CRYPTOGRAPHIC ANALYSIS Signatureless authorization v2

What Does Your Money's Security Actually Depend On?

AI is accelerating mathematical research. Parano1d v2 connects a wallet secret to an exact request through hash-based authorization and recursive verification. Follow the mathematics that grants control.

Ignotus Nemo 27 Nov 2026 9 min read

ABSTRACT

Parano1d v2 authorizes requests through freshly randomized proofs of knowledge of an authority secret. A public hash commitment identifies the authority, the transcript binds the exact request, and recursive verification connects authorization to execution. This analysis maps the construction to its concrete implementation and examines the November 25 research on integer factorization and finite-field discrete logarithms. It traces the fixed-capacity proof-node schedule, distinguishes field arithmetic from secret-exponent assumptions, explains the native quantum chosen-request security reduction and identifies the AES-128 resource comparison used for concrete security assessment. Public source links and executable checks accompany the analysis.

The mathematics that grants control

A breakthrough in integer factorization changes the security of RSA. Parano1d v2 places ownership in a different relation: knowledge of a secret for a public hash commitment, proved for an exact request. The wallet produces the proof. The network checks it. Recursive verification carries the accepted transition into State.

That dependency matters when mathematical progress becomes a security headline. In late November, discussion of AI-assisted cryptography accelerated around reports of advances inside research laboratories. Scott Aaronson clarified his account: the factoring claims circulating publicly were rumors. Alongside that discussion, published work gives concrete algorithms and complexity bounds that can be examined directly.

The useful question for a payment system is precise: which problem must an adversary solve to authorize a payment belonging to someone else? This article follows that question through Post-Quantum Signatureless Authorization, v2 and its fixed-capacity implementation.

Three new algorithms, three exact targets

The November 25 Cogentic release in the Google Research repository includes three particularly relevant number-theory papers. Their input problems identify where they enter a cryptographic assessment.

Published number-theory results and the mathematical objects they operate on.

Work	Target and stated result	Cryptographic connection
Unconditional Integer Factorization	Complete factorization of an integer *N*. The paper gives a zero-error randomized algorithm with asymptotic expected time $L_N[1/3, 105]$.	RSA depends on the difficulty of recovering the prime factors of a public modulus. This result addresses that input problem through a rigorous subexponential bound.
Deterministic Subexponential Integer Factoring	Integer factorization with a stated deterministic bit-operation bound $O(\exp(3(\ln N)^{3/4}))$. The work also studies quadratic nonresidue construction.	It changes the provable algorithmic bounds for factoring and related number-theory tasks. Concrete key-size assessment also uses finite costs and thresholds.
Discrete Logarithms in Finite Fields of Fixed Characteristic	Given nonzero field elements *g* and *h*, recover an exponent *x* satisfying $g^x = h$. The stated asymptotic complexity is below quasipolynomial for fixed characteristic.	The target is the multiplicative group of a finite field. Its relevance to a protocol follows from that protocol's use of a secret exponent in this group.

For the first result, the notation is $L_N[\alpha, c] = \exp(c(\ln N)^{\alpha}(\ln \ln N)^{1-\alpha})$. The exponent, constants and computational model all matter. The papers provide specific objects to review, rather than a single security rating for every system that uses arithmetic.

From a secret to a permitted request

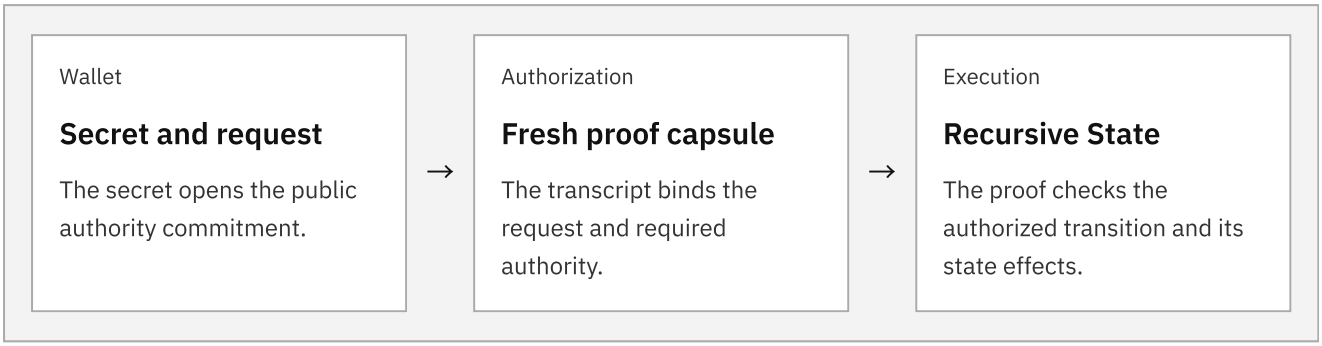
A Parano1d wallet holds a 256-bit spend secret. The address function interprets it as two field elements, combines them with a fixed address-domain capacity and evaluates the width-four Poseidon2b permutation. Two output lanes form the 32-byte public authority commitment.

$$a = \mathrm{Addr}(s) = \mathrm{rate}(P_0(s_0, s_1, c_{\mathrm{addr},0}, c_{\mathrm{addr},1}))\,.$$

The public authorization statement pairs this commitment with the canonical request identifier:

$$m = \mathrm{ID}(I), \qquad x = (m, a).$$

The capsule proves knowledge of a secret satisfying the address equation for that statement. The verifier reconstructs its challenges from the request, authority, commitments and ordered prover messages. The application checks the request's state references, policy and freshness.



Control follows the secret-to-address relation and the exact authorized request. Public verification connects that authorization to execution.

In the wallet implementation, the prover evaluates the address permutation and checks its output against the required authority before constructing the capsule. Fresh operating-system randomness supplies the source-bank masks, terminal pads and companion data for each authorization attempt.

Binary fields are the arithmetic

The construction represents its authority trace over GF(2^{128}) and uses GF(2^{256}) for extension-field challenges. Addition, multiplication and squaring make the computation efficient to express and verify inside the proof system.

A finite-field discrete-logarithm instance has the form $h = g^x$, where x is the secret exponent to recover. The authority relation has the form $a = \mathrm{Addr}(s)$, where s is the input to a fixed permutation circuit. Its nonlinear operation $x \mapsto x^7$ uses a public, fixed exponent. The prover establishes the full sequence of round equations, mixing layers and output constraints.

This identifies the relevant cryptanalytic target: invert the selected authority map, recover information about its secret through repeated proofs, force transcript challenges, or equivocate on a commitment used by an accepted proof. A new algorithm becomes relevant by giving an advantage on one of those tasks.

The fixed-characteristic logarithm paper is therefore useful to examine at the level of its reductions and algebraic techniques. Applying its stated solver directly requires the multiplicative-group input relation it solves. The v2 ownership equation specifies the separate inversion problem that an authorization attack must address.

The commitment path is part of the protocol

The authorization bank contains 2,048 cells. A degree-ten reduction connects the address trace to committed source values. A companion argument and 65 code queries authenticate the resulting evaluation claims.

In v2, a proof-tree node hashes its two 32-byte child digests through a fixed-capacity sponge. The left digest enters the rate lanes first. After the first permutation, the right digest enters the rate lanes of the resulting state. In flat-basis coordinates:

$$
\begin{aligned}
S_0 &= (0, 0, c_{\mathrm{node},0}, c_{\mathrm{node},1}),\\
S_1 &= P(S_0 \oplus (L_0, L_1, 0, 0)),\\
S_2 &= P(S_1 \oplus (R_0, R_1, 0, 0)),\\
\mathrm{Node}(L, R) &= \mathrm{rate}(S_2).
\end{aligned}
$$

The capacity is initialized by the node domain. The caller supplies child digests through the rate. The public proof profile selects this construction, and the native verifier and recursive circuit use the same two-permutation schedule.

Leaves have their own typed encodings and domain tags. The complete verifier recomputes Fiat–Shamir challenges, checks the Owner reduction, verifies the linked evaluation argument, derives query positions and authenticates the selected source and middle openings. Together, these checks bind the values used by the proof to its public statement.

The v2 authorization path in the public implementation.

Property	Code to inspect
Secret-to-address relation	`derive_address`
Capsule construction and fresh entropy	`prove_zk_authorization_from_state_table_for_version`
Fixed-capacity proof nodes	`ProofNodeHashVersion::FixedCapacity`
Complete native verification	`verify_zk_authorization_for_version`
Version-bound node selection	`OwnerAuthorizationVersion::V5`
Recursive transcript, leaf and path constraints	`zk_authorization_region` and `proof_node_path`

The implementation identifies the v2 fixed-capacity capsule as `OwnerAuthorizationVersion::V5`. This code-level selector fixes both the transcript construction identity and proof-node hashing mode.

Quantum security has a defined adversary

The v2 paper studies an adversary with quantum workspace and coherent forward and inverse permutation access. Authorization requests and the final forged request are classical. The adversary can obtain authorizations on adaptively chosen requests and then attempt a fresh one under the same authority.

The theorem reduces fresh-request forgery to authority-preimage recovery and canonical intent binding, with explicit simulation and extraction losses:

$$\Pr[\mathsf{Forge}_{\mathrm{Q}}] \le \mathrm{Adv}^{\mathrm{Q}}_{\mathrm{pre}} + \varepsilon^{\mathrm{Q}}_{\mathrm{sim}} + \varepsilon^{\mathrm{Q}}_{\mathrm{ext}} + \varepsilon^{\mathrm{Q}}_{\mathrm{id}}.$$

Proof hashing uses an address-compatible random-completion model. The public authority circuit fixes the permutation on its address input slice. A uniform completion supplies the shared proof-hashing and transcript permutation elsewhere. Distinct capacity roots, the prescribed absorb schedules, fresh prover coins and explicit resource envelopes define the construction analyzed by the theorem.

For at most 2^{32} charged native calls and no more authorization attempts than calls, the combined simulation and extraction loss is below 2^{-51}. This term measures the reduction's additive error at that budget. Authority inversion and intent binding have their own advantages in the displayed inequality.

Concrete strength uses an attack-resource comparison. NIST's Category 1 reference is AES-128 key search. A comparison records attack success, logical gate count, circuit depth, memory and preprocessing at matching resource limits. Published AES quantum-circuit estimates make those units explicit. Query counts and additive probability terms answer different questions within that assessment.

The concrete Poseidon2b assessment follows its round constants, mixing matrices, subfield structure, fixed capacities and complete accepted proof paths. Continuing research addresses the attack costs for this selected permutation and coupled all-root recursive composition.

Authorization that the blockchain itself proves

Parano1d checks authorization in the same binary-field arithmetic used to prove execution. The recursive relation authenticates the selected transcript, leaf hashing, Merkle paths and authority equations. It couples the permitted request to the state transition checked by the block proof.

This gives the construction a practical architectural purpose. A wallet proves authority once for an exact intent. An execution prover carries the capsule into a valid state transition. A verifying node checks the resulting recursive proof and the public State it authenticates.

Local checks reproduce the fixed-capacity hash across scalar and batched implementations and across flat and tower field bases. The authorization regression verifies the selected capsule, enforces its version identity and rejects an altered Merkle opening. The source links above expose each boundary directly.

```
cargo test --release --locked -p noid_poseidon2b native::proof_node::tests::
cargo test --release --locked -p noid_gkr
fixed_capacity_profile_keeps_owner_relation_and_rejects_other_versions
```

New research can therefore be assessed against a defined path: the secret-to-address map, the fresh request-bound proof, its transcript and commitments, and the recursive acceptance relation. Cryptographic independence begins with that explicit account of what grants control over the money.

RESEARCH RECORD

Evidence and artifacts

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

Paper
Post-Quantum Signatureless Authorization v2: paper and reproducibility archive ↗

Code
Complete version-selected authorization prover and verifier ↗

Code
Fixed-capacity proof nodes and scalar/batch correspondence tests ↗

Paper
Unconditional Integer Factorization in Time L_N[1/3, O(1)] ↗

Paper
Deterministic Subexponential Integer Factoring and Quadratic Nonresidue Construction ↗

Paper
Discrete Logarithms in Finite Fields of Fixed Characteristic ↗

Reference
NIST security evaluation criteria and AES reference resources ↗

Paper
Implementing Grover oracles for quantum key search on AES and LowMC ↗

Reference
Scott Aaronson's clarification of the factoring rumors ↗

© 2026 Parano1d Lab