PROTOCOL RESEARCH Public testnet experiments

Post-Quantum Authorization for Native-Asset Vaults

EVM and Solana vaults controlled by a Parano1d secret. Public withdrawals complete on Base Sepolia in 14.46 seconds and Solana Devnet v1 in 18.08 seconds.

Ignotus Nemo 08 Nov 2026 23 min read

ABSTRACT

A Parano1d secret controls native-asset vaults through request-bound proofs of authority. Public Base Sepolia and Solana Devnet experiments complete deposits, staged verification and atomic withdrawals under ordinary transaction limits. Base uses ten authorization transactions with L2 receipt confirmation in 14.46 seconds. Solana v1 uses 88 transactions with confirmation in 18.08 seconds and recovers from packet loss and reordering in 22.34 seconds. Domain-separated derivation creates each network's authority branch. Each vault pins a public commitment and retains a persistent nonce. A freshly randomized proof authorizes the request, while a relayer delivers the verification steps. The proof backend uses hiding FRI over a small prime field with SHA-256 and Keccak commitments. Authorization safety follows under its knowledge, privacy, derivation and request-binding conditions, together with correct host-chain execution. Quantitative quantum chosen-request security and repeated-use privacy for this external-vault profile are cryptographic research targets. Source code, signed transactions, public receipt cross-checks and mainnet fee quotes accompany the experiments.

The authorization problem

Conventional wallets authorize spending with an elliptic-curve private key. A sufficiently capable quantum adversary can recover such a key from its public key using Shor's algorithm.[11] Account abstraction offers another enforcement point: executable account rules can require a proof of authority before moving funds. This makes the account's withdrawal predicate the central design choice.

Parano1d signatureless authorization supplies a reusable secret and a public authority commitment. A proof capsule binds knowledge of that secret to a canonical request. Its published interface separates proof verification from application rules for context, policy and freshness. It also supports recursive proof composition.[1] This note applies that boundary to native-asset storage on EVM-compatible networks and Solana.

The proposed wallet uses a single Parano1d secret to manage separate native accounts. A Parano1d EVM contract holds the selected network's native asset. The measured EVM example is ETH on Base. A Solana program-owned account holds SOL. Each account accepts a withdrawal only after its on-chain Parano1d verifier accepts the corresponding authorization and its execution rules accept the request.

Generating ordinary ECDSA or Ed25519 keys from the same secret would preserve those signature schemes' security assumptions. Fresh receiving addresses can reduce some exposure, but a first Ethereum spend exposes the signing public key through its recoverable signature.[12] A conventional Solana address already exposes its public key.[7] The architecture below places proof authorization directly in the account's spending rule.

From a Parano1d secret to native addresses

Parano1d secret denotes the wallet's locally held, high-entropy, 256-bit secret *S*. For network domain *N*, account index *i* and protocol version *v*, the extension derives a separate vault authority secret:

$$s_{N,i} = \mathrm{KDF}(S, \mathrm{Encode}(\text{"parano1dx/authority"}, v, N, i))$$
$$C_{N,i} = \mathrm{Commit}_{\mathrm{profile}}(s_{N,i})$$

Typed, length-delimited encoding fixes every derivation field. C is a 32-byte public authority commitment.

The executable profile uses HKDF-SHA-256 with salt `Parano1dX/root-derivation/v1`. Its `info` is the authority domain, a 32-byte network identity and the eight-byte little-endian account index. The 56 output bytes supply fourteen four-byte words, each mapped into the KoalaBear field. HKDF supplies the extract-and-expand interface and contextual key separation.[2] Its use here requires the corresponding quantum security assumption.

The authority relation permutes those fourteen field elements and ten fixed domain-capacity elements with Poseidon2 over KoalaBear. SHA-256 of the authority domain and twelve canonical output elements gives the 32-byte commitment *C*. This external-vault profile has its own authority relation and proof parameters. The native account address is derived from *C* and the authenticated deployment manifest. The commitment identifies the authority, and the native address receives the assets.

The deployment manifest defines *N* from the intended network and deployment identity. EVM requests check the chain ID and pinned vault configuration. Each L1 or L2 receives a separate network domain and authority branch, even when its factory and account addresses match those on another chain. Solana uses a deployment-pinned network domain. Its program checks that configured domain, and the wallet authenticates the intended cluster's genesis identity and program deployment.

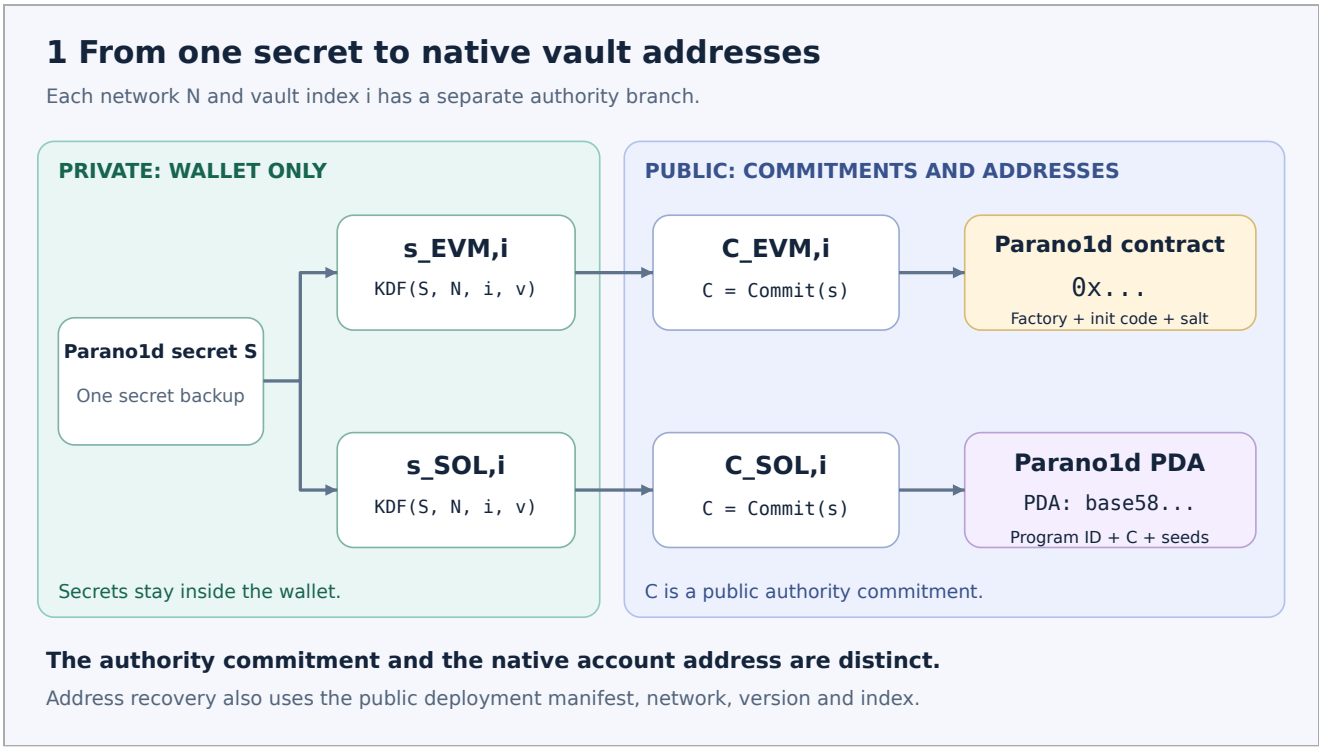


Figure 1. The Parano1d secret stays in the wallet. Independent branches produce public authority commitments and deterministic native vault addresses. The EVM branch represents one selected network, with separate derivation for each L1 or L2. The public deployment manifest is also needed to reproduce those addresses.

View full-size SVG · PNG

EVM: deterministic contract address

A Parano1d vault factory deploys a standalone contract using CREATE2. Its constructor fixes the authority commitment, verifier and parameter identity. Request reconstruction binds the host chain ID. For a target network implementing the CREATE2 semantics specified in EIP-1014, the address is:[3]

$$A_{EVM} = \text{last20}(\text{Keccak256}(0xff \,||\, F \,||\, salt \,||\, \text{Keccak256}(init_code)))$$

F is the authenticated Parano1d factory. init_code includes constructor arguments that bind C and the selected configuration.

The wallet derives the salt from typed deployment metadata. It authenticates the deployed runtime code and initialized state before displaying the contract as ready to receive funds. Anyone may pay for deployment of exactly that account. Initial authority is determined by the committed constructor data.

Solana: deterministic program-derived address

The Parano1d vault program derives its PDA from the program ID, the fixed `Parano1dX-vault` seed and the authority commitment. The network domain and account index enter the derivation of that authority from the Parano1d secret. The canonical bump completes the PDA derivation. A PDA is off the Ed25519 curve and has no corresponding signing private key. Solana grants the deriving program PDA signing privileges during cross-program invocation.[4]

Derivation yields an address. An initialization instruction then creates or safely allocates the account, assigns it to the Parano1d program and installs its authority state. The wallet verifies the network, program and initialized account before accepting deposits.

Where code and assets live

Both constructions keep the native asset in an account governed by Parano1d authorization rules. Their code and storage layouts follow the host chain's account model.

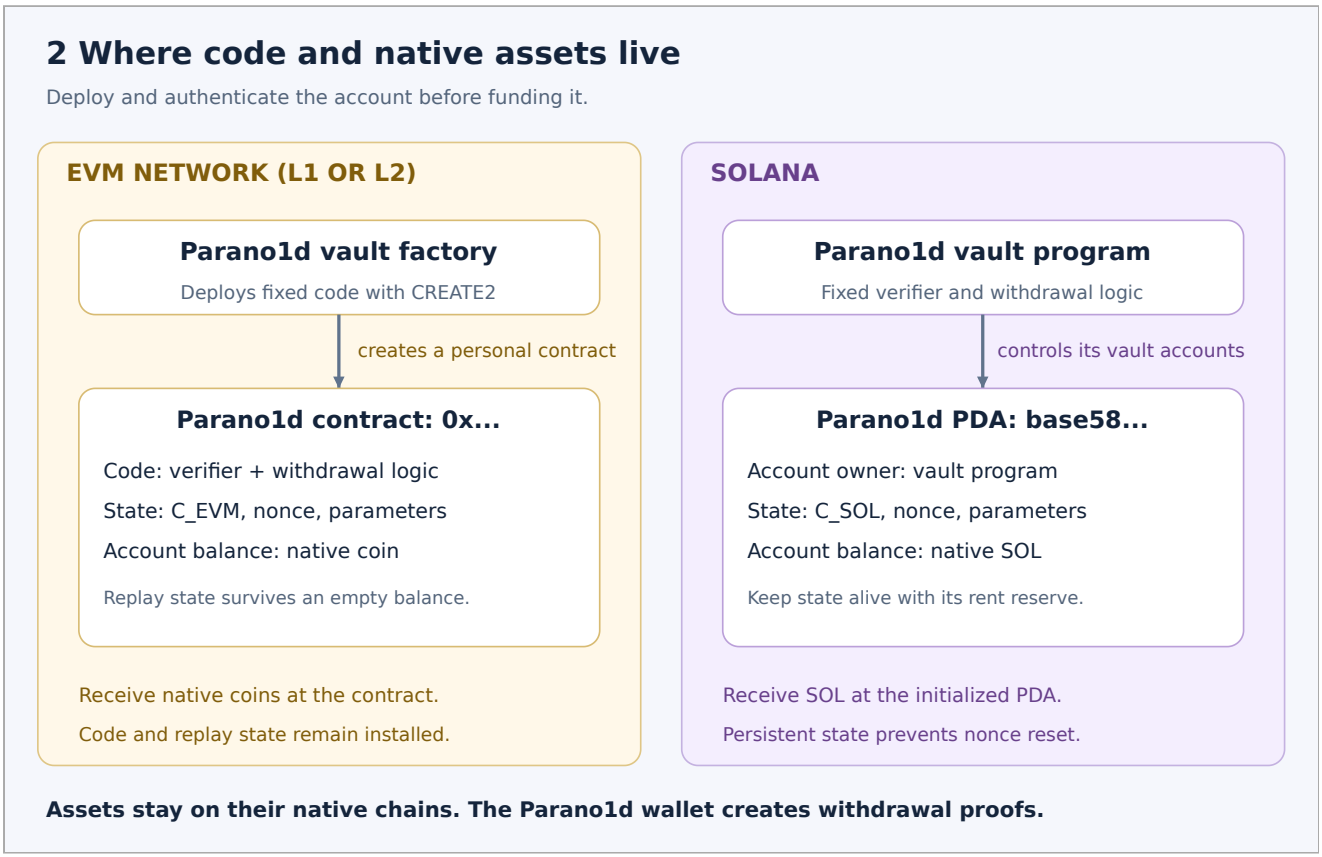


Figure 2. A compatible EVM L1 or L2 stores code and authority state in a personal Parano1d contract holding that network's native coin. Solana uses shared Parano1d program code and a personal program-owned Parano1d PDA. The Solana replay state remains rent-funded throughout the account's lifetime.

View full-size SVG · PNG

Parano1d vault contract on EVM networks

The contract stores C, a monotonic nonce and fixed protocol parameters. A simple receiving function accepts the network's native coin. The withdrawal entry point checks the proof and execution constraints before transferring that coin. The baseline uses immutable withdrawal logic and fixed verifier parameters, with every value-releasing path subject to the same authority rule.

The deployment profile also requires host-chain rules that protect a deployed contract from ordinary transaction origination at its address. Ethereum provides this protection through EIP-3607.[5] EIP-7702 account delegation retains a separate ECDSA authorization mechanism.[6] The proposed vault therefore

uses a directly deployed contract with a persistent code identity. The selected EVM chain's account rules must preserve that identity and its spending predicate.

Portability is defined by this deployment profile: required CREATE2 behavior, supported bytecode, native-value transfers, account protections and sufficient verification resources. Each network pins its own VM revision, limits and finality conditions. L2 deployments also declare their sequencing, data availability, settlement and protocol upgrade conditions. Optimism and Arbitrum document EVM-compatible execution together with network-specific differences in fees, block context and cross-chain calls.[13], [14]

Parano1d vault PDA on Solana

The shared program implements verification and withdrawals. Each PDA stores an account discriminator, version, authority commitment, parameter identity and nonce. Its lamport balance contains the SOL and the reserve needed to retain its state. Solana's owner field identifies the program permitted to modify account data and debit its lamports.[7]

The withdrawal instruction validates the PDA seeds, canonical bump, program ownership, data layout, writable accounts and recipient. It then debits its own account and credits the recipient under the runtime's account rules. Every successful withdrawal preserves the persistent state reserve. The baseline fixes the shared program's code identity by revoking its upgrade authority after deployment validation.[8]

The withdrawal protocol

The Parano1d wallet constructs a canonical request *R*. Its encoding fixes types, field lengths, field order and interpretation. The request contains the full action that the account will execute.

Table 1. Authorization context carried by a native-asset withdrawal.

Field	Binding purpose
Protocol and network	Version and authenticated deployment domain identify the request's interpretation and intended chain.
Vault and parameters	The native account address and fixed verifier configuration identify the exact authority endpoint.
Nonce	The current vault counter permits one execution in the canonical state history.
Operation and asset	The native-transfer operation and asset identifier select the chosen EVM network's native coin or SOL.
Recipient and amount	Exact native recipient bytes and integer value define the transfer.
Expiry and fee terms	A defined block or slot deadline bounds validity. Explicit fee recipient and reimbursement terms bound any payment from the vault.

The account reconstructs the request from the actual operation arguments and authenticated account context. It reads C from its own state and computes the public statement:

```
m = H_intent(domain, Encode(R))
x = (m, C)
π = Authorize(s_{N,i}, x)
```

The verifier derives x itself. The local prover uses fresh randomness for each authorization.

A relayer receives R and π, pays the host-chain transaction fee and submits the operation. The relayer can be a third party or a separate fee-paying account controlled by the user. The proof determines withdrawal authority. The host transaction's signature supplies delivery and fee payment.

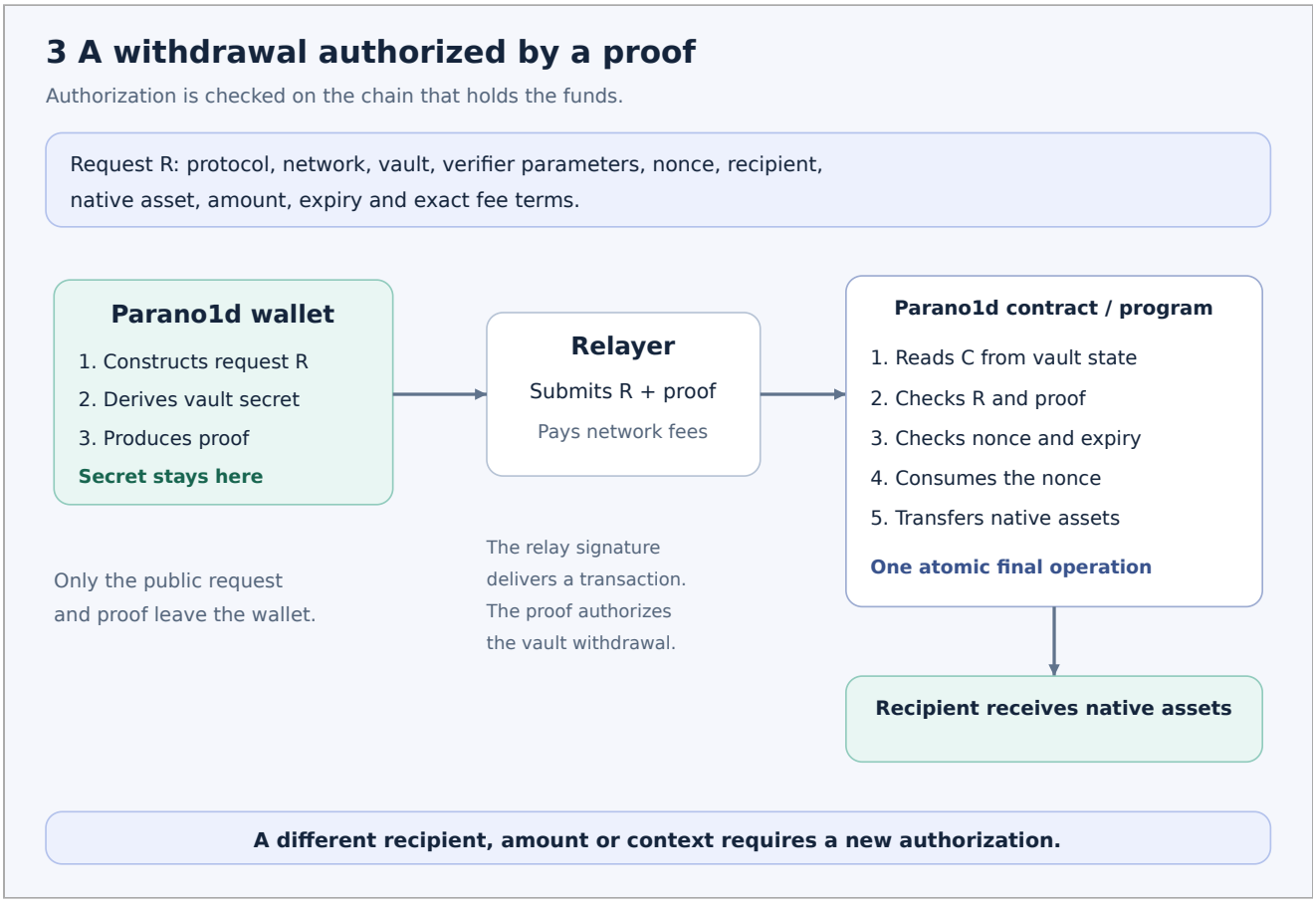


Figure 3. The Parano1d wallet produces a public proof of a specific withdrawal. The Parano1d contract or program authenticates the request, consumes its nonce and transfers native assets atomically. Staged verification may precede this final operation under the conditions in Section 7.

View full-size SVG · PNG

Atomic execution rule

```
Withdraw(R, proof):
    check canonical request and fixed protocol configuration
    check network, vault identity, operation and native asset
```

```
    check R.nonce == state.nonce and request has not expired
    check recipient, amount, fee terms and available balance
    m := IntentID(R)
    require CheckAuth((m, state.C), proof)
    increment state.nonce with overflow rejection
    transfer the authorized amount and any authorized fee
    preserve the vault's persistent state
    commit all changes atomically, or revert
```

The EVM vault increments its nonce before the external value call. A reentrant call observes the next nonce and must satisfy a new request. On Solana, the instruction validates all relevant accounts and keeps the state reserve intact. The amount is range-checked for the target chain. Expiry uses an explicitly selected host-chain measure. All effects, including any vault-funded fee, follow the same atomic success rule.

Security conditions and properties

The relevant adversary observes public requests and proofs, controls relayers, submits competing transactions and makes quantum queries allowed by the authorization model. The wallet's secret and trusted request display remain secure. The host chain supplies authenticated execution and a canonical finalized history.

A1. Authority derivation. The Parano1d secret has sufficient entropy. The selected KDF separates vault domains and makes their derived authority secrets computationally indistinguishable from independent secrets under the declared quantum model.

A2. Authorization security. The pinned external-vault configuration satisfies fresh-request unforgeability and witness protection in the declared quantum adversary and oracle model, with explicit resource budgets.

A3. Request binding. Canonical encoding and the intent identifier bind every execution-relevant request field. The target verifier agrees with the reference protocol on parsing, arithmetic, transcript and acceptance.

A4. Deployment and execution. The wallet authenticates the network, account code, authority state and verifier parameters. All value-releasing paths enforce the specified policy. Host execution and finality preserve these rules.

A5. Persistent freshness. The nonce increases without wraparound. Its state survives balance depletion and every permitted account lifecycle transition.

> ### Proposition 1. Conditional withdrawal authorization
>
> Under A1–A4, every successful native-asset release through the specified account rules corresponds to an authorized canonical request, except with probability bounded by the derivation, authorization-forgery and request-binding advantages:

> $\Pr[\text{unauthorized release}] \leqslant \varepsilon_{KDF} + \varepsilon_{Auth} + \varepsilon_{Bind}$.
>
> The bound conditions on authenticated deployment, correct execution and a secure local wallet. The authorization term carries the primitive's full model and resource conditions.

Argument. Replace derived authority secrets with independent secrets, incurring the derivation advantage. The execution rule accepts a release only after checking the statement reconstructed from the actual request and stored commitment. An accepted request with a fresh identifier yields an authorization forgery. If its identifier was previously authorized but its action differs, the two canonical requests yield a binding failure. The unchanged authorized request executes its approved action. A union bound gives the stated inequality.

Replay exclusion

Under A5, two successful executions cannot consume the same nonce in one canonical state history. The first execution advances the counter before value release. Any later execution with the old counter fails. Atomic rollback couples counter changes to transfers, and reentrancy checks preserve the same invariant during a transaction. A chain reorganization is handled under the host finality condition.

Relayer independence

A relayer can submit an already authorized request. Changing its recipient, amount, vault, network or fee terms changes the statement and requires a new authorization. Delaying or withholding submission affects availability. Multiple relayers can carry the same request, while the nonce limits successful execution to one.

Security layers

The account's proof predicate supplies the proposed post-quantum authorization layer. Its concrete strength depends on authority-preimage hardness, proof security, request binding and the derivation configuration. The host chain supplies consensus, finality, account execution and transaction availability. On an L2, these conditions include the declared settlement and data availability architecture. A relayer's ordinary signing key controls its own fee account and submission capability. These conditions define the protection obtained for vault funds.

Fees, recovery and account lifetime

Native custody and funding

The selected EVM network's native coin is deposited directly into its Parano1d contract. SOL is deposited directly into the initialized Solana Parano1d PDA. The wallet displays separate balances by network and asset. Storage and withdrawals require host-chain fees and, on Solana, a persistent state reserve. They

require no liquidity pool. Exchange between assets is a separate protocol involving counterparties or liquidity providers.

Each host chain verifies the authorization locally. The wallet derives an external-vault authority from the Parano1d secret and proves the requested action with the pinned profile. Withdrawal verification is self-contained on the selected EVM network or Solana.

Recovery from the Parano1d secret

Restoration derives the same authority branches from the Parano1d secret. A public deployment manifest supplies the KDF definition, network identity, versions, factory or program identity, code and parameter digests, and index convention. The wallet recomputes addresses and reads their current nonces. An authenticated manifest and account discovery procedure make recovery reproducible.

The baseline keeps each vault's authority commitment fixed. Loss of the secret loses its authorization capability. A future recovery or authority-rotation policy must receive the same level of protection as withdrawal. The immutable baseline can migrate funds to a newly deployed account through an ordinary authorized transfer.

Persistent Solana replay state

Solana account lifetime creates a specific replay hazard. Suppose a vault executes a request at nonce zero, closes its state account, and later recreates the same PDA with nonce zero. The old proof could again authorize the old request after fresh deposits arrive. A deterministic address alone does not preserve the spending history.

The baseline therefore keeps the Parano1d PDA initialized and rent-funded. Withdrawal logic preserves its state reserve and rejects reset or closure paths. An alternative layout could place replay state in a separate permanent controller account. Supporting closure and recreation would require a persistent incarnation record, with that incarnation bound into every request.

The EVM contract likewise retains its code and nonce when its native-coin balance reaches zero. Its specified lifecycle preserves the authority predicate through every reachable operation. The wallet waits for the network's required deployment finality before directing funds to either account.

Executable verification and resource costs

The Base and Solana implementations release native funds after checking a complete authorization. Public Solana Devnet experiments use v1 transactions, 3,584-byte upload chunks, fourteen ordered verification windows and a 1,400,000-CU limit. Base Sepolia supplies a public EVM demonstration. A separate local Base configuration and mainnet fee-oracle quote measure execution and publication costs.

> **Complete withdrawals on Base and Solana**
>
> Signed local transactions completed native ETH custody on Base and native SOL custody on Solana. Each vault released 0.25 native coin after full verification. Independent relayers and interrupted-client recovery passed.

Full successful withdrawal jobs on 8 November 2026. Base is the measured EVM example.

Metric	Base (EVM)	Solana
Complete withdrawal	10 transactions	87 transactions and 14 ordered verification windows
Network fee estimate	$1.40, including L1 data and the configured operator charge	0.000470 SOL, about $0.04, with zero priority fee
Verification resources	103,890,467 gas in total. At most 11,820,999 gas per transaction	At most 1,140,062 CU per verification transaction
Temporary witness storage	Authenticated cache hash in vault storage	42,127 bytes. Mainnet reserve quote: 0.2146554 SOL
Wallet proof generation	53.03 ms	54.39 ms
Verifier runtime	19,169 bytes of EVM code	SBF program executed in LiteSVM

The Base estimate is **$1.3937 for execution, $0.0056 for Ethereum data and $0.0000 in operator charges**. The mainnet oracle quote fixes block 54,705,166 on 08 November 2026, 09:50 UTC. It includes a priority fee of 0.001 gwei. Base execution, L1 data and configured operator charges determine the network fee.[17] A relayer service price is a separate commercial choice.

At this quote, reducing verifier execution by one million gas saves approximately $0.0134. The job requires capacity in at least 3 blocks under the recorded execution and DA limits. Actual elapsed time also depends on sequencer ordering, load, submission and confirmation policy.

The Base run uses chain ID 8453 and a network-specific authority derived from the Parano1d secret. Execution gas comes from local signed transactions. The read-only quoting code evaluates every unsigned RLP transaction through `getL1Fee` and every operator charge through `getOperatorFee` at one pinned block. The oracle supplies signature padding. The estimate checks each transaction's L1 fee upper bound and DA footprint.

The Solana account retains a 256-byte header, a 9,792-byte verifier cache, the previous 7,212-byte verified query and a 24,867-byte reusable witness window. Its quoted mainnet reserve is 0.2146554 SOL. The read-only mainnet quote records the rent requirement and fees for measured transaction layouts. The account returns its full actual balance on closure. The recorded local job has 72 upload or

completion-probe transactions, 14 completed verification windows, account creation and closure. Fresh proof randomness changes delta sizes and transaction counts. Shared code deployment, vault creation and rejected tests are separate costs.

Solana v1 carries inline compute and account-data limits. The signed local v1 wires are checked directly. LiteSVM executes the same SBF instructions with matching limits expressed through its supported compute-budget path. The public experiments below execute v1 on Devnet.

Measured local cycle and relayer recovery

One CPU sample on the recorded Intel Core i7-1365U host. Runtime-specific local measurements.

Stage	Base JavaScript EVM harness	Solana SBF / LiteSVM harness
Native proof generation	53.03 ms	54.39 ms
Public witness expansion	99.59 ms	105.33 ms
All successful VM submissions	7.315 s	0.078 s
Local fixture processing	7.593 s	0.297 s

The latency record separates wallet proving, public transport preparation and local VM work. Fixture processing includes the prover's reference checks, witness expansion, signing and successful submissions. It excludes deployment and rejection probes. Separate read-only RPC observations record endpoint round trips and recent chain cadence. The public Devnet observations below measure delivery, sequencing, inclusion and observed finality for the tested client. Several dependent transactions may execute in one block when ordering and capacity permit.

The Solana transport accepts 3,584-byte upload chunks in any arrival order. A protected chunk bitmap supports concurrent delivery and idempotent retries. Each delta uses the previous accepted query, and a group commits only after every query passes. This gives 14 ordered verification windows. The recorded 87 transactions include 72 upload or completion-probe transactions, 14 completed verification windows, creation and closure. Relayer pipelining must preserve the required execution order and recover missing or rejected uploads. The public runs below record complete cycles with normal and interrupted delivery. Repeated-load median and tail measurements are the next evaluation target.

Both harnesses reconstruct the accepted progress after an interrupted client. Base recovers its cache from the public proof and authenticates it against the stored hash. Solana resumes a partial window from its persisted chunk bitmap and handles a lost verification response. Old-frame retries preserve the accepted prefix. The Solana receipts, native streaming checks and full result record provide the recorded cases and resource limits.

Public Solana Devnet withdrawals

Two complete public Devnet jobs released native test SOL after all 64 query checks. The program was deployed with verified ELF bytes and frozen upgrade authority. Each v1 job used fourteen verification windows and direct QUIC delivery to validator TPU ports. Every upload included a conditional completion check. The transaction that completed the bitmap also verified the entire window. Confirmed account notifications advanced the relayer to the next window. RPC supplied state reads and WebSocket notifications.

Observed public Devnet withdrawals on 8 November 2026. Each timer starts before fresh proof preparation and includes creation of the temporary proof account. Shared deployment and vault setup precede the timer.

Delivery case	Confirmed withdrawal	Finality observed	Unique transactions	Network fee
Normal v1 delivery	18.08 s	21.63 s	88	0.000445 test SOL
Dropped packet and reordered v1 delivery	22.34 s	26.04 s	91	0.000460 test SOL

The interrupted-delivery case shuffled the signed window batch and withheld one packet. The relayer recovered missing chunks from the protected bitmap. Repeated transmission reuses the same signed wire until its blockhash expires. This preserves the transaction identity through delivery retries.

The public Devnet record links the captured runs. Automated read-only receipt checks covered every captured signature, all fourteen finalized verification transactions, the actual payout balance changes, complete reserve refunds, request binding and immutable program bytes. The observed temporary reserves were 0.21075396 to 0.22599396 test SOL, returned in full.

These observations describe the tested client, Devnet scheduling and its confirmation policy. Mainnet latency under production load remains a deployment measurement.

Public Base Sepolia withdrawal

A complete public Base Sepolia job released native test ETH after all 64 query checks. The ten authorization transactions consumed 103,886,415 gas and paid 0.000623630745 test ETH in actual receipt fees. The vault consumed its nonce once. Its authority and verifier parameters were checked against the deployed code and public proof. The final withdrawal transaction records the native-asset release.

Public Base Sepolia, chain ID 84532, on 8 November 2026. Deployment and deposit precede the timer.

Observation	Result
L2 receipt confirmation	14.46 s
Authorization transactions	10

Observation	Result
Execution gas	103,886,415
Actual network fee	0.000623630745 test ETH

The timer includes fresh proof preparation, witness expansion, read-only rejection checks and ten sequential L2 receipt observations. The result record separately captures host-network safe and finalized observations. Testnet receipt fees and the pinned Base Mainnet dollar estimate have separate scopes.

A read-only cross-check through a second public RPC verified every signed transaction, canonical block membership, deployed code hashes, request fields, all query positions, payout balances, nonce consumption and exact fee reconciliation. The public result contains the complete transaction journal.

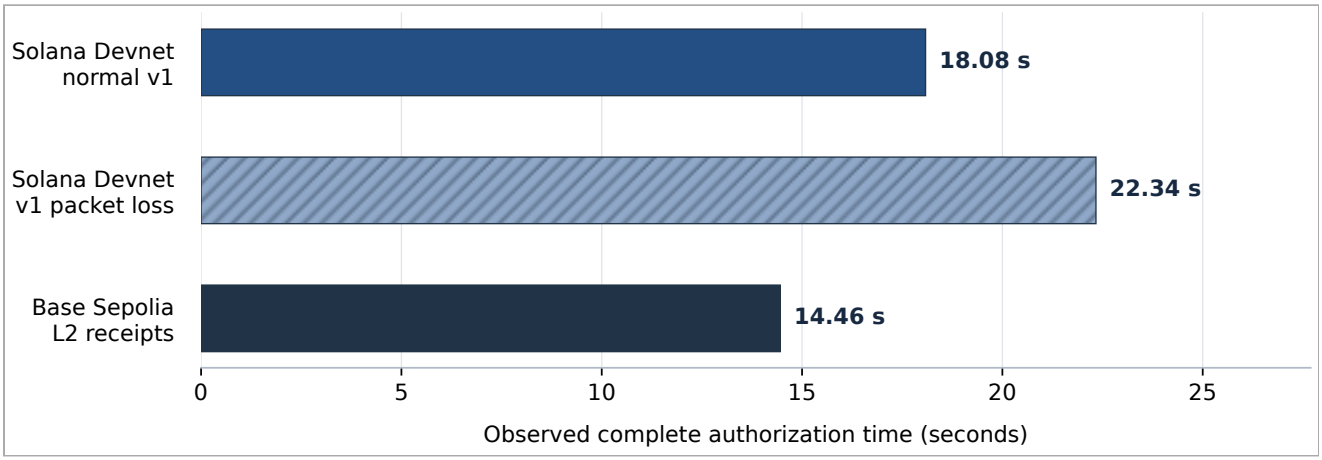


Figure 4. Public authorization runs with receipt cross-checks. Each timer begins before proof preparation. Solana includes temporary-account creation and confirmed account observations. Base includes read-only rejection checks and L2 receipt observations. Shared deployment and deposits precede both timers. Finality levels appear separately in the result records.
View full-size SVG · PNG

The external-vault authority profile

The field is KoalaBear, with modulus $2^{31} - 2^{24} + 1$. The extension has degree eight and defining polynomial $X^8 - 3$. The authority uses Poseidon2 of width 24 with eight full rounds and 23 partial rounds. Its trace has 256 rows and 40 main columns. The public statement contains twelve authority outputs and sixteen limbs of the request identifier.

The proof enables HidingFriPcs, hiding Merkle commitments and the uni-STARK randomization polynomial. Its parameters are 64 queries, blowup eight, folding arities 4, 4, 4, 4, 2, eight random codewords and ten base-field salt elements per committed row. A fresh 256-bit operating-system seed supplies each proof's randomness. SHA-256 and Keccak-256 outputs form a 64-byte transcript and Merkle digest. The source pins the backend, authority relation, preprocessing and canonical codec.

This profile derives an external authority from the Parano1d secret. Its quantitative security analysis concerns these exact prime-field parameters. The published binary-field authorization model supplies the interface and design context.[1]

Solana: a reusable verification window

The client first uploads the common public statement and proof commitments. The program checks their checksum, canonical field encodings, authority commitment and outer AIR equation. It derives a protected transcript cache. The reusable window then carries groups of up to five queries. Sorting the sampled query multiset preserves every sampled index and its multiplicity. A twelve-byte bitmap selects literal leaves, authentication nodes and inverse hints. Other components are copied from the previous fully verified query. Expansion recovers an ordinary query witness before verification.

Every query checks four input commitment paths, five FRI paths, the opening reductions, the complete fold chain and the final polynomial. A group replaces the retained witness and advances the protected counter only after every query passes. Retrying the current window preserves the cache and accepted prefix. Every verification transaction rechecks the vault, request, nonce, expiry and parameter identity. The final group increments the vault nonce and transfers SOL atomically. Closing the temporary account returns its full reserve. The persistent 80-byte vault retains its nonce and state reserve.

Base: authenticated cached EVM arithmetic

The factory derives the Parano1d contract address with CREATE2 from the authority, index and pinned configuration. The first authorization transaction checks the outer AIR equation. The second computes transcript challenges, powers and common opening reductions, then stores the resulting cache hash in the vault's verification job. Later transactions authenticate that cache and verify groups of query chains.

Each query carries conventional authentication paths to the same four input and five FRI roots checked by Rust. All 64 query chains must pass. The job fixes the request and base proof, and the vault accepts the next contiguous group only. The final group consumes the nonce and transfers the native coin. The runtime code fits EIP-170, and every measured transaction remains below the EIP-7825 gas cap.[15] The measured accounting includes the calldata floor of EIP-7623.[16]

The staged acceptance invariant

A verification job fixes the authority, request, parameters and proof commitments. Its progress changes only after the next required predicate accepts. Induction on the progress counter gives acceptance of every preceding predicate for that same job. The terminal transition therefore enforces the outer equation, authenticated openings and every query chain before release. Request binding prevents substitution, and the consumed vault nonce prevents a second release. Merkle binding supports the correspondence between shared native openings and independent EVM paths.

The complete configuration, predicate decomposition and reproduction commands are in the protocol record and experiment instructions.

Validation and security conditions

Local VM runs, publicly finalized Solana Devnet v1 withdrawals and a public Base Sepolia withdrawal establish the native-coin lifecycle under the recorded execution conditions. Validation covers altered amounts and recipients, replay, invalid openings, partial verification and independent fee-paying relayers. Base checks cover skipped query positions, modified caches and recovery of the accepted prefix. Solana checks cover incomplete windows, unauthorized writes, expired requests, altered nonces, modified FRI rows, Merkle paths, inverse hints and account-size changes between verification steps. Native streaming checks also reject malformed delta encodings. Relayer restart and lost-response cases preserve the verified prefix. The receipts record the successful and rejected transactions.

The production security work is a concrete quantum chosen-request and repeated-use privacy analysis for the specified backend, cryptanalysis of its authority configuration, and an independent audit of the verifier ports and account lifecycle. Production deployment validation covers mainnet scheduling, fees and finality under load. L2 fee calculations must include that network's settlement-data charge. The artifacts provide local Base execution, public Base Sepolia and Solana Devnet v1 withdrawals, automated receipt cross-checks and reproducible measurements.

Proof size and public compression

The normal v1 capsule carries 278,085 encoded bytes. The byte accounting estimates 224,312 bytes for a global Merkle frontier and 194,848 bytes when public preprocessing is embedded. Those representations require at least 55 and 48 full 4,096-byte payloads before framing. A thirty-transaction target calls for a smaller proof representation. The next research direction is transparent recursive compression of the public capsule. Its quantum knowledge extractor must recover an accepted inner capsule in the same adaptive authorization experiment, with a quantified extraction loss. The wallet retains the Parano1d secret and masking seed. The transport analysis records this target.

A wallet can present the entire authorization as one operation while a relayer submits its transactions. A sponsor can provide the temporary Solana reserve and recover it on closure. Native storage requires host-chain fees and state funding. Asset exchange additionally requires counterparties or liquidity.

References

1. Ignotus Nemo. Post-Quantum Signatureless Authorization. Zenodo, 29 October 2026. DOI: 10.5281/zenodo.23778742. Authority primitive, quantum security model and reproducibility artifact.
2. Hugo Krawczyk and Pasi Eronen. RFC 5869: HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC Editor, May 2010.
3. Vitalik Buterin. EIP-1014: Skinny CREATE2. Ethereum Improvement Proposals.
4. Solana documentation. Program Derived Addresses. Deterministic addresses, off-curve derivation and program authority.
5. Dankrad Feist, Dmitry Khovratovich and Marius van der Wijden. EIP-3607: Reject transactions from senders with deployed code. Ethereum Improvement Proposals.

6. Vitalik Buterin et al. EIP-7702: Set Code for EOAs. Ethereum Improvement Proposals.
7. Solana documentation. Account Structure. Lamports, program ownership and account state.
8. Solana documentation. Program Deployment. Program upgrades and immutable deployment.
9. Solana documentation. Transactions. Atomic execution and transaction size formats.
10. Solana documentation. Compute Budget. Per-transaction compute limits.
11. Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing 26(5), 1997.
12. Ethereum documentation. Quantum resistance. Account signatures, public-key exposure and quantum threats.
13. Optimism documentation. Differences from Ethereum. EVM equivalence, transaction fees and L1-to-L2 execution context.
14. Arbitrum documentation. Solidity support. EVM bytecode support and network-specific execution semantics.
15. Giulio Rebuffo and Toni Wahrstätter. EIP-7825: Transaction Gas Limit Cap. Ethereum Improvement Proposals.
16. Toni Wahrstätter and Vitalik Buterin. EIP-7623: Increase calldata cost. Ethereum Improvement Proposals.
17. Base documentation. Network Fees. L2 execution, L1 publication, operator fees and the minimum base fee.
18. Solana Foundation. Larger Transaction Sizes. The v1 format, 4,096-byte limit, inline resource configuration and cluster activation.

RESEARCH RECORD

Evidence and artifacts

The links below preserve the implementation, measurement or derivation behind this article.

PDF

Full research article (PDF) ↗

Code

Parano1dX source code and reproduction ↗

Architecture

Protocol and exact security conditions ↗

Artifact

Reproduce the Base and Solana experiments ↗

Data

Public measurements and automated receipt cross-checks ↗

Data

Base Sepolia transaction journal ↗

Data

Solana Devnet v1 withdrawals and captured results ↗

Paper

Signatureless authorization model and interface ↗

© 2026 Parano1d Lab