COVARA

Launch openly.
Move privately.

A privacy focused launchpad
and capital workspace

Designed around Sigilo. Targeting Robinhood Chain.

TECHNICAL & PRODUCT WHITEPAPER

Version 0.5 / 23 October 2026

Executable local implementation. Real proofs and contracts.
Test assets only. No public deployment, audit or endorsement implied.
Working brand; clearance pending. No platform token required initially.

Abstract

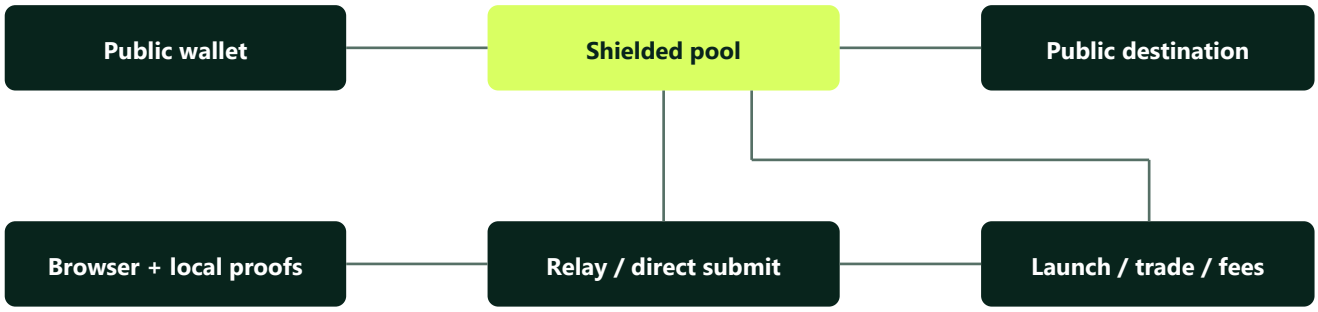
COVARA combines Sigilo-derived shielded value transfers with fixed-price token launches, atomic buys and sells, and owner-authorized payments to encrypted fee recipients. The product provides one browser workspace, with an explicit view of the public information exposed by each action. Its current release runs actual Solidity contracts on a local EVM and generates real Groth16 proofs in the user's browser.

Privacy is applied to note ownership and recipient values. Project terms, market execution, launch contribution amounts, receipt-to-claim links, fee sources and batch totals remain public. This distinction is fundamental: COVARA does not claim that a public blockchain hides all financial activity.

Version 0.5 includes complete-note exits, portable recovery transactions and independent preparation and submission tools. Fresh proofs can be generated from a phrase and saved public history without the main API or relay. The target public network is Robinhood Chain. Public deployment remains disabled pending cryptographic review, verified setup provenance and production infrastructure. The platform does not require a native COVARA token.

1. Product and system

The local system consists of a browser wallet and prover, an event-indexing service, an operator-funded local relay, shielded pools and four settlement modules. A shared ETH pool allows local projects to use a common pool, while each project has its own token pool, launch escrow, constant-product market, router and fee ledger. The frontend makes the disclosure boundary visible before execution.



Public inputs and outputs remain visible. New modules need independent review.

Indexer reconstructs public events; private keys and witnesses stay on the user device.

The deployment helper accepts only local chain 31337. It uses disposable test tokens and known development accounts. The local faucet supplies test ETH, the operator seeds market liquidity, and the local fee-signing endpoint represents the test project's owner. These conveniences are not a production custody or authorization model.

2. Shielded notes and proof statement

A note contains a bounded integer value, an owner's receiving-key value and fresh randomness. The pool stores a cryptographic commitment rather than the plaintext note. A recipient scans encrypted note payloads using a private viewing key. Spending requires a proof of authorization and Merkle membership, and reveals a deterministic nullifier that prevents the same note from being spent twice.

The inherited circuit supports four input slots, four output slots and a depth-20 commitment tree. Values are bounded to unsigned 128-bit integers. Ordinary operations publish four equal-length encrypted payloads, including randomized zero-value padding. Each pool holds one asset; proof context binds the operation to the chain and pool.

```
sum(input values) + public amount in = sum(output values) + public amount out + explicit fee
```

The proof and contract composition bind public amounts, the root, nullifiers, output commitments, ciphertexts and external execution context. Recipient, executor, submitter, expiry, unwrap preference, intent and bundle identifier are part of that context. The contract rejects invalid proofs, spent nullifiers, unknown roots and expired operations.

The browser checks proving-artifact hashes and verifies its generated proof before submission. Hash equality establishes the identity of the artifact bytes. It does not establish a trustworthy setup ceremony or prove circuit soundness. The inherited setup is explicitly classified as development-only because independent ceremony provenance has not been established.

3. Transfers and privacy boundaries

An internal transfer consumes existing notes and creates encrypted output notes. A relayer receives a proof and public operation data, not the user's phrase, spending key or private witness. A public deposit or withdrawal crosses a visibility boundary: amount, pool and public endpoint can be observed. A normal partial withdrawal may create a private change note.

Flow	Concealed relationship	Public information
Internal transfer	Recipient keys and note values	Pool, commitments, nullifiers, timing and ciphertexts
Launch participation	Normal wallet association and encrypted claim recipient	Terms, contribution amount, receipt identifier and claim linkage
Buy or sell	Ownership of shielded input/output notes	Market, assets, amounts, fees, execution price and timing
Fee batch	Recipient keys and each encrypted note value	Authorizing owner, source, asset, total and timing
Public recovery	Remaining unrelated private notes	Recipient, recovered amount and gas payer when submitted directly

Timing, amount matching, small anonymity sets, RPC access, browser compromise and endpoint metadata can weaken practical privacy. A valid proof does not prevent all correlation. COVARA makes no total-anonymity claim.

4. Fixed-price launches

A project publishes immutable soft cap, hard cap, token price, close time, creator, fee owner and pool addresses. Opening requires depositing token inventory for the entire hard cap. A contribution withdraws shielded ETH directly into the launch contract with an intent bound to a receipt identifier. Over-cap contributions revert in full; the implementation does not silently accept partial fills.

Each receipt is controlled by a separate signing key derived from the user's recovery phrase, chain, launch address and receipt index. The initial UI supports 16 receipt slots per launch. A signature binds a claim to its destination pool operation, allowing allocation or refund value to become an encrypted note.

Cancellation is available only before the first contribution. Anyone can settle when the hard cap or deadline is reached. Successful settlement reserves allocated inventory, credits a 50-basis-point fee and credits the creator's remaining proceeds to the fee ledger. Failed settlement opens full principal refunds. Public recovery can pay the underlying token if a destination pool cannot accept a deposit.

The receipt identifier is public at contribution and claim. This release does not implement the original blueprint's stronger unlinkable allocation construction. Separate receipt keys hide a routine wallet association; they do not erase receipt-to-claim linkage.

5. Atomic trading

Each local project has one immutable constant-product token/WETH market. Seed liquidity is permanently locked. Trades charge 20 basis points of the input asset to the project owner's fee balance. The price calculation uses integer arithmetic and the current reserves.

A trade combines an input withdrawal proof, public market execution and an output deposit proof in a single transaction. The router binds direction, amounts and the output operation identifier into the input intent. The reviewed expected output must match execution exactly. If a quote changes, either proof fails or any downstream operation fails, the entire transaction reverts.

This design gives failure atomicity between shielded endpoints. It does not hide the market execution. There is no arbitrary external router or unreviewed DEX call target in the implemented path. Broader asset and venue support would require additional integration review.

6. Private fee recipients

The fee contract records credits by owner and asset. A batch routes an owner's available balance into one to four encrypted recipient notes. Exact integer allocation conserves the total; deterministic remainder handling prevents unassigned dust. Recipients may be any compatible COVARA receive addresses on the selected chain.

The owner's domain-separated authorization binds the nonce, chain, fee contract, destination pool and full operation. A relayer cannot substitute recipient notes after that authorization. Public recovery instead binds a recipient, amount, deadline and nonce. Replay protection applies to both paths.

The circuit proves value conservation. It does not prove that a privately agreed percentage was fair or that an off-chain beneficiary list was honored. The implemented product is an owner-authorized payout mechanism. Recurring recipient-controlled hidden-share entitlements remain a separate research requirement. The UI now offers an external owner-wallet signing path as well as the explicit local test fixture. The adapter computes the domain digest locally, compares it to the contract, verifies the selected owner and signature, and rejects network, deployment or nonce changes. External-wallet and public-network validation remain open.

7. Recovery and availability

The phrase is the user's recovery authority. Device backups encrypt the phrase using PBKDF2-SHA256 with a random salt and 310,000 iterations, plus AES-256-GCM with a random nonce and authenticated application context. A device password is not a replacement for an independent phrase backup. Locking cancels the proof worker and drops key references; JavaScript cannot guarantee physical memory erasure.

A complete-note exit spends up to four whole notes without producing change or inserting leaves. The recovery planner also separates batches when a combined amount would exceed the circuit's 128-bit limit. Each batch receives its own review. A deposit cap of zero cannot block this exit path. The local tests recover seven notes in two batches while verifying that the tree root and leaf count remain unchanged.

The browser can export a public recovery proof and exact calldata. Exporting moves no funds. A static recovery desk can inspect that file and submit it through an injected EVM wallet without the COVARA relay or API. The wallet must have gas and a connection to the same deployed pool. The submission path checks chain, bytecode, artifact set, expiry, root, nullifiers and simulated execution, and rechecks account/network after the connection prompt.

The independent preparation tool restores a phrase against saved public encrypted history, validates event order and roots, checks pinned proving-file hashes and generates a new proof in a worker. The phrase is not persisted or transmitted to an API. Private state clears when the user leaves, explicitly clears the session, or stops interacting for five minutes. The displayed balance is historical. A saved history file is not a certificate of canonical chain state; missing spend events or later activity can make it misleading. Live checks and the on-chain verifier remain mandatory. A separate history page refreshes records from a chosen RPC without receiving a phrase. It checks the network, verified code hash, artifact set, event blocks and reconstructed tree at one pinned block. Cancellation discards partial results. RPC trust remains necessary; these checks do not establish consensus. Separately hosted distribution and external-wallet/device validation remain open. A full million-leaf tree was not constructed in the current tests; capacity independence is supported by the exit code path and unchanged-leaf assertions.

8. Service and wallet integrity

The browser rebuilds the note tree, compares the current root and leaf count, then checks candidate nullifiers against a pinned chain block. The local service serializes writes and checkpoints the chain together with public encrypted events. It also indexes direct pool transactions from newly mined blocks. Startup reserves the HTTP port and takes an exclusive workspace lock before loading saved state, preventing duplicate processes from overwriting the same checkpoint.

The service binds to loopback, restricts host/origin and request sizes, exposes read-only RPC methods and serves an explicit set of public directories. No API accepts a mnemonic or private witness. The development service retains its Anvil checkpoint index. A separate read-only canonical indexer pins deployment code and artifacts, replaces orphaned events within a retained header window, verifies reconstructed roots at the indexed block, and preserves its checkpoint on deeper reorganizations. It is tested against real local-chain reorganizations. Confirmation depth is not L1 finality; public operations still require network-specific finality handling, storage performance, monitoring and controlled gas signing.

The submission desk keeps a small public-transaction bundle. Independent preparation loads its own worker of approximately 3.11 MB uncompressed, plus an on-demand prover chunk and saved proving files. The main wallet also carries a substantial proving stack. The browser preparation workflow bounds history files to 50 MB and 100,000 events; large histories need further performance work. Desktop execution has been exercised; physical mobile proving performance and memory limits are not yet measured.

9. Evidence and limits

The release contains 18 protocol integration scenarios, six additional offline recovery scenarios, four canonical-index/reorganization scenarios, eleven unit checks, seven main API checks, five standalone server checks and seven direct-history scenarios. Integration coverage includes proof/ciphertext tampering, replay rejection, atomic buy/sell rollback, signed fee routing, launch caps and settlement, private refunds, public recovery, multi-batch independent exits, wallet network/account changes and checkpoint restoration. Browser validation separately covers the four utility flows, exports, recovery inspection, independent phrase restore/proving while the main service is stopped, and responsive layouts.

Solidity builds enforce the EIP-170 runtime size limit. Dependency scanning reports 20 low advisories and no moderate, high or critical advisories in the recorded scan. This is not a clean audit report. Individual source licenses include MIT and GPL components; the package must not be represented as MIT-only.

The primary unresolved security condition is independent review of the circuit, setup and composed contracts. Test success cannot certify cryptographic security, protect against an unknown dependency flaw or replace review of operational authority.

10. Robinhood release path

Robinhood Chain is the intended public network. Official configuration identifies chain 46630 for testnet and 4663 for mainnet, with ETH for gas. The local release deliberately accepts neither. Public RPC attempts from this environment did not establish a working connection; dedicated infrastructure and a funded deployment plan are not configured.

The next public milestone is a capped testnet transfer rehearsal after setup provenance and review are resolved. The implemented owner-wallet adapter and canonical indexer require external-wallet and network-specific operational validation. Authenticated relay operations, public-network history operations, device testing and verified deployments must precede a public beta. Launch, trade and fee modules require feature-specific review and exposure limits even though all four execute locally.

Building on Robinhood Chain does not imply Robinhood affiliation, endorsement or a brokerage listing. No campaign posts have been published and no platform-token sale is proposed by this release. COVARA remains a working brand pending trademark and domain clearance.

11. Sources and reproducibility

Primary implementation evidence is included in **contracts/**, **packages/protocol/**, **services/**, **apps/web/**, the lockfile and **output/validation/**. The source-provenance manifest records copied Sigilo files and their original hashes. The current implementation record, operations guide and recovery guide specify supported behavior and remaining work.

Official network references: Robinhood network configuration and Robinhood developer connections. These describe the target network, not a COVARA deployment.

Reproduce the local release with **npm ci --ignore-scripts**, **npm run verify**, and **npm run dev**. Run API checks while the local server is active. Start the independent desk with **npm run recovery:serve**. Keep local chain state and any private wallet material out of distributable packages.