

# 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 material provides a Universal Composability (UC) treatment that unifies transaction-security (verifiability, balance safety, no double-spending) and privacy (amount, balance, fund-correlation) using an ideal functionality  $\mathcal{F}_{VCT^*}$ . In  $\mathcal{F}_{VCT^*}$ , local prechecks return FAIL without any public footprint. Invalid outcomes are triggered by the result of a cross-chain verification service  $\mathcal{F}_{XVer}$ . Transfers follow a single-consumption relay state machine  $\text{Send} \rightarrow \{\text{Finished}, \text{Invalid}\}$ . 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 $(\mathcal{G}_{\text{Ledger}}, \mathcal{F}_{\text{CRS}}, \mathcal{F}_{XVer})$ -HYBRID WORLD

Fig. S1 summarizes  $\mathcal{G}_{\text{Ledger}}$  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**  $\mathcal{G}_{\text{Ledger}}$ .

**State:** For each chain  $X \in \{C, D, R\}$ , a totally ordered log  $L_X$  with deterministic finality.

**Interfaces:**  $\text{Post}(X, \text{tx})$  appends to  $L_X$  and finalizes.  $\text{Read}(X, \text{qid})$  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  $\mathcal{G}_{\text{Ledger}}$ .

Fig. S2 depicts  $\mathcal{F}_{\text{CRS}}$ , 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  $\mathcal{F}_{\text{CRS}}$ .

**CRS Functionality**  $\mathcal{F}_{\text{CRS}}$ .

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

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

### Cross-Chain Verification Functionality $\mathcal{F}_{XVer}$ (CrossVerify).

*Goal:* Mediate all cross-chain checks. Its *Accept/Reject* result triggers whether  $\mathcal{F}_{VCT^*}$  will publicly output *Invalid*.

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

#### Interfaces:

**VerifyDep(Tid):** Fetch  $Tx$  by Tid from source ledger. If  $Tx.\text{sta} \neq \text{Valid}$  return *Reject*. Parse  $\chi = \{\text{pubk}, E_z, E_z^*, E_v, E_v^*\}$  and proof  $\pi_{\text{dep}}$ . Compute  $r \leftarrow \Pi.\text{Verify}(\text{vk}_{\text{dep}}, \chi, \pi_{\text{dep}})$ . If  $r = 1$  then  $\text{Post\_R}(R_{\text{AccN}}, Tx)$  to  $L_R$  and return *Accept*, else *Reject*.

**VerifyWd(Tid):** Same with  $\chi = \{\text{pubk}, E_z, E_z^*, E_v, E_v^*\}$  and  $\pi_{\text{wd}}$ . On  $r = 1$   $\text{Post\_R}(R_{\text{AccN}}, Tx)$ .

**VerifySend(Tid<sub>A</sub>, M):** Parse  $\chi = \{\text{pubk}_A, E_{zA}, E_{zA}^*, M\}$  and  $\pi_{\text{send}}$ . Compute  $r \leftarrow \Pi.\text{Verify}(\text{vk}_{\text{send}}, \chi, \pi_{\text{send}})$ . If  $r = 1$  set  $Tx.\text{sta} = \text{Send}$  and  $\text{Post\_R}(Tx_{\text{send}})$  to  $L_R$ , else *Reject*.

**VerifyRece(Tid<sub>A</sub>, Tid<sub>B</sub>, M<sub>rece</sub>, E<sub>zB</sub>):** Parse  $\chi = \{\text{pubk}_B, E_{zB}, E_{zB}^*, M_{\text{rece}}\}$  and  $\pi_{\text{rece}}$ . Obtain  $Tx_{\text{send}}$  via Tid<sub>A</sub> from  $L_R$  and parse  $M_{\text{send}}$  and  $\text{sta}$ . Obtain  $E_{zB}^{\text{ledger}}$  from chain D's account certificate. Accept iff  $\Pi.\text{Verify}(\text{vk}_{\text{rece}}, \chi, \pi_{\text{rece}}) = 1 \wedge M_{\text{rece}} = M_{\text{send}} \wedge E_{zB} = E_{zB}^{\text{ledger}} \wedge \text{sta} = \text{Send}$ . On reject: set  $\text{sta}_{\text{send}} = \text{Invalid}$  and  $\text{Post\_R}(Tx_{\text{send}})$ . Return *Reject*. On accept: set  $\text{sta}_{\text{send}} = \text{Finished}$ ,  $\text{Post\_R}(Tx_{\text{send}})$  and  $\text{Post\_R}(R_{\text{AccN}}, Tx_{\text{rece}})$ . Return *Accept*.

**Remarks:** All four interfaces perform a visible  $\text{Post\_R}$  on *Accept*. *RECEIVE* additionally annotates Tid<sub>A</sub> as *Finished/Invalid*.

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

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

Fig. S3 summarizes the CrossVerify interfaces and their visible actions. The two explicit checks in *VerifyRece*, namely  $M_{\text{rece}} = M_{\text{send}}$  and  $E_{zB} = E_{zB}^{\text{ledger}}$  with  $\text{sta} = \text{Send}$ , 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  $\mathcal{G}_{\text{Ledger}}$  provides deterministic finality. If  $\mathcal{F}_{XVer}$  accepts a RECEIVE with  $\Pi.\text{Verify}(\text{vk}_{\text{rece}}, \chi, \pi_{\text{rece}}) = 1$ ,  $M_{\text{rece}} = M_{\text{send}}$ ,  $E_{zB} = E_{zB}^{\text{ledger}}$  and the paired send has  $\text{sta} = \text{Send}$ , then (i) the credited receiver is the ledger owner of  $E_{zB}^{\text{ledger}}$  (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:  $M_{\text{rece}} = E(m, k)$ ,  $E_{zB} = EA(z_B, \text{pubk}_B)$ ,  $E_{zB}^* = EA(z_B^*, \text{pubk}_B)$ , and the balance relation  $z_B^* - z_B = m$ . Equality  $M_{\text{rece}} = M_{\text{send}}$  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  $E_{zB} = E_{zB}^{\text{ledger}}$  ties  $\text{pubk}_B$  to the ledger-certified receiver account. Forging a different identity would require either (a) ledger tampering (ruled out by  $\mathcal{G}_{\text{Ledger}}$ ), or (b) a second accepting proof with different public key but the same  $E_{zB}^{\text{ledger}}$ , which violates the circuit's key-consistency. Finally,  $\text{sta} = \text{Send}$  plus the relay SU invariant forces a single post-finality transition of Tid<sub>A</sub> to  $\{\text{Finished}, \text{Invalid}\}$ , precluding double credit or later mutation.  $\square$

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

## S4. LEAKAGE POLICY AND PUBLIC VIEW

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

#### Public Leakage $\mathcal{L}_{\text{pub}}$ .

The ideal world exposes only: operation type; Tid and  $\text{sta}$ ; verification keys; zk-SNARK proofs; headers  $(M, E_k, E_z)$ ; 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

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

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

## S5. $\mathcal{F}_{VCT^*}$ IDEAL FUNCTIONALITY WITH VISIBLE LEAKAGE

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

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

**Internal State:**

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

**Public Leakage:** only  $\mathcal{L}_{\text{pub}}$  (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  $\mathcal{L}_{\text{pub}}$ ). Else, allocate Tid, set  $Tx[Tid] = \{\text{kind} = \text{dep}, \text{sta} = \text{Valid}, \text{owner} = a, m\}$ . Output  $\mathcal{L}_{\text{pub}}(\text{op} = \text{dep}, \text{Tid}, \text{sta} = \text{Valid})$ . Invoke  $\mathcal{F}_{XVer}.\text{VerifyDep}(\text{Tid})$ : on Accept, update  $(v[a], z[a]) \leftarrow (v^*, z^*)$ , set  $\text{sta} \leftarrow \text{Deposit}$  and output  $\mathcal{L}_{\text{pub}}(\text{op} = \text{dep}, \text{sta} = \text{Deposit})$ . On Reject(reason), set  $\text{sta} \leftarrow \text{Invalid}$  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  $\text{sta} = \text{Valid}$  and output. Invoke  $\mathcal{F}_{XVer}.\text{VerifyWd}(\text{Tid})$ : on Accept set  $\text{sta} = \text{Withdraw}$ , update  $(v[a], z[a]) \leftarrow (v^*, z^*)$ , output. On Reject(reason) set  $\text{sta} = \text{Invalid}$ , output.

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

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

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

## S6. CONCURRENCY AND SERIALIZABILITY UNDER $\mathcal{G}_{\text{Ledger}}$

Concurrent executions of cross-chain transfers preserve safety and privacy without additional mechanisms. First,  $\mathcal{G}_{\text{Ledger}}$  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  $Tid_A$  and are idempotent: two RECEIVE annotations for the same  $Tid_A$  commute to the unique terminal label in  $\{\text{Finished}, \text{Invalid}\}$ . Third,  $\mathcal{F}_{XVer}$  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  $\mathcal{F}_{VCT^*}$  with  $\mathcal{S}$ ). Then there exists a permutation  $\pi$  that orders all visible relay steps by their ledger-finalized positions such that executing the steps sequentially in order  $\pi$  yields the same public transcript and the same  $\mathcal{F}_{VCT^*}$  state.*

*Proof.* Because  $L_R$  is totally ordered at finality, ordering by finality induces a legal sequential schedule. Per- $Tid_A$  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.  $\square$

**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  $\text{Send} \rightarrow \{\text{Finished}, \text{Invalid}\}$  enforces the single-use invariant SU. A RECEIVE/Accept triggers  $\text{Finished}$ , and a RECEIVE/Reject triggers  $\text{Invalid}$ .

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 Tid<sub>A</sub>:*  $\perp \xrightarrow{\text{Send}} \text{Send} \xrightarrow{\text{Receive/Reject}} \{\text{Finished}, \text{Invalid}\}$ .

**Visible actions:**

- (1) Post\_R(Tid<sub>A</sub>, Send, M); (2) Annotate\_R(Tid<sub>A</sub>, Finished); (3) Annotate\_R(Tid<sub>A</sub>, Invalid); (4) Read\_R(Tid<sub>A</sub>).

**SU:** For any Tid<sub>A</sub>, at most one transition Send → {Finished, Invalid}, and no updates afterwards.

**Enforcement:**  $\mathcal{G}_{\text{Ledger}}$  provides deterministic finality. RSM writes are idempotent and reject any second post-finality update for the same Tid<sub>A</sub>.

*Note:* DEPOSIT, WITHDRAW, and RECEIVE also issue Post\_R events (Fig. S3) but they do not affect SU, which is defined per-sender Tid<sub>A</sub>.

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  $\mathcal{L}_{\text{pub}}$ .
- **Ledgers.**  $\mathcal{G}_{\text{Ledger}}$  offers deterministic finality and safety: the adversary may delay delivery but cannot rewrite finalized states nor create conflicting finalized histories.
- **CRS/ZK.**  $\mathcal{F}_{\text{CRS}}$  samples CRS. zk-SNARKs are complete, knowledge-sound, and zero-knowledge with a simulator trapdoor available to  $\mathcal{S}$  only.
- **CrossVerify.**  $\mathcal{F}_{\text{XVer}}$  mediates all cross-chain decisions. Invalid is triggered only by  $\mathcal{F}_{\text{XVer}}$ 's Reject, while  $\mathcal{F}_{\text{VCT}^*}$  emits the public Lpub reflecting the decision.  $\mathcal{F}_{\text{XVer}}$  may delay but not violate ledger safety or SU, which relies on  $\mathcal{G}_{\text{Ledger}}$ .
- **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:  $\mathcal{G}_{\text{Ledger}}$  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 II: 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.**

$$\text{DEPOSIT: } z^* - z = m, \quad v - v^* = n_c m, \quad v^* \geq 0,$$

$$\text{WITHDRAW: } z - z^* = m, \quad v^* - v = n_c m, \quad z^* \geq 0,$$

$$\text{SEND: } z_A^* = z_A - m, \quad m > 0, \quad z_A^* \geq 0,$$

$$\text{RECEIVE: } z_B^* - z_B = m.$$

**Ciphertext-binding constraints.**

$$\text{DEPOSIT: } E_z = \text{EA}(z, \text{pubk}), \quad E_z^* = \text{EA}(z^*, \text{pubk}), \quad E_v = \text{EA}(v, \text{pubk}), \quad E_v^* = \text{EA}(v^*, \text{pubk});$$

$$\text{WITHDRAW: } E_z = \text{EA}(z, \text{pubk}), \quad E_z^* = \text{EA}(z^*, \text{pubk}), \quad E_v = \text{EA}(v, \text{pubk}), \quad E_v^* = \text{EA}(v^*, \text{pubk});$$

$$\text{SEND: } M = \text{E}(m, k), \quad E_{zA} = \text{EA}(z_A, \text{pubk}_A), \quad E_{zA}^* = \text{EA}(z_A^*, \text{pubk}_A);$$

$$\text{RECEIVE: } M_{\text{rece}} = \text{E}(m, k), \quad E_{zB} = \text{EA}(z_B, \text{pubk}_B), \quad E_{zB}^* = \text{EA}(z_B^*, \text{pubk}_B).$$

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

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

**State kept by  $\mathcal{S}$ .** (i) A map  $T[\text{Tid}]$  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  $\mathcal{G}_{\text{Ledger}}$  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  $\mathcal{G}_{\text{Ledger}}$ . Reads are answered by polling  $L_X$  until the requested entry reaches finality (or timeouts dictated by  $\mathcal{Z}$ ), then emitting only  $\mathcal{L}_{\text{pub}}$ .

**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<sub>A</sub> is rejected and  $T[\text{Tid}_A]$  remains unchanged by SU.

**Conflicting transactions.** If the adversary submits conflicting SEND transactions with the same Tid<sub>A</sub>,  $\mathcal{S}$  relays both to  $\mathcal{G}_{\text{Ledger}}$ , but only the unique finalized one is used by  $\mathcal{F}_{\text{XVer}}$ . If two conflicts were finalized (ruled out by  $\mathcal{G}_{\text{Ledger}}$ 's safety), the reduction in Sec. S12 would extract a violating accepting proof. For RECEIVE, if  $(M_{\text{rece}}, E_{zB})$  conflicts with the paired send or with the ledger certificate,  $\mathcal{S}$  invokes  $\mathcal{F}_{\text{XVer}}$  which rejects and marks  $\text{Tid}_A \rightarrow \text{Invalid}$ .

**$\mathcal{G}_{\text{Ledger}}\text{-}\mathcal{F}_{\text{VCT}^*}$  synchronization.**  $\mathcal{S}$  mirrors every finalized relay action into  $\mathcal{F}_{\text{VCT}^*}$ 's internal  $Tx[\cdot]$  and status map before emitting the corresponding  $\mathcal{L}_{\text{pub}}$  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  $\mathcal{L}_{\text{pub}}$ , delegates cross-chain decisions to  $\mathcal{F}_{\text{XVer}}$ , and mirrors finalized relay annotations. On RECEIVE/Reject,  $\mathcal{S}$  records  $Tx_{\text{rece}} = \text{Invalid}$  and annotates the paired  $\text{Tid}_A \rightarrow \text{Invalid}$ . On RECEIVE/Accept, it annotates  $\text{Tid}_A \rightarrow \text{Finished}$  and outputs the updated relay/account records.

##### Simulator $\mathcal{S}$ .

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

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

On SEND: precheck or FAIL. Otherwise register Tid<sub>A</sub> with sta = Valid, emit  $\mathcal{L}_{\text{pub}}$  with simulated proof. Call  $\mathcal{F}_{\text{XVer}}.\text{VerifySend}$ . On Accept, set sta = Send, decrease  $z[a]$ , emit. On Reject, set Invalid.

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

**SU:**  $\mathcal{S}$  rejects any second annotation after finality, keeping  $\text{Send} \rightarrow \{\text{Finished}, \text{Invalid}\}$  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 adversary.

$H_0$  (**Real**). Real execution of  $\Pi$  with  $\mathcal{A}$  in the  $(\mathcal{G}_{\text{Ledger}}, \mathcal{F}_{\text{CRS}}, \mathcal{F}_{\text{XVer}})$ -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  $\mathcal{F}_{\text{CRS}}$ . Witnesses are never used.

**Lemma 3** ( $H_0 \stackrel{c}{\approx} H_1$ ). 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** ( $H_1 \stackrel{c}{\approx} H_2$ ). 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  $\mathcal{F}_{\text{VCT}^*}$  and  $\mathcal{F}_{\text{XVer}}$  and emulating the relay via its visible actions.  $\mathcal{S}$  outputs exactly  $\mathcal{L}_{\text{pub}}$  and enforces SU via RSM.

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

**Lemma 6** (Bridge Lemma: Ledger → Atomicity). If  $\mathcal{G}_{\text{Ledger}}$  provides deterministic finality and the relay state machine enforces SU, then for any Tid<sub>A</sub> the public transcript admits no observable intermediate state between Send and {Finished, Invalid}. Hence, the transition is atomic and Tid<sub>A</sub> 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  $\text{EXEC}_{\Pi, \mathcal{A}, \mathcal{Z}}^{\text{real}} \stackrel{c}{\approx} \text{EXEC}_{\mathcal{F}_{\text{VCT}^*}, \mathcal{F}_{\text{XVer}}, \mathcal{S}, \mathcal{Z}}^{\text{ideal}}$  in the  $(\mathcal{G}_{\text{Ledger}}, \mathcal{F}_{\text{CRS}}, \mathcal{F}_{\text{XVer}})$ -hybrid model.

*Proof.* Chain the lemmas:  $H_0 \xrightarrow{c} H_1 \xrightarrow{c} H_2 \xrightarrow{c} H_3$ .  $H_3$  is the ideal execution with  $\mathcal{S}$  talking to  $\mathcal{F}_{\text{VCT}^*}$  and  $\mathcal{F}_{\text{XVer}}$ . Verifiability follows from soundness (no false acceptance), and balance safety from per-operation relations and SU (no double-consumption), strengthened by the Bridge Lemma.  $\square$

## 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  $C_j \in \{\text{dep}, \text{wd}, \text{send}, \text{rece}\}$ , and let the reduction  $\mathcal{R}$  guess  $j$  uniformly at random. Then*

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

*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  $\mathcal{G}_{\text{Ledger}}$  and keeps shadow balances from  $\mathcal{L}_{\text{pub}}$ . Whenever  $\mathcal{A}$  outputs a finalized accepting transaction  $(x, \pi)$  of type  $C_j$ :

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

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  $\text{negl}$ ) or implies an inconsistent accepting proof covered above.  $\square$

## S13. PRIVACY THEOREMS

We write  $\text{View}^{H_i}$  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  $\Delta_i := |\Pr[\text{View}^{H_i} \in \mathcal{W}] - \Pr[\text{View}^{H_{i+1}} \in \mathcal{W}]|$  for any distinguisher/event  $\mathcal{W}$ . By the triangle inequality,

$$|\Pr[\text{View}^{H_0} \in \mathcal{W}] - \Pr[\text{View}^{H_3} \in \mathcal{W}]| \leq \sum_{i=0}^2 \Delta_i.$$

We now bound each  $\Delta_i$  by a standard reduction:

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

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

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

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  $\text{View}^{\text{ideal\_amt}}$  be the distribution that replaces all plaintext amounts by dummies while preserving  $\mathcal{L}_{\text{pub}}$ .  $\Pi$  has amount privacy if  $\text{View}^{\text{real}} \xrightarrow{c} \text{View}^{\text{ideal\_amt}}$ .*

**Theorem 2** (Amount Privacy).  *$\text{View}^{H_0} \xrightarrow{c} \text{View}^{H_1}$  by ZK (witness-stripping), and  $\text{View}^{H_1} \xrightarrow{c} \text{View}^{H_2}$  by IND-CPA. Hence,  $\text{View}^{H_0} \xrightarrow{c} \text{View}^{H_2}$ . Since  $H_3$  only emits  $\mathcal{L}_{\text{pub}}$ ,  $\text{View}^{H_2} \xrightarrow{c} \text{View}^{H_3}$ . Moreover,*

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

### S13.3 Balance Privacy

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

**Theorem 3** (Balance Privacy). *ZK removes witness dependence ( $H_0 \rightarrow H_1$ ), and hiding (or IND-CPA if encrypted) swaps balances/commitments ( $H_1 \rightarrow H_2$ ), while  $H_2 \rightarrow H_3$  leaves only  $\mathcal{L}_{\text{pub}}$ . Moreover,*

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

### S13.4 Fund-Correlation Privacy

**Definition 3** (Fund-Correlation Privacy). *For two histories  $Q_0, Q_1$  that agree on  $\mathcal{L}_{\text{pub}}$  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  $M_{\text{rece}} = M_{\text{send}}$  is enforced by  $\mathcal{F}_{X\text{Ver}}$  from the public transcript, thus  $\text{View}^{H_2}$  and  $\text{View}^{H_3}$  remain identical for any re-pairing preserving  $\mathcal{L}_{\text{pub}}$ . Therefore  $\text{View}^{H_0}$  is indistinguishable. Moreover,*

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