VENUO TECHNICAL PAPER

# Zero-Knowledge Allocation Claims for Scheduled Token Sales on Solana

*A commitment, membership proof, and nullifier construction for private primary-sale allocation*

Venuo Protocol, Draft v1.0

September 2026

**Abstract**

Scheduled token sales conventionally deliver allocation to the wallet that funded participation, publishing a permanent link between contribution and delivery. We present a construction that replaces direct delivery with a zero-knowledge claim. A participant contributes a fixed lot and appends a secret-derived commitment to a sale-scoped Merkle tree. When contributions close, the sale freezes one final root. The participant later proves membership under that root, reveals a one-time nullifier, and binds settlement to a chosen destination without revealing which commitment is spent. A Solana program verifies the proof and atomically settles either the fixed token allocation or a cancelled-sale refund. We specify the roles, lifecycle, cryptographic relation, account model, privacy objective, threat model, and evaluation methodology for this construction, and state precisely what it does and does not hide.

***Scope.*** This paper specifies a protocol design. It narrows the private state to one entitlement secret and one finalized membership proof; it does not describe a general shielded-balance system, a private secondary market, or a deployed and audited implementation.

# 1 Introduction

Direct-delivery token sales create a persistent graph edge between the account that pays for participation and the account that receives tokens. Address reuse, exchange withdrawal patterns, and public profiles let that edge become an identity and portfolio signal that outlives the sale itself.

We separate a sale into two phases. **Contribution** creates a public entitlement commitment derived from a private secret; it does not select a token destination. **Claim** proves entitlement after the sale closes and binds the payout to a destination chosen at claim time, which may differ from the contributing wallet.

**Security objective.** For a finalized sale and a fixed allocation lot, a passive chain observer should not learn which commitment under the final root authorized a given claim, except through side information outside the proof relation itself. This is a source-set ambiguity, not full transaction anonymity: the proof hides the witness, while unlinkability in practice remains conditional on timing, infrastructure behavior, and wallet history.

**Scope.** Venuo v1 is a private allocation layer for scheduled primary issuance. It is explicitly **not**: a private SOL transfer protocol; a mixer or general shielded pool; a bonding-curve market or private exchange; a way to hide claim destinations or allocation amounts; an identity, KYC, or Sybil-resistance system; or a guarantee against MEV, censorship, or network-level observation.

A contribution window that closes before claims open yields one immutable eligibility set. This avoids races against a moving tree, gives every claim one fixed cohort for privacy analysis, and lets fixed lot sizes reduce amount fingerprinting while simplifying escrow accounting.

# 2 System Model

## *2.1 Roles*

| Role | Responsibility |
|---|---|
| Creator | Defines sale terms and escrows the full launch-token liability before activation |
| Contributor | Generates a claim secret, pays one or more fixed lots, and stores each entitlement |
| Claimant | Chooses a destination and produces a proof authorizing claim or refund |
| Submitter | Pays transaction fees and posts a destination-bound proof; may be the claimant or a sponsor |
| Venuo program | Maintains sale state, escrow, the commitment root, nullifiers, and settlement |
| Verifier | Checks the zero-knowledge relation against the sale's pinned verification key |

## *2.2 Lifecycle*

Every sale advances through one linear path with a single branch at closure:

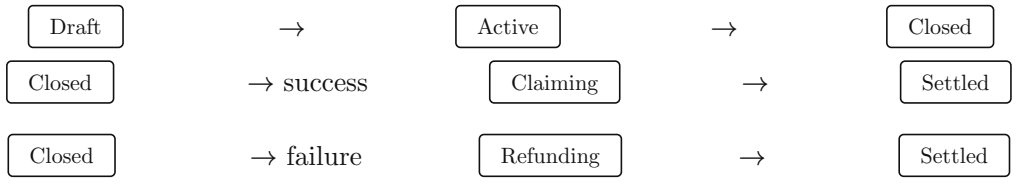


*Figure 1. Sale lifecycle. Closure evaluates one success condition and routes to exactly one of the two settlement branches.*

Draft requires full inventory funding before Active can open. Closed is reached at the close slot or at capacity, and permanently freezes the commitment root before any claim is accepted. Claiming and Refunding are mutually exclusive per sale: a successful sale never opens refunds, and a cancelled sale never opens token claims. Permissionless callers, not the creator, may trigger closure, claims, and refunds once objective conditions hold.

# 3 Cryptographic Construction

### *3.1 Field and encodings*

The proof relation operates over the BN254 scalar field of order $q = 21888242871839275222246405745257275088548364400416034343698204186575808495617$. A Solana public key is a 32-byte string. We encode it injectively as two 128-bit limbs taken directly from the raw bytes in little-endian order, $(\mathrm{lo}, \mathrm{hi}) = (\mathrm{LE}_{128}(\mathrm{pk}[0..16]), \mathrm{LE}_{128}(\mathrm{pk}[16..32]))$, and range-constrain both limbs in-circuit. A key is never reduced modulo $q$, and any public input at or above $q$ is rejected before verification.

### *3.2 Secrets and hash domains*

Each entitlement uses a fresh non-zero secret $s$ sampled by rejection into the scalar field from a cryptographically secure source — never by reducing raw bytes modulo $q$, which would bias the distribution. Reusing a secret within one sale yields the same nullifier and therefore one spendable entitlement regardless of how many commitments reference it.

Venuo hashes with Poseidon over the scalar field under six distinct domain tags: sale context, entitlement leaves, Merkle nodes, nullifiers, claim bindings, and the empty leaf. Domain constants and Poseidon parameters are versioned together with the verification key and a shared test-vector suite, so an implementation's encoding manifest — not this paper — is normative for a given deployment.

### *3.3 Sale context*

The sale context ctx scopes every entitlement to one deployment and one immutable configuration. It commits to the protocol and circuit version, the network domain and program ID, the sale account and creator, the launch-mint and token-program identity, the fixed lot and allocation, the tree depth and lot capacity, and the contribution and claim windows. The Solana program derives ctx from its own canonical accounts at initialization; a client never supplies context directly.

### *3.4 Entitlement, nullifier, and claim binding*

For secret $s$, context ctx, fixed allocation $a$, action $\mathrm{op} \in \{\mathrm{CLAIM}, \mathrm{REFUND}\}$, and destination limbs $(d_{\mathrm{lo}}, d_{\mathrm{hi}})$:

```
cm = H_leaf(s, ctx, a) (entitlement commitment) nf = H_nullifier(s, ctx) (one-time nullifier)
binding = H_claim(op, nf, ctx, a, d_lo, d_hi) (destination binding) root = MerkleRoot(cm,
siblings, direction_bits)
```

The commitment cm is public once contributed; its preimage stays private. The public claim instance is $x = (\mathrm{root}, \mathrm{nf}, a, \mathrm{ctx}, \mathrm{binding})$; the private witness is $w = (s, \mathrm{op}, d_{\mathrm{lo}}, d_{\mathrm{hi}}, \mathrm{siblings}[20], \mathrm{bits}[20])$.

The circuit accepts exactly when every direction selector is Boolean, both destination limbs lie in $[0, 2^{128})$, op is one of the two fixed encodings, cm folds through the supplied path to the public root, the public nf and binding recompute correctly from the witness, and the allocation is canonically encoded within the supported range. The Solana program separately re-checks that root, ctx, and $a$ equal canonical sale state — proof validity alone never authorizes settlement.

# 4 Commitment Tree and Settlement

### *4.1 Append-only Poseidon tree*

Venuo uses a binary Poseidon Merkle tree of fixed depth 20 (capacity $2^{20}$ leaves). Domain-separated empty subtree roots are defined recursively by $z_0 = H_{\mathrm{empty}(0)}$ and $z_{\{i+1\}} = H_{\mathrm{node}(z_i, z_i)}$, and every parent is $H_{\mathrm{node(left,\ right)}}$ with order-sensitive hashing. Leaves append left to right; a frontier of per-level partial subtrees allows $O(20)$-cost insertion. While Active, every accepted contribution advances the current root. At Closed, insertion stops permanently and the current root is copied into an immutable `finalRoot` — claims and refunds accept `finalRoot` only, never an intermediate or historical value. Tree capacity of $2^{20}$

is an engineering bound, not a privacy metric: effective privacy is set by the plausible honest cohort after side information, not by raw leaf count.

### *4.2 Contribution and settlement flow*

| Contributor | Venuo program | Destination |
|---|---|---|
| 1. Pay fixed lot, submit commitment → | | |
| | 2. Transfer lot to vault, append leaf, update root | |
| | 3. Close: freeze `finalRoot`, evaluate outcome | |
| 4. Build membership proof, choose destination → | | |
| | 5. Verify proof, recheck context/allocation, consume nullifier | |
| | 6. Settle: transfer tokens or refund → | 7. Receive settlement |

*Figure 2. Contribution and settlement steps. Contribution (1–2) and settlement (4–6) are each atomic; the destination need not fund the contribution.*

### *4.3 Claim and refund acceptance*

A token claim succeeds only when the sale is in Claiming with the claim window open; the public `root`, `ctx`, and allocation equal canonical sale state; the proof's destination limbs match the destination token account owner; `binding` recomputes from action `CLAIM`; the Groth16 proof verifies under the sale's pinned key; the nullifier is unused; and the mint, token program, vault, and destination account are all canonical. The transaction then writes a permanent nullifier record and transfers the exact allocation in the same instruction. A refund follows the identical relation with action `REFUND` in the Refunding phase, returning one fixed lot instead of tokens. Because claim and refund share one nullifier domain, at most one terminal action can ever complete for a given entitlement — an observer who copies a valid proof and submits it first still pays the **original** bound destination, so copying cannot redirect funds, only race for inclusion.

## 5 Privacy Model

### *5.1 What is protected and what remains visible*

| Protected by the proof | Visible on-chain regardless |
|---|---|
| Which commitment authorizes a claim | Funding wallet and contribution amount |
| Direct contribution-to-destination edge | Sale, allocation lot, and contribution timing |
| Redirection via a copied proof | Claim destination, allocation, and claim timing |
| Reuse of a spent entitlement | RPC, relayer, browser, and device metadata |

### *5.2 Effective claim set*

The cryptographic relation omits the spent leaf, but real observers combine it with side information. For finalized commitments $S$ and a claim $c$, we define the **effective claim set** as the subset of $S$ that remains plausible after filtering by same-sale and same-lot membership, already-spent nullifiers, contribution-to-claim timing proximity, shared fee-payer or account-creation patterns, shared RPC/relayer/browser/device identifiers, and destination funding or downstream consolidation behavior. The **size** of this set is not a uniform probability of identification — observers hold unequal side information, and one strong off-chain link can dominate a large on-chain set. Creator-controlled or Sybil commitments do not contribute independent privacy to an honest claimant, so reporting should separate raw commitment count from the lower-bound honest plausible set whenever it is estimable.

Venuo does not hide contributions, destinations, allocations, claim timing, or the fact that a claim occurred, and it does not protect a claimant whose device or secret is compromised.

# 6 Threat Model and Security Properties

| Adversary | Capability |
|---|---|
| Chain observer | Reads every transaction, account, log, timestamp, and token flow |
| Malicious claimant | Submits malformed proofs, wrong accounts, or duplicate/racing claims |
| Malicious creator | Injects Sybil commitments or misrepresents sale terms |
| Transaction observer | Copies a valid proof to front-run inclusion |
| Infrastructure operator | Logs RPC, relayer, browser, and analytics metadata |
| Setup adversary | Retains Groth16 toxic waste to attempt proof forgery |

Given knowledge-soundness of the proof system and collision-resistance of Poseidon at the target security level, five properties hold: **claim authorization** (an accepted proof implies a witness satisfying membership, nullifier, and binding); **settlement uniqueness** (deterministic nullifier derivation plus a permanent record admits at most one terminal action per entitlement); **destination integrity** (changing the destination changes `binding` and invalidates the proof); **cross-context replay resistance** (context binds network, program, sale, mint, lot, allocation, and version, so a proof from one deployment cannot verify in another); and **escrow solvency** (activation requires full liability funding, and Claiming/Refunding remain mutually exclusive so creator withdrawal never touches refund reserves).

| Residual risk | Mitigation and boundary |
|---|---|
| Proof forgery | Bounded by ceremony integrity and a pinned verification key; setup compromise is catastrophic |
| Nullifier-PDA prefunding | Program must allocate/assign the signed PDA rather than assume a zero-lamport address |
| Sybil privacy inflation | Raw leaf count is never reported as honest anonymity |
| Timing correlation | Phase separation and claimant-chosen delay reduce, but do not eliminate, correlation |
| Indexer correlation | Clients reconstruct paths locally or verify complete snapshots; no owned-leaf queries |
| Upgrade compromise | Multisig, timelock, reproducible builds, and eventual immutability bound this risk |

# 7 Limitations, Related Work, and Conclusion

## *7.1 Limitations*

Contributions and claim destinations are public by design; a thin or Sybil-dominated sale yields little practical unlinkability; claim-secret compromise transfers entitlement authority outright; the trusted setup and any upgrade authority are high-value trust points; one writable tree serializes contributions within a sale; complete insertion history must remain available for witness reconstruction; and Venuo makes no claim about token quality, value, liquidity, or legal compliance.

## *7.2 Related work*

The commitment-plus-nullifier pattern follows the line of work established by Zerocash and the Zcash protocol [1], [2], and the destination and Merkle-tree conventions used here are close to Semaphore's group-membership circuits [3]. Groth16 [4] and Poseidon [5] supply the proof system and hash primitive respectively. Privacy Cash and Elusiv [6], [7] demonstrate the same primitive deployed on Solana with production verifier and relayer infrastructure; Token-2022 Confidential Balances [8] shows the complemen-

tary limit of the design space, where encrypted amounts alone leave the account graph fully public — motivating the membership-proof approach used here instead.

### *7.3 Conclusion*

A public contribution creates a secret-backed entitlement; a frozen sale root creates one immutable eligibility set; a zero-knowledge membership proof binds that entitlement to a freshly chosen destination; and a sale-scoped nullifier enforces one-time settlement. Solana account validation and atomic execution connect this cryptographic authorization to exactly the escrowed assets it is permitted to move. The result does not make a token sale invisible — it removes one specific protocol-level mapping, between contribution commitment and claim, while leaving public sale rules, deterministic allocation, and independently verifiable settlement intact.

## References


[1] E. Ben-Sasson *et al.*, "Zerocash: Decentralized Anonymous Payments from Bitcoin." [Online]. Available: https://eprint.iacr.org/2014/349

[2] "Zcash Protocol Specification," 2024. [Online]. Available: https://zips.z.cash/protocol/protocol.pdf

[3] "Semaphore: Zero-Knowledge Group Membership Protocol." [Online]. Available: https://docs.semaphore.pse.dev/technical-reference/circuits

[4] J. Groth, "On the Size of Pairing-Based Non-interactive Arguments." [Online]. Available: https://eprint.iacr.org/2016/260

[5] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A New Hash Function for Zero-Knowledge Proof Systems." [Online]. Available: https://eprint.iacr.org/2019/458

[6] "Privacy Cash: Solana Program and Circuits." [Online]. Available: https://github.com/Privacy-Cash/privacy-cash

[7] "Elusiv Protocol Source Archive." [Online]. Available: https://github.com/amathxbt/elusiv

[8] "Confidential Transfer — Token-2022 Extensions." [Online]. Available: https://solana.com/docs/tokens/extensions/confidential-transfer