu, Weicheng Zhou, Guang Cheng, Ruidong Li, and Hua Wu

S1. Introduction

This supplementary note provides a standalone Universal Composability (UC) treatment that unifies transaction-security (verifiability, balance safety, no double-spending) and privacy (amount, balance, fund-correlation) using an ideal functionality $\mathcal{F}_{\mathsf{VCT}^\star}$. In $\mathcal{F}_{\mathsf{VCT}^\star}$, local prechecks return FAIL without any public footprint. $\mathsf{Invalid}$ outcomes are triggered by the result of a cross-chain verification service $\mathcal{F}_{\mathsf{XVer}}$. Transfers follow a single-consumption relay state machine $\mathsf{Send} \rightarrow \{\mathsf{Finished}, \mathsf{Invalid}\}$. We provide the UC proof with a simulator, a bridge lemma, a hybrid sequence, and explicit privacy bounds.

S2. Setting and Notation

We consider two consortium chains C, D and a relay chain R with deterministic finality (BFT). Each account a keeps a native balance $v[a]$ and a zk-balance $z[a]$. The public conversion coefficient is n_c. Operations are Deposit, Withdraw, Send, Receive. Each transfer has a unique identifier Tid.

S3. Global Functionalities in the $(\mathcal{G}_{\mathsf{Ledger}}, \mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{XVer}})$-Hybrid World

Fig. S1 summarizes $\mathcal{G}_{\mathsf{Ledger}}$ as the unique finalized log per chain, providing a deterministic total order and finality points. The later UC proof relies on this finality to serialize visible actions on the relay ledger L_R and to rule out post-finality rewrites.

Global Ledger Functionality $\mathcal{G}_{\mathsf{Ledger}}$.
State: For each chain $X \in \{C, D, R\}$, a totally ordered log L_X with deterministic finality.
Interfaces: $\mathsf{Post}(X, \mathsf{tx})$ appends to L_X and finalizes. $\mathsf{Read}(X, \mathsf{qid})$ returns authenticated answers.
Safety: No two finalized entries share a Tid. Finalized order is consistent to all honest parties.
Adversary: May delay but cannot rewrite finalized entries nor create conflicting finalized states.

Fig. S1. Global ledger functionality $\mathcal{G}_{\mathsf{Ledger}}$.

Fig. S2 depicts $\mathcal{F}_{\mathsf{CRS}}$, which provisions the CRS and circuit keys. The zero-knowledge simulator trapdoor is available only to the ideal-world simulator S. No application plaintexts or balances are exposed by $\mathcal{F}_{\mathsf{CRS}}$.

CRS Functionality $\mathcal{F}_{\mathsf{CRS}}$.
On $\mathsf{Setup}(\lambda)$ sample a common reference string CRS and publish it to all parties and the adversary. For each circuit type $i \in \{\mathsf{dep}, \mathsf{wd}, \mathsf{send}, \mathsf{rece}\}$, parties register $(\mathsf{pk}_i, \mathsf{vk}_i)$ consistent with CRS. A simulation trapdoor for zk-SNARKs is available to the simulator only.

Fig. S2. CRS functionality $\mathcal{F}_{\mathsf{CRS}}$.

Cross-Chain Verification Functionality $\mathcal{F}_{\mathsf{XVer}}$ (CrossVerify).
Goal: Mediate all cross-chain checks. Its *Accept/Reject* result triggers whether $\mathcal{F}_{\mathsf{VCT}^\star}$ will publicly output $\mathsf{Invalid}$.

State: Consults $\mathcal{G}_{\mathsf{Ledger}}$ and uses the relay log L_R for visible actions.

Interfaces:
$\underline{\mathsf{VerifyDep}(\mathsf{Tid})}$: Fetch Tx by Tid from source ledger. If $Tx.\mathsf{sta} \neq \mathsf{Valid}$ return Reject. Parse $\chi = \{\mathsf{pubk}, E_z, E_z^*, E_v, E_v^*\}$ and proof π_{dep}. Compute $r \leftarrow \Pi.\mathsf{Verify}(\mathsf{vk}_{\mathsf{dep}}, \chi, \pi_{\mathsf{dep}})$. If $r = 1$ then $\mathsf{Post\_R}(R_{\mathsf{AccN}}, Tx)$ to L_R and return Accept, else Reject.
$\underline{\mathsf{VerifyWd}(\mathsf{Tid})}$: Same with $\chi = \{\mathsf{pubk}, E_z, E_z^*, E_v, E_v^*\}$ and π_{wd}. On $r = 1$ $\mathsf{Post\_R}(R_{\mathsf{AccN}}, Tx)$.
$\underline{\mathsf{VerifySend}(\mathsf{Tid}_{\mathsf{A}}, M)}$: Parse $\chi = \{\mathsf{pubk}_A, E_{zA}, E_{zA}^*, M\}$ and π_{send}. Compute $r \leftarrow \Pi.\mathsf{Verify}(\mathsf{vk}_{\mathsf{send}}, \chi, \pi_{\mathsf{send}})$. If $r = 1$ set $Tx.\mathsf{sta} = \mathsf{Send}$ and $\mathsf{Post\_R}(Tx_{\mathsf{send}})$ to L_R, else Reject.
$\underline{\mathsf{VerifyRece}(\mathsf{Tid}_{\mathsf{A}}, \mathsf{Tid}_{\mathsf{B}}, M_{\mathsf{rece}}, E_{zB})}$: Parse $\chi = \{\mathsf{pubk}_B, E_{zB}, E_{zB}^*, M_{\mathsf{rece}}\}$ and π_{rece}. Obtain Tx_{send} via $\mathsf{Tid}_{\mathsf{A}}$ from L_R and parse M_{send} and sta. Obtain E_{zB}^{ledger} from chain D's account certificate. Accept iff $\Pi.\mathsf{Verify}(\mathsf{vk}_{\mathsf{rece}}, \chi, \pi_{\mathsf{rece}}) = 1 \ \wedge \ M_{\mathsf{rece}} = M_{\mathsf{send}} \ \wedge \ E_{z_B} = E_{zB}^{\mathsf{ledger}} \ \wedge \ \mathsf{sta} = \mathsf{Send}$. On reject: set $\mathsf{sta}_{\mathsf{send}} = \mathsf{Invalid}$ and $\mathsf{Post\_R}(Tx_{\mathsf{send}})$. Return Reject. On accept: set $\mathsf{sta}_{\mathsf{send}} = \mathsf{Finished}$, $\mathsf{Post\_R}(Tx_{\mathsf{send}})$ and $\mathsf{Post\_R}(R_{\mathsf{AccN}}, Tx_{\mathsf{rece}})$. Return Accept.

Remarks: All four interfaces perform a visible Post_R on Accept. RECEIVE additionally annotates $\mathsf{Tid}_{\mathsf{A}}$ as Finished/Invalid.

Fig. S3. Cross-chain verification functionality $\mathcal{F}_{\mathsf{XVer}}$.

S3.1 Sufficiency of $\mathcal{F}_{\mathsf{XVer}}$ Checks for Integrity

Fig. S3 summarizes the CrossVerify interfaces and their visible actions. The two explicit checks in VerifyRece, namely $M_{\mathsf{rece}} = M_{\mathsf{send}}$ and $E_{z_B} = E_{zB}^{\mathsf{ledger}}$ with $\mathsf{sta} = \mathsf{Send}$, can prevent identity forgery and amount tampering.

Lemma 1 (Integrity via Pair-Consistency and Identity-Binding)**.** *Suppose the zk-SNARK system is knowledge-sound and $\mathcal{G}_{\mathsf{Ledger}}$ provides deterministic finality. If $\mathcal{F}_{\mathsf{XVer}}$ accepts a* RECEIVE *with $\Pi.\mathsf{Verify}(\mathsf{vk}_{\mathsf{rece}}, \chi, \pi_{\mathsf{rece}}) = 1$, $M_{\mathsf{rece}} = M_{\mathsf{send}}$, $E_{z_B} = E_{zB}^{\mathsf{ledger}}$ and the paired send has $\mathsf{sta} = \mathsf{Send}$, then (i) the credited receiver is the ledger owner of E_{zB}^{ledger} (no identity forgery), and (ii) the transferred amount m equals the one committed to in the paired* SEND *(no amount tampering).*

Proof. Knowledge soundness of the accepting proof yields witnesses (z_B, z_B^*, m, k) satisfying the ciphertext-binding relations of Sec. S9: $M_{\mathsf{rece}} = \mathsf{E}(m, k)$, $E_{zB} = \mathsf{EA}(z_B, \mathsf{pubk}_B)$, $E_{zB}^* = \mathsf{EA}(z_B^*, \mathsf{pubk}_B)$, and the balance relation $z_B^* - z_B = m$. Equality $M_{\mathsf{rece}} = M_{\mathsf{send}}$ pins the same m (and length) as used in the paired send's circuit. Otherwise two different witnesses would be required, contradicting soundness because the public input includes M. Equality $E_{z_B} = E_{z_B}^{\mathsf{ledger}}$ ties pubk_B to the ledger-certified receiver account. Forging a different identity would require either (a) ledger tampering (ruled out by $\mathcal{G}_{\mathsf{Ledger}}$), or (b) a second accepting proof with different public key but the same E_{zB}^{ledger}, which violates the circuit's key-consistency. Finally, $\mathsf{sta} = \mathsf{Send}$ plus the relay SU invariant forces a single post-finality transition of $\mathsf{Tid}_{\mathsf{A}}$ to $\{\mathsf{Finished}, \mathsf{Invalid}\}$, precluding double credit or later mutation. $\square$

Mismatch handling and state sync. If any of the three conditions fails, $\mathcal{F}_{\mathsf{XVer}}$ (i) rejects the receive, (ii) marks the paired $\mathsf{Tid}_{\mathsf{A}} \to \mathsf{Invalid}$ if it exists, and (iii) posts the visible annotation to L_R. Because E_{zB}^{ledger} is read from C/D at finality, the relay state remains consistent with on-chain certificates. Future attempts with the same $\mathsf{Tid}_{\mathsf{A}}$ are blocked by SU.

S4. LEAKAGE POLICY AND PUBLIC VIEW

Fig. S4 specifies the public view $\mathcal{L}_{\mathsf{pub}}$: it reveals only the operation type, identifier Tid, status $\mathtt{sta}$, verification key and proof, plus headers $\big(M,$ optionally $E_k/E_z\big)$ and basic metadata/lengths. Plaintext amounts and balances are never leaked.

Public Leakage $\mathcal{L}_{\mathsf{pub}}$.
The ideal world exposes only: operation type; Tid and sta; verification keys; zk-SNARK proofs; headers $(M, E_k, \underline{E_z})$; metadata such as heights, timestamps; and lengths. No plaintext amounts or balances leak. The optional E_z is present in RECEIVE to support identity-binding.

Fig. S4. Leakage visible to the environment.

Functional form. We formalize the leakage as a function

$$\mathcal{L}_{\mathsf{pub}}: \ \mathsf{Transcript} \ \longrightarrow \ \{\mathsf{op}, \ \mathsf{Tid}, \ \mathsf{sta}, \ \mathsf{vk}, \ \mathsf{proof}, \ (M, E_k, E_z), \ \mathsf{meta}, \ \mathsf{len}\}$$

and define indistinguishability only over histories with identical $\mathcal{L}_{\mathsf{pub}}$ output (same preimage class).

S5. $\mathcal{F}_{\mathsf{VCT}^\star}$ Ideal Functionality with Visible Leakage

Fig. S5 instantiates $\mathcal{F}_{\mathsf{VCT}^\star}$. Local prechecks return `FAIL` without any public footprint. Visible outcomes are driven by $\mathcal{F}_{\mathsf{XVer}}$. For RECEIVE, pair consistency $M_{\text{rece}} = M_{\text{send}}$ and identity-binding $E_{z_B} = E_{z_B}^{\mathsf{ledger}}$ are checked by $\mathcal{F}_{\mathsf{XVer}}$ (outside the circuit). On RECEIVE/Accept, the relay annotates $\mathrm{Tid}_A \to \mathtt{Finished}$. On RECEIVE/Reject, it annotates $\mathrm{Tid}_A \to \mathtt{Invalid}$.

Ideal Functionality $\mathcal{F}_{\mathsf{VCT}^\star}$.

Internal State:
(i) Balances: for each account a, $(v[a], z[a])$.
(ii) Transaction index $Tx[\mathsf{Tid}] = \{\mathsf{kind} \in \{\mathsf{dep}, \mathsf{wd}, \mathsf{send}, \mathsf{rece}\},\ \mathsf{sta},\ \mathsf{owner},\ \mathsf{peer},\ m,\ M,\ E_k,\ \underline{E_z}\}$, where only m is private.
(iii) Status $\mathsf{sta} \in \{\mathsf{Valid}, \mathsf{Invalid}, \mathsf{Send}, \mathsf{Receive}, \mathsf{Finished}, \mathsf{Deposit}, \mathsf{Withdraw}\}$ is monotone.
(iv) **Single-use Invariant (SU):** any sending $\mathsf{Tid_A}$ transitions from Send to $\{\mathsf{Finished}, \mathsf{Invalid}\}$ at most once and no updates thereafter.

Public Leakage: only $\mathcal{L}_{\mathsf{pub}}$ (as above).

Interfaces:

DEPOSIT(a, m). Local precheck: $m > 0$ and $\exists(v^*, z^*):\ \ z^* - z[a] = m,\ v[a] - v^* = n_c m,\ v^* \geq 0$. If not, **return FAIL** (no $\mathcal{L}_{\mathsf{pub}}$). Else, allocate Tid, set $Tx[\mathsf{Tid}] = \{\mathsf{kind} = \mathsf{dep}, \mathsf{sta} = \mathsf{Valid}, \mathsf{owner} = a, m\}$. Output $\mathcal{L}_{\mathsf{pub}}(\mathsf{op} = \mathsf{dep}, \mathsf{Tid}, \mathsf{sta} = \mathsf{Valid})$. Invoke $\mathcal{F}_{\mathsf{XVer}}.\mathsf{VerifyDep}(\mathsf{Tid})$: on Accept, update $(v[a], z[a]) \leftarrow (v^*, z^*)$, set $\mathsf{sta} \leftarrow \mathsf{Deposit}$ and output $\mathcal{L}_{\mathsf{pub}}(\mathsf{op} = \mathsf{dep}, \mathsf{sta} = \mathsf{Deposit})$. On $\mathsf{Reject}(\mathsf{reason})$, set $\mathsf{sta} \leftarrow \mathsf{Invalid}$ and output.

WITHDRAW(a, m). Local precheck: $m > 0$ and $\exists(v^*, z^*):\ z[a] - z^* = m,\ v^* - v[a] = n_c m,\ z^* \geq 0$. If not, **return FAIL**. Else, allocate Tid, set $\mathsf{sta} = \mathsf{Valid}$ and output. Invoke $\mathcal{F}_{\mathsf{XVer}}.\mathsf{VerifyWd}(\mathsf{Tid})$: on Accept set $\mathsf{sta} = \mathsf{Withdraw}$, update $(v[a], z[a]) \leftarrow (v^*, z^*)$, output. On $\mathsf{Reject}(\mathsf{reason})$ set $\mathsf{sta} = \mathsf{Invalid}$, output.

SEND$(a \to b, m)$. Local precheck: $m > 0$ and $\exists z^* = z[a] - m \geq 0$. If not, **return FAIL**, else: (1) allocate sender id $\mathsf{Tid_A}$; (2) derive header $M = \mathsf{E}(m, k)$ and key encapsulation $E_k = \mathsf{EA}(\mathsf{pubk}_b, k)$; (3) set $Tx[\mathsf{Tid_A}] = \{\mathsf{kind} = \mathsf{send}, \mathsf{sta} = \mathsf{Valid}, \mathsf{owner} = a, \mathsf{peer} = b, m, M, E_k\}$ and output $\mathcal{L}_{\mathsf{pub}}(\mathsf{op} = \mathsf{send}, \mathsf{Tid_A}, \mathsf{sta} = \mathsf{Valid}, M, E_k)$; (4) invoke $\mathcal{F}_{\mathsf{XVer}}.\mathsf{VerifySend}(\mathsf{Tid_A}, M)$: on Accept set $\mathsf{sta}(\mathsf{Tid_A}) = \mathsf{Send}$, update $z[a] \leftarrow z[a] - m$, output $\mathcal{L}_{\mathsf{pub}}(\mathsf{op} = \mathsf{send}, \mathsf{Tid_A}, \mathsf{sta} = \mathsf{Send})$. On $\mathsf{Reject}(\mathsf{reason})$ set $\mathsf{sta} = \mathsf{Invalid}$ and output.

RECEIVE$(\mathsf{Tid_A})$. Local precheck: there exists a send with $Tx[\mathsf{Tid_A}].\mathsf{sta} = \mathsf{Send}$. If not, **return FAIL**, else: (1) allocate receiver id $\mathsf{Tid_B}$; (2) set $Tx[\mathsf{Tid_B}] = \{\mathsf{kind} = \mathsf{rece}, \mathsf{sta} = \mathsf{Valid}, \mathsf{owner} = \mathsf{peer}(\mathsf{Tid_A}), \mathsf{peer} = \mathsf{owner}(\mathsf{Tid_A}), m, M, E_k\}$ and output $\mathcal{L}_{\mathsf{pub}}(\mathsf{op} = \mathsf{rece}, \mathsf{Tid_B}, \mathsf{sta} = \mathsf{Valid}, M, E_k, E_z)$; (3) invoke $\mathcal{F}_{\mathsf{XVer}}.\mathsf{VerifyRece}(\mathsf{Tid_A}, \mathsf{Tid_B}, M_{\mathsf{rece}}{=}M, E_{zB})$: on Accept set $z[\mathsf{owner}(\mathsf{Tid_B})] \leftarrow z[\mathsf{owner}(\mathsf{Tid_B})] + m$, set $Tx[\mathsf{Tid_B}].\mathsf{sta} = \mathsf{Receive}$ and output. On $\mathsf{Reject}(\mathsf{reason})$ set $Tx[\mathsf{Tid_B}].\mathsf{sta} = \mathsf{Invalid}$ and mark $\mathsf{Tid_A} \to \mathsf{Invalid}$.

Fig. S5. Ideal functionality $\mathcal{F}_{\mathsf{VCT}^\star}$

S6. Concurrency and Serializability under $\mathcal{G}_{\mathsf{Ledger}}$

Concurrent executions of cross-chain transfers preserve safety and privacy without additional mechanisms. First, $\mathcal{G}_{\mathsf{Ledger}}$ exports a single totally ordered log per chain with deterministic finality. Hence, any two relay actions have a well-defined order in L_R. Second, RSM (Fig. S6) and the SU invariant operate per $\mathsf{Tid_A}$ and are idempotent: two RECEIVE annotations for the same $\mathsf{Tid_A}$ commute to the unique terminal label in $\{\mathsf{Finished}, \mathsf{Invalid}\}$. Third, $\mathcal{F}_{\mathsf{XVer}}$ reads only finalized certificates and relay entries. Thus, its decisions are functions of immutable inputs.

Lemma 2 (Serializability)**.** *Let h be any concurrent history produced by the real protocol (or $\mathcal{F}_{\mathsf{VCT}^\star}$ with $\mathcal{S}$). Then there exists a permutation π that orders all visible relay steps by their ledger-finalized positions such that executing the steps sequentially in order π yields the same public transcript and the same $\mathcal{F}_{\mathsf{VCT}^\star}$ state.*

Proof. Because L_R is totally ordered at finality, ordering by finality induces a legal sequential schedule. Per-$\mathsf{Tid_A}$ single-use prevents write-write races on the same entry. Different Tid values touch disjoint state except for reads of finalized certificates. Thus, operations on different Tid commute. □

Privacy under concurrency. The hybrids are defined over the entire transcript, independent of interleaving. Replacing proofs and swapping ciphertexts commute with merging histories. Therefore, the privacy theorems remain valid for arbitrary concurrency.

S7. Relay State Machine: Visible Actions and Invariant

Fig. S6 shows the relay state machine per sender identifier Tid_A: a single terminal transition $\mathtt{Send} \to \{\mathtt{Finished}, \mathtt{Invalid}\}$ enforces the single-use invariant SU. A RECEIVE/Accept triggers $\mathtt{Finished}$, and a RECEIVE/Reject triggers $\mathtt{Invalid}$.

Non-terminal retries are not permitted. Concurrent visible steps are ordered by finality on L_R, and relay writes are idempotent post-finality.

Relay State Machine RSM **(publicly visible).**
State per $\mathsf{Tid_A}$: $\perp \xrightarrow{\mathsf{Send}} \mathsf{Send} \xrightarrow{\mathsf{Receive/Reject}} \{\mathsf{Finished}, \mathsf{Invalid}\}$.
Visible actions:
(1) $\mathsf{Post\_R}(\mathsf{Tid_A}, \mathsf{Send}, M)$; (2) $\mathsf{Annotate\_R}(\mathsf{Tid_A}, \mathsf{Finished})$; (3) $\mathsf{Annotate\_R}(\mathsf{Tid_A}, \mathsf{Invalid})$; (4) $\mathsf{Read\_R}(\mathsf{Tid_A})$.
SU: For any $\mathsf{Tid_A}$, at most one transition $\mathsf{Send} \to \{\mathsf{Finished}, \mathsf{Invalid}\}$, and no updates afterwards.
Enforcement: $\mathcal{G}_{\mathsf{Ledger}}$ provides deterministic finality. RSM writes are idempotent and reject any second post-finality update for the same $\mathsf{Tid_A}$.
Note: DEPOSIT, WITHDRAW, and RECEIVE also issue $\mathsf{Post\_R}$ events (Fig. S3) but they do not affect SU, which is defined per-sender $\mathsf{Tid_A}$.

Fig. S6. Relay state machine and single-consumption invariant.

S8. SECURITY MODEL AND ASSUMPTIONS

- **Adversary & Corruption.** The adversary $\mathcal{A}$ is PPT and corruption is static. The environment $\mathcal{Z}$ supplies inputs and observes outputs restricted to $\mathcal{L}_{\mathsf{pub}}$.
- **Ledgers.** $\mathcal{G}_{\mathsf{Ledger}}$ offers deterministic finality and safety: the adversary may delay delivery but cannot rewrite finalized states nor create conflicting finalized histories.
- **CRS/ZK.** $\mathcal{F}_{\mathsf{CRS}}$ samples CRS. zk-SNARKs are complete, knowledge-sound, and zero-knowledge with a simulator trapdoor available to $\mathcal{S}$ only.
- **CrossVerify.** $\mathcal{F}_{\mathsf{XVer}}$ mediates all cross-chain decisions. $\mathsf{Invalid}$ is triggered only by $\mathcal{F}_{\mathsf{XVer}}$'s Reject, while $\mathcal{F}_{\mathsf{VCT^*}}$ emits the public Lpub reflecting the decision. $\mathcal{F}_{\mathsf{XVer}}$ may delay but not violate ledger safety or SU, which relies on $\mathcal{G}_{\mathsf{Ledger}}$).
- **Primitives.** Signatures are EUF-CMA. Encryption EA is IND-CPA with unique decodability. Payload encryption E is IND-CPA. Commitments are binding and hiding.
- **Scope Declaration.** Joint attacks that combine zk-proof forgery with collusive ledger rewriting are out of scope: $\mathcal{G}_{\mathsf{Ledger}}$ already rules out any alteration of finality or fabrication of conflicting finalized states. Cross-chain node collusion can at most cause delays/withholding, not ledger tampering.

S9. REAL PROTOCOL Π: STATEMENT–WITNESS RELATIONS

Each operation has a zk-SNARK C_i with public inputs (headers/keys/status) and private witness (v, z, v^*, z^*, m, k). The relations are composed of balance/amount constraints and ciphertext-binding constraints:
Balance/amount constraints.

$$\begin{aligned}
\text{DEPOSIT: } & z^* - z = m, \quad v - v^* = n_c m,\ v^* \geq 0, \\
\text{WITHDRAW: } & z - z^* = m, \quad v^* - v = n_c m,\ z^* \geq 0, \\
\text{SEND: } & z_A^* = z_A - m,\ m > 0,\ z_A^* \geq 0, \\
\text{RECEIVE: } & z_B^* - z_B = m.
\end{aligned}$$

Ciphertext-binding constraints.

$$\begin{aligned}
\text{DEPOSIT: } & E_z = \mathsf{EA}(z, \mathsf{pubk}), \quad E_z^* = \mathsf{EA}(z^*, \mathsf{pubk}), \quad E_v = \mathsf{EA}(v, \mathsf{pubk}), \quad E_v^* = \mathsf{EA}(v^*, \mathsf{pubk}); \\
\text{WITHDRAW: } & E_z = \mathsf{EA}(z, \mathsf{pubk}), \quad E_z^* = \mathsf{EA}(z^*, \mathsf{pubk}), \quad E_v = \mathsf{EA}(v, \mathsf{pubk}), \quad E_v^* = \mathsf{EA}(v^*, \mathsf{pubk}); \\
\text{SEND: } & M = \mathsf{E}(m, k), \quad E_{zA} = \mathsf{EA}(z_A, \mathsf{pubk}_A), \quad E_{zA}^* = \mathsf{EA}(z_A^*, \mathsf{pubk}_A); \\
\text{RECEIVE: } & M_{\mathsf{rece}} = \mathsf{E}(m, k), \quad E_{zB} = \mathsf{EA}(z_B, \mathsf{pubk}_B), \quad E_{zB}^* = \mathsf{EA}(z_B^*, \mathsf{pubk}_B).
\end{aligned}$$

In particular, the pair-consistency $M_{\mathsf{rece}} = M_{\mathsf{send}}$ and the identity-binding predicate $E_{z_B} = E_{z_B}^{\mathsf{ledger}}$ are **not** part of the zk-circuit. They are checked by $\mathcal{F}_{\mathsf{XVer}}$ during RECEIVE.

S9.1 Adversarial Scheduling and Conflict Handling in $\mathcal{S}$

State kept by $\mathcal{S}$. (i) A map $T[\mathsf{Tid}]$ to the latest visible relay status; (ii) a de-dup set Q of processed query identifiers; (iii) a buffer B of pending adversarial posts with delivery times.
Delay & reordering. $\mathcal{S}$ forwards adversarial posts to $\mathcal{G}_{\mathsf{Ledger}}$ through the external scheduling interface and may delay/reorder them arbitrarily before finality, but it must not change the order of finalized items returned by $\mathcal{G}_{\mathsf{Ledger}}$. Reads are answered by polling L_X until the requested entry reaches finality (or timeouts dictated by $\mathcal{Z}$), then emitting only $\mathcal{L}_{\mathsf{pub}}$.

Replay/duplication. On any adversarial replay with the same Tid or query identifier, $\mathcal{S}$ responds with the already visible outcome, ensuring idempotence. For RECEIVE, a second annotation for the same Tid_A is rejected and $T[\mathsf{Tid}_\mathsf{A}]$ remains unchanged by SU.
Conflicting transactions. If the adversary submits conflicting SEND transactions with the same Tid_A, $\mathcal{S}$ relays both to $\mathcal{G}_{\mathsf{Ledger}}$, but only the unique finalized one is used by $\mathcal{F}_{\mathsf{XVer}}$. If two conflicts were finalized (ruled out by $\mathcal{G}_{\mathsf{Ledger}}$'s safety), the reduction in Sec. S12 would extract a violating accepting proof. For RECEIVE, if $(M_{\mathsf{rece}}, E_{zB})$ conflicts with the paired send or with the ledger certificate, $\mathcal{S}$ invokes $\mathcal{F}_{\mathsf{XVer}}$ which rejects and marks $\mathsf{Tid}_\mathsf{A} \to \mathsf{Invalid}$.
$\mathcal{G}_{\mathsf{Ledger}}$–$\mathcal{F}_{\mathsf{VCT}^\star}$ synchronization. $\mathcal{S}$ mirrors every finalized relay action into $\mathcal{F}_{\mathsf{VCT}^\star}$'s internal $Tx[\cdot]$ and status map before emitting the corresponding $\mathcal{L}_{\mathsf{pub}}$ item. Thus, the visible transcript and the ideal state remain consistent. Any subsequent environment query is answered from T and the finalized L_R.

We encode the balance updates and ciphertext-binding relations in a standard R1CS form. Any IND-CPA PKE and symmetric encryption scheme can be used as long as the public inputs expose key-consistency.

S10. Detailed Simulator $\mathcal{S}$ in the Ideal World

Fig. S7 outlines the ideal-world simulator $\mathcal{S}$. It emits only $\mathcal{L}_{\mathsf{pub}}$, delegates cross-chain decisions to $\mathcal{F}_{\mathsf{XVer}}$, and mirrors finalized relay annotations. On RECEIVE/Reject, $\mathcal{S}$ records $Tx_{\mathrm{rece}} = \mathtt{Invalid}$ and annotates the paired $\mathrm{Tid}_A \to \mathtt{Invalid}$. On RECEIVE/Accept, it annotates $\mathrm{Tid}_A \to \mathtt{Finished}$ and outputs the updated relay/account records.

Simulator $\mathcal{S}$.
World: $(\mathcal{G}_{\mathsf{Ledger}}, \mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{XVer}}, \mathcal{F}_{\mathsf{VCT}^\star})$-hybrid. $\mathcal{S}$ controls the adversarial interface and produces only $\mathcal{L}_{\mathsf{pub}}$.

On DEPOSIT/WITHDRAW from a corrupt party: perform the same local precheck as $\mathcal{F}_{\mathsf{VCT}^\star}$. On failure, return FAIL (no output). On success, register Valid in $\mathcal{F}_{\mathsf{VCT}^\star}$. Simulate an accepting zk-proof via $\mathcal{F}_{\mathsf{CRS}}$ and emit $\mathcal{L}_{\mathsf{pub}}(\mathsf{sta} = \mathsf{Valid})$. Then call $\mathcal{F}_{\mathsf{XVer}}$.VerifyDep/Wd and emit the resulting terminal state (Deposit/Withdraw or Invalid).

On SEND: precheck or FAIL. Otherwise register Tid_A with $\mathsf{sta} = \mathsf{Valid}$, emit $\mathcal{L}_{\mathsf{pub}}$ with simulated proof. Call $\mathcal{F}_{\mathsf{XVer}}$.VerifySend. On Accept, set $\mathsf{sta} = \mathsf{Send}$, decrease $z[a]$, emit. on Reject, set Invalid.

On RECEIVE(Tid_A,E_{zB}): require $Tx[\mathsf{Tid}_\mathsf{A}].\mathsf{sta} = \mathsf{Send}$. Else FAIL. Register Tid_B with $\mathsf{sta} = \mathsf{Valid}$, include E_{zB} in Lpub, emit. Call $\mathcal{F}_{\mathsf{XVer}}$.VerifyRece. On Accept, credit receiver, set $\mathsf{sta}(\mathsf{Tid}_\mathsf{B}) = \mathsf{Receive}$ and rely on $\mathcal{F}_{\mathsf{XVer}}$ to annotate $\mathsf{Tid}_\mathsf{A} \to \mathsf{Finished}$. On Reject, set $\mathsf{sta}(\mathsf{Tid}_\mathsf{B}) = \mathsf{Invalid}$ and annotate $\mathsf{Tid}_\mathsf{A} \to \mathsf{Invalid}$.

SU: $\mathcal{S}$ rejects any second annotation after finality, keeping $\mathsf{Send} \to \{\mathsf{Finished}, \mathsf{Invalid}\}$ single-consumption.

Fig. S7. Simulator behavior and visible actions.

S11. Hybrid Sequence for the Main UC Theorem

We use a four-step sequence. Let $\mathcal{Z}$ be any PPT environment and $\mathcal{A}$ any PPT real-world a dversary.
H_0 **(Real).** Real execution of Π with $\mathcal{A}$ in the $(\mathcal{G}_{\mathsf{Ledger}}, \mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{XVer}})$-hybrid model.
H_1 **(Witness-Stripping via ZK).** Replace every accepting zk-SNARK proof in H_0 by a simulated proof for the same public statement using the trapdoor from $\mathcal{F}_{\mathsf{CRS}}$. Witnesses are never used.

Lemma 3 ($\mathsf{H}_0 \overset{c}{\approx} \mathsf{H}_1$)**.** *By zk-SNARK zero-knowledge, $\mathcal{Z}$ cannot distinguish H_0 from H_1 except with* negl *advantage.*

H_2 **(Cipher/Commit Swap).** Independently and operation-wise, replace each amount/auxiliary ciphertext (and balance commitments if any) by encryptions/commitments of length-matching dummies.

Lemma 4 ($\mathsf{H}_1 \overset{c}{\approx} \mathsf{H}_2$)**.** *By IND-CPA and commitment hiding, $\mathcal{Z}$ cannot distinguish H_1 from H_2 except with* negl *advantage.*

H_3 **(Ideal + Simulator).** Replace the real protocol by the simulator $\mathcal{S}$ of Fig. S7 interacting with $\mathcal{F}_{\mathsf{VCT}^\star}$ and $\mathcal{F}_{\mathsf{XVer}}$ and emulating the relay via its visible actions. $\mathcal{S}$ outputs exactly $\mathcal{L}_{\mathsf{pub}}$ and enforces SU via RSM.

Lemma 5 ($\mathsf{H}_2 \overset{c}{\approx} \mathsf{H}_3$)**.** *The public distribution in H_2 equals that in H_3: both expose only $\mathcal{L}_{\mathsf{pub}}$ and identical status transitions. Any deviation would either violate SU, contradicting Fig. S6 with $\mathcal{G}_{\mathsf{Ledger}}$ finality, or create an accepting proof without a valid statement, contradicting soundness.*

Lemma 6 (Bridge Lemma: Ledger $\to$ Atomicity)**.** *If $\mathcal{G}_{\mathsf{Ledger}}$ provides deterministic finality and the relay state machine enforces SU, then for any Tid_A the public transcript admits no observable intermediate state between Send and $\{\mathsf{Finished}, \mathsf{Invalid}\}$. Hence, the transition is atomic and Tid_A cannot be consumed twice.*

Theorem 1 (Main UC Theorem: Verifiability & Balance Safety)**.** *For every PPT $\mathcal{A}$ there exists a PPT $\mathcal{S}$ such that $\mathsf{EXEC}^{\mathsf{real}}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}^{\mathsf{ideal}}_{\mathcal{F}_{\mathsf{VCT}^\star},\mathcal{F}_{\mathsf{XVer}},\mathcal{S},\mathcal{Z}}$ in the $(\mathcal{G}_{\mathsf{Ledger}}, \mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{XVer}})$-hybrid model.*

Proof. Chain the lemmas: $\mathsf{H}_0 \overset{c}{\approx} \mathsf{H}_1 \overset{c}{\approx} \mathsf{H}_2 \overset{c}{\approx} \mathsf{H}_3$. H_3 is the ideal execution with $\mathcal{S}$ talking to $\mathcal{F}_{\mathsf{VCT}^\star}$ and $\mathcal{F}_{\mathsf{XVer}}$. Verifiability follows from soundness (no false acceptance), and balance safety from per-operation relations and SU (no double-consumption), strengthened by the Bridge Lemma. □

S12. Reduction Construction and Probability Quantification

We formalize the stepwise reduction that captures overspend/false-accept as a break of zk-SNARK soundness.

Lemma 7 (Soundness-Reduction Lemma)**.** *Let $\mathcal{A}$ be any PPT adversary that, in the real world, causes an accepting transaction that violates the corresponding relation (overspend, balance underflow, or acceptance of a mismatched Send/Receive pair). Consider a single-circuit SNARK soundness game for $C_j \in \{\mathsf{dep}, \mathsf{wd}, \mathsf{send}, \mathsf{rece}\}$, and let the reduction $\mathcal{R}$ guess j uniformly at random. Then*

$$\Pr\big[\mathcal{R} \text{ breaks SNARK}\big] \ \geq\ \tfrac{1}{4}\Pr\big[E_{\mathsf{over}} \ \vee\ E_{\mathsf{false}}\big] \ -\ \mathsf{negl}(\lambda). \tag{1}$$

Proof. $\mathcal{R}$ joins the soundness game for a randomly chosen circuit C_j under CRS set by the challenger. It sets the system CRS to this value and runs $\mathcal{A}$ unchanged. It emulates ledgers via $\mathcal{G}_{\mathsf{Ledger}}$ and keeps shadow balances from $\mathcal{L}_{\mathsf{pub}}$. Whenever $\mathcal{A}$ outputs a finalized accepting transaction (x, π) of type C_j:

- If it triggers E_{over}, i.e., a relation violation occurs, output (x, π).
- If it triggers E_{false}, i.e., the Send/Receive pair is inconsistent, output (x, π).

If the violating transaction is for circuit C_i, $\mathcal{R}$ succeeds exactly when $j = i$, which occurs with probability $1/4$. Conditioning on the bad event and averaging over the guess yields the factor $1/4$ in (1). Repeated consumption either violates SU (absorbed in negl) or implies an inconsistent accepting proof covered above. □

S13. Privacy Theorems

We write $\mathsf{View}^{\mathsf{H}_i}$ for the environment's view in H_i, and all indistinguishability arguments below are taken with respect to the same hybrid sequence.

S13.1 Quantitative Advantage Accounting

Let $\Delta_i := \big|\Pr[\mathsf{View}^{\mathsf{H}_i} \in \mathcal{W}] - \Pr[\mathsf{View}^{\mathsf{H}_{i+1}} \in \mathcal{W}]\big|$ for any distinguisher/event $\mathcal{W}$. By the triangle inequality,

$$\big|\Pr[\mathsf{View}^{\mathsf{H}_0} \in \mathcal{W}] - \Pr[\mathsf{View}^{\mathsf{H}_3} \in \mathcal{W}]\big| \ \leq\ \sum_{i=0}^{2} \Delta_i.$$

We now bound each Δ_i by a standard reduction:

$$\begin{aligned}
&\Delta_0 \leq \mathrm{Adv}_{\Pi}^{\mathrm{ZK}}(\lambda) && \text{(witness-stripping via the ZK simulator)},\\
&\Delta_1 \leq \mathrm{Adv}_{\mathsf{E}}^{\mathrm{IND\text{-}CPA}}(\lambda) && \text{(cipher/commit swap for amounts and auxiliaries)},\\
&\Delta_2 \leq \mathsf{negl}(\lambda) && \text{(syntactic replacement with the simulator that outputs only } \mathcal{L}_{\mathsf{pub}}).
\end{aligned}$$

Therefore the privacy advantage of any PPT distinguisher $\mathcal{D}$ is at most

$$\mathrm{Adv}_{\Pi}^{\mathsf{priv}}(\lambda) \ \leq\ \mathrm{Adv}_{\Pi}^{\mathrm{ZK}}(\lambda) \ +\ \mathrm{Adv}_{\mathsf{E}}^{\mathrm{IND\text{-}CPA}}(\lambda) \ +\ \mathsf{negl}(\lambda).$$

For balance privacy replace the swapped components accordingly, namely balances and certificates, yielding the same bound.

S13.2 Amount Privacy

Definition 1 (Amount Privacy)**.** *Let $\mathsf{View}^{\mathsf{ideal\_amt}}$ be the distribution that replaces all plaintext amounts by dummies while preserving $\mathcal{L}_{\mathsf{pub}}$. Π has amount privacy if $\mathsf{View}^{\mathsf{real}} \overset{c}{\approx} \mathsf{View}^{\mathsf{ideal\_amt}}$.*

Theorem 2 (Amount Privacy)**.** $\mathsf{View}^{\mathsf{H}_0} \overset{c}{\approx} \mathsf{View}^{\mathsf{H}_1}$ *by ZK (witness-stripping), and* $\mathsf{View}^{\mathsf{H}_1} \overset{c}{\approx} \mathsf{View}^{\mathsf{H}_2}$ *by IND-CPA. Hence,* $\mathsf{View}^{\mathsf{H}_0} \overset{c}{\approx} \mathsf{View}^{\mathsf{H}_2}$. *Since* H_3 *only emits* $\mathcal{L}_{\mathsf{pub}}$, $\mathsf{View}^{\mathsf{H}_2} \overset{c}{\approx} \mathsf{View}^{\mathsf{H}_3}$. *Moreover,*

$$\mathrm{Adv}_{\Pi}^{\textit{amt-priv}} \ \leq\ \mathrm{Adv}^{\textit{ZK}} \ +\ \mathrm{Adv}^{\textit{IND-CPA}} \ +\ \mathsf{negl}(\lambda).$$

S13.3 Balance Privacy

Definition 2 (Balance Privacy)**.** *Define $\mathsf{View}^{\mathsf{ideal\_bal}}$ by replacing pre/post balances or their commitments with dummies while preserving $\mathcal{L}_{\mathsf{pub}}$. Π has balance privacy if $\mathsf{View}^{\mathsf{real}} \overset{c}{\approx} \mathsf{View}^{\mathsf{ideal\_bal}}$.*

Theorem 3 (Balance Privacy)**.** *ZK removes witness dependence ($\mathsf{H}_0 \to \mathsf{H}_1$), and hiding (or IND-CPA if encrypted) swaps balances/commitments ($\mathsf{H}_1 \to \mathsf{H}_2$), while $\mathsf{H}_2 \to \mathsf{H}_3$ leaves only $\mathcal{L}_{\mathsf{pub}}$. Moreover,*

$$\mathrm{Adv}_{\Pi}^{\textit{bal-priv}} \ \leq\ \mathrm{Adv}^{\textit{ZK}} \ +\ \big(\mathrm{Adv}^{\textit{Hiding}} \textit{ or } \mathrm{Adv}^{\textit{IND-CPA}}\big) \ +\ \mathsf{negl}(\lambda).$$

S13.4 Fund-Correlation Privacy

Definition 3 (Fund-Correlation Privacy)**.** *For two histories Q_0, Q_1 that agree on $\mathcal{L}_{\mathsf{pub}}$ and on the multiset of hidden amounts and balance deltas but differ in pairings across users/time, the joint views are computationally indistinguishable.*

Theorem 4 (Fund-Correlation Privacy)**.** *In H_1 witnesses are removed. In H_2 ciphertexts/commitments are replaced by dummies that preserve only lengths. Pair-consistency $M_{\mathsf{rece}} = M_{\mathsf{send}}$ is enforced by $\mathcal{F}_{\mathsf{XVer}}$ from the public transcript, thus $\mathsf{View}^{\mathsf{H}_2}$ and $\mathsf{View}^{\mathsf{H}_3}$ remain identical for any re-pairing preserving $\mathcal{L}_{\mathsf{pub}}$. Therefore $\mathsf{View}^{\mathsf{H}_0}$ is indistinguishable. Moreover,*

$$\mathrm{Adv}_{\Pi}^{\textit{fund-corr}} \;\leq\; \mathrm{Adv}^{\textit{ZK}} \;+\; \mathrm{Adv}^{\textit{IND-CPA}} \;+\; \mathsf{negl}(\lambda).$$