Styx PROTOCOL

Formerly Protocol 01

DESIGN AND ARCHITECTURE

Private payments on Solana. Built to be checked, not believed.

A shielded payment pool with hash-based STARK proofs and hybrid post-quantum stealth addresses. This document describes what is built, how it works, and what it does not do yet.

LIVE ON SOLANA DEVNET | NOT AUDITED | NO MAINNET DEPLOYMENT | OPEN SOURCE

revision	August 2026
supersedes	June 2026 edition, issued as Protocol 01
source	github.com/IsSlashy/Protocol-01

HOW TO READ THIS

Every claim here is one a stranger can check.

This edition exists because the previous one overstated. An adversarial audit of the project's own material found 46 claims that were false and 83 that were overstated, several of them checkable in minutes by anyone with a block explorer. So this document was rebuilt against measurement.

WHAT THAT MEANS IN PRACTICE

Every figure printed here was verified against the repository, the npm registry or Solana devnet on 11 August 2026, and then handed to a second reader whose job was to disprove it. Anything that did not survive that pass was removed rather than softened. An erratum dated 2 September 2026 sits on pages 4, 11 and 17: the spend circuit those pages describe as not built shipped on 25 August 2026, and the pairing they measure no longer exists on the path the web app uses.

Where a number could not be sourced, the sentence went instead of the number. Page 18 lists the command or the address behind the load-bearing figures so a reader can repeat the measurement rather than trust the document.

WHAT IS DELIBERATELY NOT HERE

No user counts, no transaction volume, no value locked, and no performance figure without its benchmark beside it. A devnet has none of the first three worth quoting.

The words untraceable, anonymous and unlinkable do not appear as claims about this protocol. The reason is on page 17: a deposit can still be matched to its withdrawal today, and the document says so before it says anything flattering.

READ THIS BEFORE THE ARCHITECTURE

Styx runs on Solana devnet and has not been audited. There is no mainnet deployment. Test tokens are not funds. Everything described in the pages that follow exists and runs; none of it has been reviewed by anyone outside the project, and one privacy property described on page 17 is not closed yet.

THE PROBLEM

A public ledger publishes your customer register for you.

Solana settles in the clear: every transaction carries its fee payer, its signers, its account list and its program id as plaintext. A payment rail built on that inherits a public register.

IT NEEDS NO ATTACKER, ONLY A READER

Nothing has to be broken, and nothing here is a defect: this is the runtime working as designed. On 11 August 2026, one unauthenticated `getProgramAccounts` call against the Solana Foundation's own Subscriptions and Allowances program, shipped in June 2026, returned 10,782 accounts, which group into five shapes of 7,237, 2,896, 342, 303 and 4.

That program is open source and was reviewed by Cantina and Spearbit. It is good infrastructure. It is also a list, and reading all of it costs one request.

THIS DOCUMENT DOES NOT EXEMPT STYX

The subscription vaults Styx has deployed on devnet keep the merchant address in the clear at a fixed byte offset, 42, of a public account. One call filtered on nothing but the account discriminator returned all 18 vaults across 9 retailers, with each vault's deposited total and claimed period count decoding beside it.

No key, no SDK, no permission. The project's own merchant SDK README says so at lines 378 to 386, and declines to call that path a privacy control.

WHO FEELS IT FIRST	WHAT THE LEDGER PUBLISHES ABOUT THEM
Recurring revenue	Which addresses pay, at what price, on what schedule, and the month they stop
Payroll	Every salary, and every change to one, at the moment it is paid
Treasury	Balances, counterparties, and the size of a move while it is still in progress
Business to business	Which suppliers a firm depends on, and what it pays each of them

The current answer is to delete the billing

Two companies that sell privacy as their product removed recurring payment rather than solve it. Mullvad published a post titled "We are removing the option to create new subscriptions". NymVPN shipped a pay as you go option with no account and no subscription. Both answers are a retreat.

This page is the diagnosis, not the claim. Styx does not close all of it: measured today, a deposit into the shielded pool can still be matched to its withdrawal, and the sender is not hidden on the send path. Page 17 states that limit with the measurement behind it.

Both counts were measured on 11 August 2026, one `getProgramAccounts` call each: 10,782 on `De1egAFMkMWZSN5rYXRj9CAdheBamobVNubTsi9avR44` at `api.mainnet-beta.solana.com`, grouped by account size and leading byte; 18 vaults on `GbVM5yvetrSD194Hnn1BXnR56F8ZWNknij7DoVP9j27c` at `api.devnet.solana.com`, filtered on discriminator `605af7ca9d1056be`.

SYSTEM OVERVIEW

Four moves, and the chain records each one.

Money enters a fixed-denomination pool, is spent as a note rather than from an address, leaves to a fresh recipient, or is collected for a merchant. Every mechanism described later sits under one of those four moves, in that order.

MOVE	INSTRUCTION	WHAT THE CHAIN RECORDS	CU
Shield	`ShieldDenominatedV3`	The payer, the pool and therefore the amount, a 0.3% fee taken on top, and one note commitment at a numbered leaf.	24,356
Pay	`SubscribePrivateStark`	A fresh vault holding the full pool denomination, and the merchant it may pay. No protocol fee on this leg.	30,762
Unshield	`UnshieldDenominatedStarkV3`	A nullifier, and 0.995 SOL credited to a recipient with no prior history after the 0.5% fee.	23,415
Claim	`ClaimPeriod`	Anyone may call it. 1,002,721,360 lamports went to the merchant named in the vault; the caller gained nothing.	5,871

Devnet, 4 to 6 August 2026, one transaction each. Those are settling instructions only. The STARK proof is verified in earlier transactions: one subscription run spent 1,913,751 compute units on two proofs across a burst of 172 transactions, 148 of them proof chunk uploads, and posted about 1.026 SOL of buffer rent that came back on close.

WHAT THE CHAIN LEARNS

The amount, always. Pools are denominated at 0.1, 1, 10, 100, 500 and 1000 SOL, so joining one publishes the size. It learns the depositor, the exit address, and the size of the crowd a spend hides in: the busiest pool holds 8 unspent notes over 34 leaves.

WHAT IT DOES NOT

Which merchant a subscriber pays is not written next to the subscriber, and the encrypted note handoff between two wallets produces no transaction at all. That path is the strongest one here and the least proven: it passes 24 tests up to the prover input and has never run against devnet.

THE ONE PAIRING THIS DESIGN DID NOT BREAK ON 11 AUGUST

Erratum, 2 September 2026. The spend circuit this page and page 17 describe as not built shipped to devnet on 25 August 2026. Its instruction carries no commitment, and on 1 September a buyer wallet was measured absent from all 109 transactions reachable from a subscription it paid for. The pairing below still holds on the phone and for any note deposited before the blinding was randomised; the fee payer remains one hop from its funder on every path.

A v3 withdrawal republishes the commitment its deposit published. The unshield instruction carries `13889827819971897881` at offset 80; the shield twenty minutes earlier carries the same value at offset 8, and a second field matches at offset 40 in both. Two public fields therefore pair that deposit with that withdrawal. Page 17 gives the mechanism and what closing it costs.

THE SHIELDED POOL

A note is a leaf. Spending it burns a nullifier.

A deposit buys a note of a fixed size. Its commitment is appended to a Merkle tree as a leaf, and the value leaves the pool only when a proof shows that leaf exists and the program writes a nullifier record that can never be written twice.

HOW A DEPOSIT BECOMES A LEAF

Pools are denominated, and the shipped client configures thirteen: six SOL sizes at 0.1, 1, 10, 100, 500 and 1000, and seven USDC. A deposit of the pool's exact denomination inserts one 32-byte commitment at the next free leaf index and raises the note count by one.

The 1 SOL pool runs at tree depth 15 and keeps a ring of its last 100 roots, so a note proved against a recent root still spends. Page 06 carries the tree hash and the constants it was compiled with.

WHY A NULLIFIER, NOT A DELETION

A withdrawal never removes the leaf. It publishes a nullifier and the program creates a 41-byte record at a program address derived from the pool and that nullifier. A second spend of the same note would have to create the same account twice, which the runtime refuses.

That guarantee has a measured, permanent price. No instruction in the deployed program closes a nullifier record, so every spend strands 1,176,240 lamports of rent: 0.1894 devnet SOL across the 161 records that exist today.

MEASURED ON DEVNET, 11 AUGUST 2026

STATE	COUNT
Commitments inserted across 72 denominated pool accounts, of which unspent	325 / 164
Nullifier records on chain, against the 325 minus 164 predicted	161
Pool sizes the shipped client configures, of which ever funded	13 / 2
The two funded pools, at 0.1 and 1 SOL: leaves, unspent notes	34 / 8, 25 / 6

Read that as a consistency check, not a proof. The 72 are pool accounts on chain across two generations; the thirteen are the sizes the client offers. A further 26 commitments sit in eighteen legacy pool accounts outside the 325, and the same record type is also written by an escrow instruction. The 1 SOL pool reconciles on its own history: 25 deposits, 6 withdrawals and 13 subscriptions leave 6 live notes.

Fixed denominations exist so that the amount is not a distinguishing feature. Every leaf in a pool carries the same value, so a withdrawal discloses which pool it came from and nothing narrower, and there is no change output to correlate. What it does not hide is which deposit that withdrawal belongs to: two public fields still match across the pair, and page 17 shows the arithmetic on our own devnet transactions.

THE PROOF SYSTEM

No curve in the proof, so there is nothing to set up.

One prime field, Goldilocks, with Poseidon in the circuit and SHA-256 in the proof's own trees. Every value below comes from the branch the deployed verifier was built from, `b7-drop-aligned-checks`, and not from master.

PARAMETER	VALUE AS COMPILED
Field	`p = 2^64 - 2^32 + 1 = 18446744069414584321`
Field code	winterfell 0.10 on the prover, an independent `goldilocks.rs` inside the program
Note tree hash	Poseidon over Goldilocks, width 3, S-box `x^7`, 30 full rounds, no partial rounds
MDS matrix	Circulant `[[3,1,1],[1,3,1],[1,1,3]]`, determinant 20, all nine minors invertible
Round constants	90, byte-identical as an ordered list between prover and verifier
Proof tree hash	SHA-256, 32-byte nodes, leaf tag `0x00`, node tag `0x01`
FRI	Blowup 16, fold factor 2, terminal polynomial of 16 coefficients, degree bound 1
Queries	27 on circuits 0, 1, 2 and 4; 22 on circuits 3, 5 and 6; 22 bits of grinding
Trace	Committed on a multiplicative coset, shift `h = 7`, at no extra wire bytes

Master has no leaf tags, no degree bound and no coset shift, so three rows above.

DOMAIN SEPARATION IN THE PROOF

These are the proof's own trees, not the pool's. A leaf is SHA-256 of `0x00` and the data, a node SHA-256 of `0x01` and its two children, so no node preimage is ever a valid leaf preimage. The tag lives in the preimage, never on the wire: it cost 16,194 to 32,960 compute units across the seven circuits and changed no proof size.

NOTHING TO SET UP, PUT AS A SEARCH

No `.ptau`, `.zkey`, `.circom` or `.r1cs` file is tracked in the repository, and every surviving mention of `alt_bn128` is a comment, one recording that the dependency was removed. SHA-256 carries the trees for cost: Solana exposes it as a syscall at roughly 85 compute units per call, against about 15,000 for a software Blake3 hash.

ONE THING THE CODE CLAIMS THAT IS NOT TRUE

The constants file says its 90 Poseidon round constants came from a Grain LFSR seeded with the field, the width, the round counts and alpha. Measured, roughly indices 41 to 72 form an arithmetic progression modulo 2^64, which no LFSR emits. Prover and verifier hold the identical list, so nothing disagrees at run time, but Poseidon's case against structured constants rests on provenance, and this provenance is wrong.

SOUNDNESS, MEASURED

The honest figure is in the forties of bits.

Soundness was derived per circuit from the deployed configuration, not quoted from a formula.

COLUMN	BITS	WHAT BINDS IT
Classical, ethSTARK list-decoding conjecture	47 to 52	Fiat-Shamir field floor, 47.75 to 52.53
Classical, unconditional	42 to 46	Query term, 42.1 to 46.6
Quantum, under Grover	21 to 26	Half the classical figure, forgery being a search

per circuit, conjectured [52,50,50,47,48,47,47], unconditional [46,46,46,42,46,42,42]

WHY MORE QUERIES BUY NOTHING

Every Fiat-Shamir challenge is a single Goldilocks element, so none carries more than 64 bits. The floor is 64 minus log2(8n + w + k + 1 + folds × lde), landing at 47.75 to 52.53. The conjectured query term is 110 to 130 bits and is cut down to it. Grinding cannot lift the floor either: the nonce is absorbed last.

WHAT THE UNCONDITIONAL COLUMN RESTS ON

There the query term is the smaller quantity, so it binds and the 22 bits of grinding are load bearing. The rate is enforced, not assumed from the blowup of 16: any terminal polynomial above 1 of 16 coefficients is rejected, giving rho = 1/16, that is 4.000 bits per query across 22 or 27 queries.

THIS IS NOT A PRODUCTION SECURITY LEVEL

At 10 GH/s of SHA-256, 2^42 is about 7.3 GPU-minutes and 2^52.5 about 7.5 GPU-days. The verifier's own test file, tests/b1_deep_binding.rs, states that nothing here is publishable as a security level. Nor does anything inside a proof bind which circuit produced it: the program stores no verification key hash, so circuit separation rests on one byte in the proof buffer account. Comments at compact_proof.rs lines 41 to 45 still compute soundness as queries times log2(blowup) and reach 104 to 124 bits, which the project no longer stands behind. Reproduction is uneven: both arrays are derived in code in that test file, which sits on branch b7-drop-aligned-checks and not on master, so cloning that branch and running it recomputes them. The quotient-degree measurement that refutes the 104 to 124 has no harness committed anywhere, and has to be asked for.

WHAT RAISING IT TAKES

Engineering, not research. The unconditional column moves with the query count. The conjectured one is floor bound by the 64-bit challenge, so a wider challenge is a design direction, not a result. Both cost compute: the accepted on-chain verification used 809,662 units of the 1,399,850 a Solana instruction allows, and that ceiling is fixed.

THE ON-CHAIN VERIFIER

Accepting a proof has one cost. Refusing has two.

Verification is a Solana program, not a service. The verifier runs on devnet as 840,168 bytes of BPF, deployed 4 August 2026, and it does the work itself: the SHA-256 transcript, the DEEP composition, and every FRI fold. Solana has no proof precompile, so there is nothing to delegate to and no off-chain step to trust.

A circuit 1 proof is 68,881 bytes, more than one transaction can carry, so it is uploaded to a buffer account in chunks: one subscription run spent 148 of its 172 transactions on chunk writes.

On 4 August 2026 at 19:03 UTC the deployed verifier accepted a circuit 1 proof on devnet for 809,662 compute units of the 1,399,850 the instruction requested. That budget is fixed by the runtime: 1,399,850 is Solana's hard ceiling of 1.4 million units per instruction, less the 150 units the ComputeBudget instruction itself costs. The headroom is 1.73 times and no configuration raises it, so a check that grew by three quarters would stop fitting. The subscription path splits the work instead: one run spent 1,913,751 units on two proofs.

THREE VERDICTS

INPUT	VERDICT	COMPUTE	WHERE IT DIES
Honest proof, circuit 1	Accepted	809,812	All eight steps pass
Proof with one byte flipped	Rejected, InvalidProof 6003	542,150	After step 3, in the DEEP and FRI arithmetic
Public input tampered with	Rejected	19,777	Before that arithmetic begins

Whole-transaction figures; 809,662 of the honest 809,812 ran inside the program. Program `DGY37k3Jt7cbrfNa9rxyLZVcFB7S7A2NqtVpkh9fWQvs`, honest run `2sLVyzPWseoVuCwPoDaZWUPRB8u8NXmmCGTVnBvZnEiJXHkPT1vAzMKMwhhHbPaXPDYSdTw39drzVCgUN49jZiBR`

The informative result is not that both bad inputs failed. It is that they failed by different mechanisms, at costs 27 times apart. One flipped byte leaves a proof that is still well formed, so it survives the cheap checks and dies later inside the arithmetic that binds the openings to the commitment. A tampered public input dies before that arithmetic begins, at 2.4 percent of an honest run. One generic failure at one cost would have been equally consistent with a verifier that refuses everything, which is the case an accepted proof alone cannot rule out.

WHAT THIS TABLE DOES NOT CARRY

Only the acceptance has a transaction signature. The two rejections are our own simulations against the deployed program, reproducible by rebuilding the harness rather than by lookup. All three ran through the verify_uniform entrypoint, which our own notes flag as a soundness liability because it identifies the circuit by probing instead of requiring the caller to name it, and which the subscription flow does not use.

STEALTH ADDRESSES

An attacker has to break both key exchanges, not one.

Each payment lands at a fresh one-time address. The shared secret behind it comes from two key exchanges at once: X25519 Diffie-Hellman, and ML-KEM-768, the lattice mechanism standardised as FIPS 203.

The two 32-byte secrets are concatenated and passed to HKDF-SHA256 as key material with an empty salt, and the 32-byte output is the stealth seed. It stays unpredictable unless an adversary recovers both inputs, so a quantum break of the curve still leaves the lattice problem, and a lattice break still leaves the curve. The code is `deriveHybridSharedSecret` in `packages/specter-sdk/src/utils/crypto.ts`.

SIZES, IN BYTES

ML-KEM-768 public key	1,184
ML-KEM-768 secret key	2,400
ML-KEM-768 ciphertext	1,088
HKDF info field	53
v2 meta-address	1,249
v2 announcement	1,153

BINDING THE ANNOUNCEMENT

The info field is not a bare constant. It is the 21-byte label `p01-hybrid-stealth-v2` then SHA-256 of the ephemeral public key concatenated with the KEM ciphertext.

Without it, two announcements in flight share an info field and an attacker can lift the ciphertext from one into the other. With it, a substitution changes the transcript hash, so the derived secret changes and the swap opens nothing.

ENFORCED AT THE SEND AND PARSE BOUNDARIES

A recipient meta-address carrying no KEM public key throws rather than degrading, and a 65-byte classic announcement is refused unless the caller asks for the legacy path by name. On devnet the stealth program owns 20 accounts of 1,220 bytes, all with a populated X25519 ephemeral key and 19 with a full 1,088-byte ciphertext. The twentieth is all zero, an abandoned chunked upload. HKDF is checked against RFC 5869 vectors, and 99 of 99 tests pass across the stealth, quantum and crypto suites. The same construction is wired independently into the mobile app and the browser extension.

WHAT THIS DOES NOT COVER

- The recipient path keeps an ungated fallback. With no KEM secret key and ciphertext, derivation silently uses the X25519 secret alone.
- The ML-KEM-768 implementation is a third-party library. Its parameters match FIPS 203, measured, but nothing here is FIPS validated or tested against the NIST vectors.
- All 20 accounts have `claimed` set to 0. The send leg runs on chain. The claim leg has never executed on devnet.

SUBSCRIPTIONS AND THE MERCHANT LEG

The subscriber funds the term up front. Anyone can pay the merchant.

A plan is one account per subscription, owned by the shielded pool program. Signup escrows the pool's whole denomination into it, and collection is a separate instruction that anyone may send.

COLLECTION NEEDS NO MERCHANT KEY

ClaimPeriod pays a merchant and needs one signature, which does not have to be the merchant's. On devnet the payout address sits at account index 1, outside the signer range, and the vault paid it 1,002,721,360 lamports: the 1,000,000,000 escrow plus the 2,721,360 rent floor of a 263-byte vault, and the account was then deleted.

That address is written into the vault at signup, not supplied by the caller, so a lost merchant key cannot strand revenue and no caller can redirect the payment.

ENTITLEMENT IS ONE ACCOUNT READ

A merchant gates access by reading one account, with no webhook to receive. Their own book is not hidden: see page 17. The check ships as hasActiveVaultAccessForVault in packages/merchant-sdk/src/vaults.ts, seven ordered checks covered by 62 passing tests.

It deliberately does not trust the vault's own is_active flag. Of the 18 vaults live on devnet on 1 August 2026, 14 had run past the periods they were funded for, and all 18 still reported is_active true.

MEASURED ON DEVNET, 11 AUGUST 2026

STATE	VALUE
Subscription vaults the pool program owns, in three sizes from 263 to 361 bytes	18
Signups on the current build, at 1,000,000,000 lamports each	4
Merchant services in the on-chain registry, every one registered by us	6
Compute units to create a vault, and to collect and close one	28,918 to 34,824 / 5,871

The collection above was signed by a key that was not the merchant's, but it was still ours, so no outside party has yet exercised the path. Two paused vaults also cannot be collected until their owner produces a resume proof.

A SUBSCRIPTION IS A PREPAID ONE-WAY ENVELOPE

Signup escrows the pool's whole denomination, so what is locked is the denomination and not the price entered: 0.1 SOL at the smallest pool. In the bytes dumped from the chain, cancel_normal and cancel_private_stark return zero occurrences, while ClaimPeriod and PauseNormal each return one. A subscriber can pause. No instruction in the deployed program can return a lamport to the subscriber.

THE RELAYER

It moves the fee payer. It does not move the correspondence.

A relayer pays the fee and signs the submission, so the user's own key is absent from the fee payer and signer fields. That buys one property precisely, and naming what it is not is the more useful half of this page.

WHAT IS DEPLOYED

The program is live and executable on Solana devnet, declared at `programs/p01_relayer/src/lib.rs:33`. It takes a job whole or in chunks when the payload will not fit one transaction, and nodes register, stake and post a heartbeat.

Job submission has run on chain. The two devnet links the project published as Stealth and as ZK Pool are one and the same `SubmitJob` transaction, dated 29 May 2026.

WHAT IT CANNOT CHANGE

A v3 withdrawal republishes the commitment its deposit inserted. That pairing sits in the instruction data, not in the signature, so it survives any change of fee payer. Page 17 carries the consequence in full. *Erratum, 2 September 2026: the circuit 7 spend, on devnet since 25 August, carries no commitment; what a relayer cannot change there is that the fee payer it supplies has a funding history of its own.*

Nor does it remove the observer. Proof chunks reach the chain as a loop from one ephemeral signer, up to 145 transactions per proof on the mobile path, through one RPC provider that sees the address and the network origin.

MEASURED ON DEVNET, 11 AUGUST 2026

program	2okhzLVr6FEq5jP19KT6VurcSutx2zE4RhkRamrk5WpW
heartbeat	4,430 CU, one every 30 seconds, 120 per hour
continuity	2,000+ consecutive, 0 failed, 16 h to 12:42 UTC
state	116: 2 nodes, 111 chunks, 1 config, 1 refund job, 1 unidentified

The trust is availability and observation. The relayer learns the payload and the origin of every request it carries, it can decline or delay one, and the two registered nodes have no demonstrated operator outside the project. The web pay interface therefore states in its own copy that its path has no relayer, and a test fails if that wording ever softens.

THE HEARTBEAT IS LIVENESS, AND TWO DEFECTS ARE OPEN

120 transactions an hour at a flat interval is a cron proving a process is alive. It is not usage, throughput or demand.

Every `RelayChunk` account the program creates is permanently unclosable at 1,037 bytes and 0.0081084 SOL of rent each, and the source names its own missing fix at `expire_pending_job.rs:17`. The running build also dates from 14 July 2026 while its correction went to a repository not wired to this deployment, so a statement about relayer behaviour describes source rather than the live service.

PROGRAMS ON DEVNET

Eight programs are live at the address they declare.

Fifteen program directories sit in the repository. Eight of them resolve to an executable devnet account at the exact address their own declare_id! names, each with a live programData record, read on 11 August 2026.

PROGRAM	PROGRAM ID	ROLE	SUCCESS
zk_shielded	GbVM5yvetrSD194Hnn1BXnR56F8ZWNKnij7DoVP9j27c	Shielded pool and notes	2026-08-06
p01_stark_verifier	DGY37k3Jt7cbrfNa9rxyLZVcFB7S7A2NqtVpkh9fWQvs	Verifies STARK proofs	2026-08-06
p01_relayer	2okhzLVr6FEq5jP19KT6VurcSutx2zE4RhkRamrk5WpW	Relay jobs and heartbeat	2026-08-11
specter	FgKhXakZGsd4PdiGgACYy8gwj1JLMYA691yQr2PhUNfL	Stealth payment accounts	2026-07-24
p01_registry	QaQwpvBi1EQpevNE21D2oNBHFsLtoLwa7aXH26zRhQB	Stealth keys and merchants	2026-08-04
p01_quantum_vault	9yVr79XkwGabckVxedz4UH78twzkgmGqXHBAX7vfJvYv	Winternitz withdrawals	2026-04-16
p01_quantum_wallet	D7RBAFcMq2gGddwFvabZKuJB1eWZFvESVrcJ3i15FFtz	STARK-gated custody PDA	not read
p01-fee-splitter	UdxXEvcAzmGsqUtoBgnNkbmfnky4En2kLxNnsVQU5BM	Fee split on a transfer	not read

Success is a successful invocation read from the chain on that date, and for the first five it is the newest one. The vault figure is its 2026-04-16 Winternitz withdrawal; the last two were not measured. Anchor.toml disagrees with four of its thirteen devnet entries and the README table with three of its eleven rows, so declare_id! is the only address this document treats as authoritative.

DECLARED, AND NOT DEPLOYED

p01_zkspl	AY38smtdsnhmfMCzmnDEefiKCeRTkEPrFXHydAF2FuCT
subscription	3eDvPJTK2gryh3GhjFgwz94iBsE3hsqZL9ChAFyiBThW
stream	C92xDDAtd21ED3MitZJ9dhuyGeig5xVx8Dgg6qrxA3vx
whitelist	5PSYrjBKke4gj8BgBgRKZNXgjmLCnojZ5yuDqUvPiG33

Those four addresses return null on devnet: nothing is deployed where the source says it is. The recurring payment path that does run lives inside zk_shielded, and streaming exists only as an account type inside specter, of which zero instances exist on chain. Three further directories are outside this count: a separate project, a removed capability, and one that calls itself undeployed.

ONE KEY CAN REPLACE ALL OF IT

Every program deployed on devnet, eighteen counting superseded builds, is upgradeable by one key, `7gWpzSZALYz3Um8G7yUxaT6Av2tvw1Cn6VAhSZSB6QmU`. No multisig, no timelock, none immutable: whoever holds that key can replace the bytecode behind any address above. Nor is the verifier's bytecode proven to come from this repository's source: its deploy commit is not on master. The 878,208 zk_shielded bytes on chain do match a local build from a commit on master, byte for byte.

PACKAGES

Twelve packages install from npm. Seven days is not a track record.

The client surface is published under the @protocol-01 scope on the public registry. Nothing is published under any Styx name.

PACKAGE, UNDER @PROTOCOL-01/	LATEST	EXPORTS	WHAT IT IS
privacy-sdk	1.0.4	66	Shielded pool client, 14 subpath entries
privacy-toolkit	1.0.4	22	Commitment and nullifier primitives
specter-sdk	0.4.3	210	Stealth addresses, hybrid post-quantum
stark-prover	0.1.3	18	The WebAssembly proving blob
merchant-sdk	0.1.3	80	Subscription vaults and merchant claim
p01-js	0.3.2	143	Umbrella client over the packages above
zk-sdk	1.0.2	32	Proof inputs and verification helpers
zkspl-sdk	0.1.3	30	Confidential SPL layer client

Four more names carry the scope: an authentication helper, an RPC helper, one left over from an integration removed in July 2026 and not yet deprecated, and streams 0.1.0, an orphan with no directory in the repository. Four packages that are in the repository are unpublished.

INSTALLED AND TYPE CHECKED

All eleven repository-backed packages installed into an empty project on 11 August 2026: 203 transitive packages, zero errors, every one loading under both require and import, all shipping type declarations.

LICENCE

The root ships MIT and ten of eleven declare MIT in the package.json npm serves. Three defects remain: two ship no LICENSE file, one declares no field, and GitHub reports the repository licence as Other.

The prover blob inside stark-prover 0.1.3 is 229,640 bytes with sha256 51a947e3, and the same size and hash come from the working tree and from packages/stark-prover/wasm/p01_stark_bg.wasm on GitHub master. specter-sdk 0.4.3 ships the identical blob.

THE DOWNLOAD COUNTER IS NOT ADOPTION

Nine were published inside a 132 second window on 4 August 2026, two on 6 August, two on 11 August, so publish age runs from zero to seven days. The registry reports 5,012 downloads across the eleven for the thirty days to 10 August, but npm counts mirror fetches and scanner traffic as installs, so the figure is noise rather than traction.

CLIENTS

The installed Android build cannot withdraw.

Three clients exist: a web app in production, a browser extension, and an Android build. The web app and the extension call the 229,640 byte WASM prover, whose SHA-256 the deployed verifier's record names as the proof source it accepts. The Android build does not, and the admission below says why.

CLIENT	WHAT IT CARRIES	TESTS
Web, protocol-01.dev	Shield, withdraw, subscribe. Eight production routes answered on 11 August 2026 and the footer reads devnet only, not audited. Proof buffers are sized to the real proof rather than padded.	527
Browser extension	Stealth sends, encrypted notes, relayer transport, merchant entitlement. ML-KEM-768 is imported in three of its own files. The packaged zip in the repository is 0.1.1 from March 2026 and is in no store.	170
Android 1.0.3	Full pool client. Every proof is padded to 145,000 bytes, so one upload is 145 chunk transactions and 1.0107 SOL of buffer rent, returned when the buffer closes.	400

1,407 tests passed across the three clients, the merchant SDK and the Rust suites, each suite run on 11 August 2026. That is a dated tally, not a claim about the tree as it stands today.

DESKTOP, WASM UNDER NODE V24.14.0, ONE MACHINE

circuit 1	153 ms
circuit 3	729 ms
all seven	7.6 s

A withdrawal needs circuits 1 and 3, so roughly nine tenths of a second of proving. Two passes agreed to within a few percent, but no benchmark harness is committed, so this is one run on one machine and not a published benchmark.

ON DEVICE, NEVER MEASURED

Pure proving time on a phone has never been measured. The repeated figure of more than 180 seconds is a client timeout in the mobile prover provider, raised from 60 seconds after circuits 1 and 3 timed out on a test handset, so it bounds the whole flow and not the prover. The instrumentation that would log a real prover figure exists only as a proposal.

WHY THE INSTALLED BUILD FAILS

It carries a 213,254 byte prover blob from before the coset change, and the verifier redeployed on 4 August 2026 requires the 229,640 byte blob: the device logged a deserialization error, code 6004, at 3,018 compute units. That reading was taken the day before the current verifier went up, so it is consistent rather than conclusive. There is no over the air update path, so the fix is a rebuild and a manual reinstall, and 1.0.3 exists only as an installed app on one phone, with no current APK to download.

DESIGN DIRECTION

The old page spent its brightest colour on its least defensible number.

Measured on 11 August 2026, the live homepage renders a cyan `0` labelled `Traces left on-chain`. Page 17 explains why that figure is wrong. Design gave it the largest type on the screen.

TWO VOICES, AND ONLY TWO

One serif carries every statement. One monospace carries every piece of evidence: addresses, signatures, byte counts, compute units. A reader can tell an assertion from a measurement without reading either.

HAIRLINES, AND A SEAL

The palette is near-monochrome: two warm paper tones and one ink. Structure is 0.5pt hairlines and unfilled borders, not shadows. Cyan survives as a seal on a section tick and a verified mark, never as a surface and never on a claim.

REMOVED	REASON
`Orbitron display face`	A third voice, competing with the evidence and legible as a console rather than a ledger.
`Cyan to pink gradient, glow, glitch`	Emphasis a reader cannot check, heaviest exactly where the copy was weakest. With cyan reduced to a seal, the pink had nothing left to mark.
`The “01” cover mark`	The project was renamed on 8 August 2026 and is no longer named for a numeral.
`01-miku.png`	One 25,904 byte fan-art PNG served as og:image, twitter:image, favicon and Apple touch icon at once, so every shared link rendered it.

THE ONE GESTURE KEPT

A canvas of slow horizontal lines drifts behind the first screen, the river the protocol is named for. It pauses on a hidden tab, stops when it leaves the viewport, and renders one static frame under reduced motion. Nothing else moves: no reveal on scroll, no counter that climbs.

WHY THIS IS INK ON PAPER

The site is warm off-white on near-black; this document inverts it. Eighteen near-black A4 pages are punishing to print and heavy on a phone, and they read as a site exported rather than a document written. An annual report is printed. Nothing else about the identity moves.

NONE OF THIS IS LIVE YET

On 11 August 2026 the production site still points og:image, twitter:image, the favicon and the Apple touch icon at that same PNG, and still sells “anonymous interactions” in four metadata tags, so the claim survives a link preview even after the visible copy is fixed. The replacement system exists as six uncommitted routes: a clone of the public master branch does not reproduce it.

SECURITY AND THREAT MODEL

A threat model is worth reading for what it concedes.

Styx defends what a note contains, and derives a fresh address per stealth payment. It does not defend the link between a deposit and its withdrawal.

WHO SEES WHAT TODAY

PARTY	WHAT IT LEARNS
Chain observer	The deposit, the withdrawal, and the pairing between them. The unshield instruction has to republish the deposit's commitment, so no client fix hides it.
Anyone, on a send	Sender identity is not hidden on the send path at all. That path carries no relayer on the web client, so the payer is the fee payer and the signer, and the amount is public with it.
RPC provider	The client address, the signer, and every proof chunk: 145 uploads per proof on the mobile path.
Relayer	Whatever it is handed, which it writes to chain: 111 chunk accounts of 1,037 bytes sit on devnet.
Recipient	The note handed over. Its secrets derive from no wallet seed, so the local store is the only copy.

IF A KEY LEAKS

A stealth viewing key spends. The derivation returns a full Ed25519 keypair from the viewing secret, so a watch-only key does not exist yet.

All 18 deployed programs are upgradeable by one key, with no multisig and no timelock: the largest operational risk here.

The Winternitz vault key is one-time. Signing twice under one public key is what breaks it.

Nothing above was found by a reviewer outside the project, because there has not been one.

THE POST-QUANTUM POSITION

Transaction authorisation stays Ed25519, because Solana verifies nothing else. A protocol claiming a fully post-quantum Solana transaction today is overstating.

Post-quantum where it can be: no curve and no pairing in the proof, only Poseidon over Goldilocks and SHA-256, and a stealth agreement that mixes X25519 with ML-KEM-768.

Under Grover a forgery is a search, so the measured 42 to 46 bit unconditional margin roughly halves, to about 21 to 26 bits. That is not a publishable security level.

NO CEREMONY, SO NOTHING TO TRUST AND NOTHING TO CLAIM

There is no trusted setup. The verifier contains no alt_bn128, bn254, pairing or Groth16, and no .ptau or .zkey is tracked. No Powers of Tau ceremony took place. Two published artefacts still imply one: @protocol-01/privacy-sdk 1.0.4 lists snarkjs as a dependency, and the @protocol-01/zkspl-sdk README describes ceremony output. Both are text to correct, not running code.

WHAT THIS DOES NOT DO YET

A deposit can still be matched to its withdrawal.

On the v3 path the withdrawal has to republish a value the deposit already emitted. That join is deterministic, so the effective anonymity set is one and no amount of pool growth dilutes it.

THE MECHANISM, MEASURED

UnshieldDenominatedStarkV3 takes stark_commitment as a public argument, because the program rebuilds sha256(nullifier || stark_commitment) to match the verified proof buffer. On the devnet pair we ran on 4 August, that u64 sits at bytes 80 to 88 of the withdrawal and bytes 8 to 16 of the deposit, twenty minutes apart.

WHY IT WAS STILL OPEN, AND WHAT CHANGED

No client-side change closes it: the value must be public or the proof cannot verify. The fix is a spend circuit that proves Merkle membership without naming the leaf, specified in docs/C7_SPEND_CIRCUIT_PLAN.md. **Erratum, 2 September 2026:** it shipped to devnet on 25 August 2026 as circuit 7, whose instruction carries no commitment. The join above survives on the phone and for notes deposited before the blinding was randomised. What no circuit removes is the fee payer, one hop from whoever funded it, so private in this document still means confidential amounts, shielded balances, and an unlinkability that is measured per spend rather than guaranteed.

No external review	Zero audits. A review able to cover a hand-written verifier was priced internally at 153,000 to 263,000 EUR.
Devnet only	Nothing executable on mainnet. The seven ids under Anchor.toml [programs.mainnet] resolve to no program at all.
No outside party	Six registry services, one owner key, and that same key is the authority that stamps the verified flag on them. Subscriber count is zero on all six.
Anonymity set	Busiest pool: 34 leaves, 8 unspent notes. Eleven of the thirteen denominations the client offers hold none.
Merchant privacy	Retailer address in the clear at byte offset 42. One unauthenticated RPC call returns all 18 vaults across 9 retailers.
Transient float	About 1.77 SOL of buffer rent for a web withdrawal and 2.02 SOL on mobile, where padding fixes both buffers at 1.0107 SOL. Refunded on close, though 44 buffers are still stranded on the verifier. One per operation, never shared.

INCLUDING THE PERMISSIONLESS CLAIM

Anyone can collect a merchant's revenue without the merchant's key, and that has run on devnet. The caller was not the merchant, but it was the founder's own wallet: one signer across that claim, the program upgrade and all six merchant registrations. No independent party has run any part of this.

HOW TO CHECK THIS

Every figure here has a command behind it.

Each row names a claim made earlier and the exact target that settles it. Every RPC call goes to https://api.devnet.solana.com except the one row that names mainnet-beta, and all of it was read on 11 August 2026.

CLAIM	REPEAT THE MEASUREMENT
Eight programs run on devnet at the address their own source declares	`grep -rn 'declare_id!' programs/` gives fifteen addresses. Submit them in one `getMultipleAccounts` call and count the non-null programData.
A STARK proof verified in one instruction, 809,662 CU of 1,399,850	`getTransaction` on signature A. Read the eight step logs and the consumed-units line.
A deposit and its withdrawal share a public field, so pairing is trivial	Decode the instruction data of B and C. Bytes 8 to 16 of the shield equal bytes 80 to 88 of the unshield: `13889827819971897881`.
The busiest pool holds 8 unspent notes over 34 leaves	Read `HfSsGRgVFJ`... the 0.1 SOL pool, with `getAccountInfo`: next_leaf_index and note_count, per `pool_v3.rs`. The 1 SOL pool `6NUS4E5PhQ`... holds 6 over 25.
The prover on npm emits the bytes the deployed verifier accepts	`npm pack @protocol-01/stark-prover`, then hash `package/wasm/p01_stark_bg.wasm`: 229,640 bytes, sha256 `51a947e3`.
Soundness is 42 to 46 bits unconditional, not the 104 to 124 the source claims	Read the block above `B2_CONJECTURED_FORGERY_BITS` in `b1_deep_binding.rs`, branch `b7-drop-aligned-checks`.
The live homepage still publishes claims this document contradicts	`curl -s https://protocol-01.dev | grep -i 'nothing they can trace'` returns three hits.

THE THREE SIGNATURES, IN FULL

```
A  proof accepted by the verifier, 2026-08-04
   2sLVyzPWseoVuCwPoDaZWUPRB8u8NXmmCGTVnBvZnEiJXHkPT1vAzMKMwhhHbPaXPDYSdTw39drzVCgUN49jZiBR
B  1 SOL shielded into the pool, 2026-08-04
   qmsMCWai56avCgHki9KtXXp2egNbWqSiqmpSxdrGuVzV1h22ioME3WSsnRci5eWzRzyrrVtJPqqNAGuBAsNQovf
C  0.995 SOL withdrawn to a fresh address, 2026-08-04
   5QptxwyyrR2Qfe9DxDY4LXiLvVaouKFrkyVYWh4iNztpBbPqYn4pCgtJ1pkLA88VMq4Ub5LJcEqwzBxTnRWgBJ18
```

THREE THINGS A READER CANNOT REPEAT

The deployed verifier's bytes match a hash recorded in this repository, but the commit behind them was never pushed, so that custody chain ends with us. The forged-proof and tampered-input rejections are simulations with no signature to look up. And no on-device proving time can be repeated: the 180 second figure quoted for it is a client timeout constant, and the instrumentation for a real one exists only as a proposal.