

PoUT NETWORK

TECHNICAL ARCHITECTURE, REAL-WORLD UTILITY, AND CONDITIONAL TOKEN MIGRATION

Technical overview / ecosystem design memorandum / public discussion draft

Revision: 0.5 | Reference date: 20 October 2026 | Status: Technical research and conditional migration planning

Purpose. A technical, evidence-constrained explanation of the PoUT project, its experimental Rust components, candidate utility model, existing Solana community token, and a planned, technically conditional migration pathway to native-chain integration. This memorandum is not an offer to purchase tokens, an investment prospectus, a security audit, a token redemption agreement, or a guarantee of a mainnet launch.

Disclosure boundary. Public-source modules are identified explicitly. Internal research results are summarized without releasing unpublished protocol internals. Pseudocode below describes possible interface contracts and security requirements; it is NOT deployed protocol code.

0. Executive technical position

PoUT (Proof of Useful Traffic) is an experimental architecture pursuing a narrow question: can independently observable, useful network service be converted into an auditable resource signal while resisting replay, fabricated demand, multiple identities, outsourcing and coalition-wide cost sharing? The current answer is NOT YET ESTABLISHED. The project contains implementation evidence for several cryptographic and execution subproblems; those results do not establish a secure permissionless consensus mechanism.

The existing, tradable POUT community token on Solana is intended to retain a material role within the planned PoUT ecosystem. An eventual transition/integration with a future native network is a planned engineering objective. Technical migration models and synthetic accounting tests already exist; no actual source-chain bridge, conversion, native allocation, ratio, migration date or enforceable token-holder entitlement has been selected or implemented. Possible interim source-chain service integrations remain subject to separate technical, security and legal approval.

1. Research target and measurable service predicate

A cryptographic proof of a computed relation is not synonymous with proof of a useful real-world event. A verifier may confirm knowledge of source bytes or consistency of a transcript without learning whether a requested file was delivered to an independent recipient, held for an agreed duration, or used for economically independent demand.

Let $e = (\text{provider}, \text{requester}, \text{object}, \text{challenge}, \text{interval}, \text{transcript})$.

$V_{\text{crypto}}(e) \in \{0,1\}$ -- cryptographic statement validity

$A_{\text{policy}}(e,H) \in \{0,1\}$ -- authorized and unconsumed in history H

$U_{\text{service}}(e,O) \in \{0,1\}$ -- independently observed useful service O

$W(e,H,O) \geq 0$ -- hypothetical consensus weight

Required separation:

$V_{\text{crypto}}(e)=1$ does NOT imply $U_{\text{service}}(e,O)=1$.

$A_{\text{policy}}(e,H)=1$ does NOT imply independent economic demand.

Distinct provider IDs do NOT imply distinct physical resources.

A practical service candidate might be bounded delivery of specified bytes to an independently authorized requester, within a fixed interval, with independently sampled transfer evidence. This is a research direction, not a deployed consensus eligibility rule. A future design must decide what counts as “useful,” who supplies demand, who verifies delivery, and whether an attacker can cheaply act as provider, customer and verifier simultaneously.

1.1 Coalition cost, not per-identity cost

$C_{\text{coalition}} = (C_{\text{prep}}, C_{\text{online}}, C_{\text{byte_time}}, C_{\text{transfer}}, C_{\text{external}})$

C_{prep} : shared preprocessing / source acquisition

C_{online} : challenge-dependent computation

$C_{\text{byte_time}}$: stored bytes * retained time (e.g. byte-seconds)

C_{transfer} : actual transmitted bytes / measured link use

C_{external} : payments to independent parties (net external cost)

If coalition Z controls n nominal provider identities:

$C(Z) \neq \sum_i C(\text{identity}_i)$ if work/source/storage is shared.

The same prepared data must be charged ONCE, not n times.

Candidate soundness objective (NOT PROVED):

$E[C_{\text{coalition}} \mid N_{\text{accepted}}=k] \geq g(k, \text{horizon}, \text{adversary assumptions})$

with g growing meaningfully for additional genuinely useful work.

Any ratio such as expected total coalition cost / expected accepted events needs a fixed horizon, cost units, a non-zero denominator and a canonical selected history. Ratios are undefined when the experiment accepts zero events. Hash counts and proof counts cannot automatically be converted into storage or bandwidth lower bounds.

1.2 Falsifiable service-to-consensus hypothesis (research model)

The proposed eligibility gate must treat validity of a cryptographic relation, authorization of a request, independent evidence of useful service, and canonical consumption as four non-interchangeable predicates. The expression below is a test specification, not an implemented PoUT consensus rule.

$e := (\text{request}, \text{provider}, \text{object}, \text{challenge}, \text{interval}, \text{response}, \text{observation})$

$E(e, H) := V_{\text{relation}}(e) * A_{\text{request}}(e) * O_{\text{useful}}(e) * (1 - R_{\text{replay}}(e, H))$

$w(e, H) := E(e, H) * f(\text{evidence_size}, \text{verified_service}, \text{policy})$

$W_Z(H) := \sum_{\{e \text{ in canonical_history}(H), \text{attributed_to}(Z)\}} w(e, H)$

$E(e, H), A_{\text{request}}(e), O_{\text{useful}}(e), R_{\text{replay}}(e, H) \text{ in } \{0, 1\}$

A nonzero O_{useful} requires a separately justified observation channel; a signed receipt produced by a cooperating customer and provider is not inherently independent. A copied witness may satisfy the same R1CS even when a second admission is denied by selected-history accounting.

$C_Z(T) := (\text{prep_ops}, \text{online_ops}, \text{storage_byte_seconds}, \text{transmitted_bytes}, \text{external_spend})$

Optional scalar $C^*_Z(T) := \alpha * \text{online_ops} + \beta * \text{byte_seconds}$

$+ \gamma * \text{transmitted_bytes} + \delta * \text{external_spend}$

$\Pr[W_Z(H_T) \geq k \text{ and } C^*_Z(T) < G(k)] \leq \epsilon(k, T) \text{ // target, UNPROVEN}$

Any scalarization coefficients $\alpha/\beta/\gamma/\delta$ must be independently specified, consistently dimensioned and tied to a fixed adversarial horizon; otherwise $G(k)$ has no defensible quantitative meaning. A coalition may reuse one source, outsource computation, control multiple provider/requester accounts and retain off-canonical work. J2 exhibits a finite valid shared-source case.

2. Existing architecture and status of proof

Subsystem	Present implementation or research evidence	Unresolved boundary
Rust execution / CWC	Bounded serial reference and genuinely parallel candidate-state execution; deterministic outcomes checked under artificial operations.	Not autonomous BlockDAG ordering or permissionless consensus.
Public Groth16 verification	BLS12-381 pairing checks against retained synthetic public proof vectors, external pin checks and malformed-input tests.	Public package excludes full prover, circuit/setup and live service.
Research file relation	Finite SHA-256 file-source, challenge, encoding and response relation in private development.	Knowledge/consistency is not delivery, retention or demand.
Native ratchet / R1CS	Constraint synthesis, actual assignment checks and an isolated joint file-ratchet model.	No new joint Groth16 proof, no physical resource certification.
DAG / CWC integration	Finite isolated DAG/CWC research models.	Operational security/liveness and weight admission unproven.
Live networking	Experimental encrypted transfer/recovery components under development.	Round7 reliability acceptance open; central discovery/settlement.
Token and migration	Existing owner-reported Solana community token; synthetic Rust migration accounting model.	Solana mint confirmed by the project owner; on-chain program, decimals, authorities and eligible supply require verification. No approved bridge, conversion ratio or native allocation.

Evidence scope: the public v0.1.0 repository reports 41 passing default workspace tests for its original release, plus an October 2026 accounting-controls addendum whose newly added tests were not independently rerun for the published package. Internal engineering records as of the examined 20 October J1/J2 checkpoint report 870 locally passing Rust tests (0 failures; seven helper fixtures marked ignored in parent tests). The counts refer to DIFFERENT source trees and must not be combined. Hosted CI and independent external audit are not claimed. [P1], [P2], [I1]

3. Joint relation: illustrative cryptographic structure

The privately tested joint research model binds a finite file opening, audit context and a ratchet transition. It has a 546-byte public frame, 18 explicit packed field inputs and 800,242 measured R1CS constraints. These facts are local research observations, not verified public prover evidence or a production specification. Cryptographic internal design remains subject to security and intellectual-property review before source publication.

Representative commitment pattern (SCHEMATIC; not a protocol export):

$F = H(\text{domain_file} \parallel \text{object} \parallel \text{context} \parallel \text{challenge} \parallel \text{response})$

$R = H(\text{domain_state} \parallel \text{previous_state} \parallel \text{scope} \parallel \text{challenge})$

$L = H(\text{domain_joint} \parallel \text{canonical_file_statement} \parallel \text{audit_context})$

$\text{JointRelation}(\text{source_witness}, \text{old_state}, \text{public_inputs}) :=$

$\text{FileRelation}(\text{source_witness}, \text{file_statement})$

$\& \text{RatchetTransition}(\text{old_state}, \text{ratchet_statement})$

$\& (\text{ratchet_statement.payload_commitment} == L)$

```
& (derived_challenge == committed_challenge)
& CanonicalPublicInputBinding(public_inputs)
```

R1CS form: for all rows j ,

```
<a_j, z> * <b_j, z> = <c_j, z> over field Fr.
```

In the finite J1 corpus, a correctly formed file-A component and separately correct ratchet-B component were intentionally combined. The mismatched joint assignment violated the payload-binding equations. J2 produced a different, equally important positive counterexample: the SAME source bytes and old state can form valid distinct statements under different permitted synthetic provider labels. Hence object-binding controls prevent the tested cross-object substitution, but cannot alone certify independent resources or unique real participants. [I1]

3.1 Statement satisfaction, binding and protocol acceptance

```
R1CS_SAT(x,w) := AND_j [ <a_j,z> * <b_j,z> = <c_j,z> ] over Fr
```

```
JointAccepted(x,pi,H,O) := VerifyProof(vk,x,pi)
```

```
    AND AuthenticatedExpectedInput(x)
```

```
    AND ExternalServiceObservation(O)
```

```
    AND Unconsumed(x,H)
```

Only the equation-satisfaction layer of the new joint J1/J2 research circuit has been exercised with synthetic assignments; no new proof π was generated or verified for that joint circuit. The composite acceptance expression is aspirational and cannot yet be evaluated end-to-end. Evidence that two separately valid components fail an incompatible joint assignment does not imply a general soundness theorem, proof of storage, or independently caused customer demand.

4. Desired real-world use cases (not existing commercial services)

4.1 Verifiable data delivery

A client requests object o , byte range r and time window $[t_0, t_1]$. An authenticated service endpoint delivers bytes and emits an independently checkable response. Exact bytes, challenge scope, requester authorization and replay controls matter. Cryptographic transcripts alone are insufficient: the claimant and observer may collude. Future tests must include two independent copies, one shared store behind many identities, and a responder without local data using delegation.

4.2 Resource-aware storage / replication

An optional storage service could expose capacity, availability and challenge responses over time. Proof of knowing a chunk at an instant does not prove separate physical replicas, duration, demand or complete file availability. Any storage reward rule must specify bytes, time, sample coverage, concurrency and cross-provider sharing costs.

4.3 Scheduling and execution of dependent workloads

CWC research explores conflict-aware parallel execution of bounded artificial operations. A future network could use similar scheduling to process independent activities concurrently; performance or correctness in the current test model is not evidence of production transaction throughput or full GHOSTDAG consensus safety.

5. Intended ecosystem role for the existing POUT token

Design requirement: preserve a meaningful ecosystem role for the existing Solana POUT token. The project intends to investigate service-related source-chain utility and a reviewed pathway for eventual integration/migration if a viable native network is developed. The migration question is a planned

workstream with documented technical models, not merely an unspecified possibility; its final mechanism, allocation, interoperability requirements and execution conditions remain unresolved. No current ownership, revenue, consensus or native redemption right follows from this planning objective.

5.1 Candidate utility layers

Candidate layer	Technical purpose	Status / unresolved decision
Service payment / credit accounting	Allow customers to purchase approved storage/transfer/execution units, with transparent unit pricing and receipts. Potential use of existing POUT as a source-chain payment/settlement asset via a controlled gateway.	Not deployed; requires demand, custody, pricing, payment finality and refund rules.
Native conversion pathway	Subject to approved and verified migration, offer a one-time conversion or other explicitly described treatment of eligible existing token units.	Planned technical path with Rust synthetic claim accounting; no ratio, native allocation, lock/burn procedure or date approved.
Governance participation	Potentially use token-based signaling on non-security-critical protocol parameters or treasury proposals.	Not implemented; token voting is Sybil- and wealth-sensitive and is not consensus security.
Ecosystem access / accounting	Possible fee settlement, usage credits or approved technical program access.	Must specify actual delivery and avoid artificial demand incentives.
Consensus role	Token-based bonds or penalties could be considered only as a separately reviewed hybrid security component.	No native staking, emissions, slashing or consensus weight currently authorized.

The token is not currently required to verify cryptographic statements or run the published prototype. No holding period, guaranteed discount, automatic revenue share, yield, buyback, mandatory burn, future allocation or 1:1 conversion is promised. Future service access should retain a safe non-financial participation route for research and community testing.

6. Planned Solana-to-native migration: technical alternatives and invariants

Migration planning is an active, conditional engineering workstream, not an executed bridge. A dedicated synthetic Rust accounting model and documented alternatives (burn, lock and finalized snapshot with additional circulation controls) already exist. The project intends to preserve continuity for the presently traded Solana POUT token if a future native chain reaches the required security and operational gates. The selection of a concrete migration procedure remains open. No holder action, conversion amount, claim ratio, snapshot slot, source burn/lock, native entitlement or execution date has been approved.

6.0 Source token identification: project-owner-confirmed mint

The project owner has confirmed the following Solana mint address as the existing tradable POUT community token. This establishes its project-side identification in this document, but does not independently verify on-chain mint parameters, token-program identity, circulating supply, authorities or extensions. Those facts must be checked before any migration can be executed.

Project-owner-confirmed Solana mint (on-chain audit pending): FQ47SAH4xuRJAfffAbqZSX1kq3ewE3GxnLuGYypump

CoinGecko separately indexes a PoUT-Network / POUT Solana asset (abbreviated contract FQ47S...Yypump) and reports a 1B maximum supply. These third-party, time-dependent figures are not an independent audit of the mint account, token program, token distribution or quantity eligible for migration. [P3], [P4]

Before any actionable migration terms, independently inspect the confirmed mint on Solana: chain/genesis, token program, decimals, supply, mint/freeze authorities, extensions, token accounts and finalized transaction history. Reconcile these facts with a signed, versioned source-asset manifest. Owner confirmation identifies the intended token; it does not establish eligibility quantities or approve claims.

6.1 Conversion models being evaluated

Method	Possible advantage	Principal failure mode
Verified burn -> claim	A correctly finalized burn may retire eligible source units as corresponding target claims are issued.	Irreversible source loss if target chain, bridge or claims fail; verifying burn/finality is nontrivial.
Audited lock -> claim	Source backing can remain explicitly locked while corresponding target claims circulate.	Custody, upgrade, emergency-unlock, solvency and two-way redemption risks.
Finalized snapshot eligibility	May simplify genesis entitlement preparation with a fixed slot/root.	Source token remains tradable; snapshot alone can create two simultaneously circulating economic representations.

No method is selected. A signed multisig attestation is explicitly trusted, not a trustless Solana light client. No one should transfer, burn or lock tokens based on this design memorandum.

6.2 Proposed on-chain claim semantics — illustrative, unimplemented

```

struct MigrationClaimV1 {
  source_chain_id: Bytes32,
  source_mint: Pubkey,
  source_token_program: Pubkey,
  migration_event_id: Bytes32,
  finalized_source_locator: Bytes,
  eligible_allocation_id: Bytes32,
  source_amount_base_units: u128,
  target_chain_id: Bytes32,
  target_genesis_hash: Bytes32,
  target_recipient: PublicKey,
  target_amount_base_units: u128,
  proof_version: u16,
  consent_expiry: u64,
  owner_authorization: Signature,
  source_finality_evidence: BoundedProof
}

nullifier := H(D_claim || migration_event_id || eligible_allocation_id)

accept_claim(c, state):
  require verify_source_identity_and_finality(c)
  require verify_owner_consent(c) && verify_target_recipient(c)
  require c.proof_version in approved_versions
  require current_time < c.consent_expiry
  require !state.consumed[c.nullifier]
  require checked_remaining_reserve >= c.target_amount_base_units
  atomically:

```

```

state.consumed[c.nullifier] = true
state.reserve -= c.target_amount_base_units
state.balance[c.target_recipient] += c.target_amount_base_units
return immutable_receipt(c.nullifier)

```

NOTE: illustrative pseudocode, NOT an implemented claim contract.

The nullifier must bind a stable source allocation and migration event, rather than a user-chosen nonce, signature or destination. Retrying with a new signature must not duplicate claims. Concurrent claims, interrupted execution, target-chain reorgs and acknowledgment loss must preserve exactly-once accounting. A request for a new destination after an earlier allocation is consumed must not silently pay twice.

6.3 Supply and double-circulation constraints

Assume an approved rational conversion p/q , integer base units.

$\text{TargetAmount}_i = \text{floor}(\text{SourceAmount}_i * p / q)$

Unapproved today: p , q , decimals, eligibility, deadline and reserve.

$\sum_i \text{TargetAmount}_i \leq \text{MigrationReserveApproved}$

$\text{Claimed} + \text{UnclaimedReserve} = \text{MigrationReserveApproved}$

$\text{MintedNative} = \text{GenesisAllocations} + \text{ApprovedEmissions}$

$\text{ServicePayment}(a,b,x)$: $\text{balance}[a] -= x$; $\text{balance}[b] += x$; minted unchanged

Source-side condition for true 1-way conversion:

eligible source units demonstrably retired or immobilized
before equivalent native claim becomes freely spendable.

Snapshot alone + freely trading source token

!= safe one-for-one conversion.

The rounding policy must specify dust handling and never round every claim upward silently. A full supply audit must distinguish circulating source units, burned source units, locked source backing, unclaimed target reserve, allocated target units, treasury supply and future emissions. Native claims against tokens held by exchanges, liquidity pools, multisigs or custodial wallets require additional beneficial-owner rules; an exchange wallet is not a list of its customers.

6.4 Preconditions for a migration decision

Before selecting a method: (i) verify the real mint/program/supply and ownership facts; (ii) complete native-chain genesis and consensus-security prerequisites; (iii) choose an independently reviewable source-finality mechanism; (iv) audit claim replay, binding and supply invariants; (v) test competing claims, crash recovery and reorgs on worthless environments; (vi) review wallets/custodians and accessible claim tooling; (vii) approve and publish legal terms; (viii) independently review bridge/claim security; and (ix) notify holders well before any irreversible step.

6.5 Staged migration engineering route - planned, not selected

The following sequence describes how the existing Solana asset could be integrated with a later PoUT native network. It is a concrete engineering roadmap, not a token redemption instruction. Migration remains a stated development objective. The implementation mechanism is not fixed: if a candidate fails its security requirements, it must be revised or replaced and cannot be activated until the necessary gates pass.

M0 / Identify: freeze a manifest = (source genesis, token program, mint, decimals, finalized supply, authority state, exact audit slot, target genesis/version). Cross-check independent source observations and project ownership. Nothing economic can be inferred from a symbol match.

M1 / Define: select ONE model after security review: (a) finalized burn-to-claim; (b) audited lock-to-claim with verified backing and exit rules; or (c) reviewed eligibility snapshot PLUS an explicit mechanism addressing source-token circulation. Snapshot-only issuance of an interchangeable native asset is not a completed migration design.

M2 / Bind: specify a versioned canonical claim payload binding source event/allocation, finalized proof, source amount, target genesis, native recipient, conversion rule, consent expiry and proof version. Define verified ownership signatures and independent claim evidence. Exchange/LP custody and partial claims require separate policies.

M3 / Execute on worthless test networks: perform duplicate-claim, concurrent-claim, interrupted-settlement, reorganization, wrong-mint, wrong-recipient, invalid finality and source/target double-circulation controls. Establish exactly-once durable consumption and a bounded, independently reproducible claim ledger.

M4 / Review and activate only after native security, operating readiness, wallet support, claim-accessibility, legal review, external audit and holder notice. Until all gates pass, there is no burn/lock request and the existing source token continues to exist independently on Solana.

A basic potential claim relation, conditional on an approved conversion policy, is:

eligible_source_amount -> verified_source_event -> stable_allocation_id -> native_claim

nullifier = H(D_migration || event_id || stable_allocation_id)

native_amount = floor(source_amount * p / q) // approved integer policy only

Invariant: SUM(claimed_native) <= approved_fixed_reserve

Invariant: completed_claim(allocation_id) is atomic, durable, and at-most-once

Invariant: source circulation and target circulation cannot both represent the same unrestricted transferable entitlement under a purported 1:1 replacement model.

The actual choice between a Solana-facing service utility path, one-way native conversion, or an approved hybrid must be evaluated against wallet usability, custody/finality risk and protocol economics. None is implemented for real assets.

6.6 Migration ledger state machine and reconciliation (proposed)

The following model specifies testable properties for a future claim processor without selecting a burn, lock or snapshot mechanism. All quantities are integer base units in explicitly verified denominations; source-finality proofs, native consensus, recipient authorization, and the migration policy remain prerequisites.

allocation_key := H(D_alloc || source_genesis || mint || migration_event || source_allocation)

nullifier := H(D_null || migration_event || allocation_key)

credit_i := floor(eligible_units_i * p / q) // approved p,q only

0 <= SUM_i credit_i <= R_migration // fixed approved native reserve

R_remaining := R_migration - SUM_{successful claims i} credit_i

forall allocation_key: consumed[allocation_key] in {0,1}

claim_commit := ATOMIC(consumed[key]:=1, R_remaining-=credit, balance[dst]+=credit)

The consumption key is stable across changed signatures, destination addresses, claim nonces and policy revisions: changing a user-selected nonce must not make the same source allocation claimable again.

Interrupted acknowledgments return the original result; rejected or rolled-back target blocks must roll back nullifier and balance together. Partial claims, if ever supported, require a checked remaining-quantity ledger rather than a reusable whole-allocation flag.

Burn model: Native_spendable <= floor(FinalizedBurnedSource * p / q)

Lock model: `Native_backed <= floor(FinalizedLockedSource * p / q)`
 Lock exit: `finalized_native_burn -> verified_source_unlock` // not vice versa
 Snapshot: `historical_balance_proof != retired_source_circulation`

These inequalities describe conservation requirements under an approved conversion model; they are not statements about present Solana supply or current POUT-holder rights. A token still freely transferable after a snapshot cannot automatically be treated as extinguished, and a source bridge/custodian needs separate solvency and failure analysis. No burn, lock, migration ratio, claim window or automatic native-token credit is live or approved.

6.7 Verification pipeline and activation dependency

`source_event -> finalized_source_evidence -> verified_allocation`
 `-> independently authorized recipient -> stable nullifier`
 `-> bounded_claim_validation -> atomic_target_credit`
 `-> durable_finalized_receipt -> reconciliation_audit`

The existing Rust simulator covers synthetic accounting invariants, not Solana light-client verification, operational source-chain finality, a deployed bridge or an audited native ledger. Migration is the planned integration path for the present POUT asset, while its precise engineering mechanism is an open design decision. The activation dependency is technical and security-based rather than calendar-based; implementation and eligibility will only be defined after those prerequisites are satisfied.

7. Research gates and conditional development sequence

The following gates describe technical dependencies, not scheduled delivery dates. Advancement depends on validated research, implementation evidence and security review—not market capitalization, trading activity or community expectations. No date for testnet release or token migration is implied.

Research gate	Research / implementation gate	Evidence required before proceeding
Service predicate	Resource-service predicate and coalition attack model; evaluate two copies / shared store / delegated responder; finish bounded negative controls.	Measured service outcomes and coalition-wide cost with explicit observer/demand assumptions; clear failed hypotheses.
Consensus model	Consensus feasibility specification with canonical accounting history and failure model.	Precisely defined eligibility/weight rule, stated adversary assumptions, independent critical review and negative controls.
Local integration	Integrate only a justified research model with deterministic execution/DAG structures in isolated local environment.	Safety/liveness arguments for the selected bounded model; recovery, fork and network reliability controls.
Controlled devnet	Controlled multi-node development network; no real value at risk.	Repeatable genesis/config, networking, fault injection, observability, platform and security gates.
Community alpha	Conditional community alpha/testnet if security and operational gates pass; migration research continues separately.	Independent participation safely constrained, published limitations, review of consensus assumptions and realistic operating budgets.

A working local chain can precede a general resource-soundness result, but it must not be marketed as secure PoUT consensus. A public permissionless network with economically meaningful security, an open validator set and token-backed settlement requires substantially stronger evidence. If the predicate fails

under collusion or shared-resource attacks, the correct outcome is redesign or deferral, not a declaration of success.

8. Failure modes, threat model and explicit non-claims

Threat	What must be tested	Currently established?
Sybil / aliases	k identities backed by one store, shared provider keys, identity changes	No physical independence guarantee
Colluding requester/provider	Provider and requester fabricate demand and receipts	No independent demand verification
Delegated responder	Node without data forwards challenge to holder	No anti-delegation lower bound
Replay / fork	Reuse of old but valid proof in divergent histories	Finite local history controls only
Preprocessing / cache	Shared advice reused across identities and intervals	Counterexamples exist; no general lower bound
Challenge manipulation	Requester/provider or attacker chooses/predicts challenge	Independent challenge issuance absent
Key rotation	New verifying key produces new logical token/opportunity	Finite conditional accounting controls
Migration duplication	Same source allocation claimed twice or two chains trade equivalent rights	No deployed claims/bridge; synthetic model only
Network stability	Fallback, deadlines, recovery, resource exhaustion	Round7 gate open

Nothing in this memorandum establishes production-ready cryptography, independent external audit, profitable utility, token appreciation, guaranteed native conversion, a token-holder entitlement, regulated status or imminent public-mainnet availability.

9. Verification, publication boundary and external technical review

The public source demonstrates a narrow verifier and CWC model; it intentionally omits unpublished consensus research, proving parameters and operational secrets. Full claims about cryptographic setup, service-resource soundness and consensus security require independent technical review and reproducible evidence. Previously published MIT/Apache-2.0 source remains subject to its existing permissions; private research is not automatically part of that public grant. Any novel mechanism should undergo technical, intellectual-property and disclosure review before source publication.

9.1 Source and evidence register

[P1] Public repository: <https://github.com/PoUTnetwork/pout-public-source-v0.1.0-2>

[P2] Public research and security limitations: docs/RESEARCH_STATUS.md and docs/SECURITY.md inside [P1].

[P3] Third-party market index: CoinGecko PoUT-Network, <https://www.coingecko.com/en/coins/pout-network> (contract abbreviated; reported maximum supply 1B; indexed market data may change).

[P4] Third-party wallet/index listing: Solflare, <https://www.solflare.com/prices/pout-network/FQ47SAH4xuRJArffifAbqZSX1kq3ewE3GxnLuGYypump/> (full mint; listing provider marks it unverified). The mint identity is independently confirmed by the project owner for this document; its on-chain parameters have not been audited.

[I1] Internal engineering status, exact-source local J1/J2 results, joint-relation controls and selected-history experiments, examined as of 20 October 2026. Private research data; no claim that the full source is public.

[I2] Internal conditional migration strategy and synthetic Rust migration/supply/claim-model evidence, examined as of 20 October 2026. This demonstrates a technical design route, not deployed bridge, finalized source-mint verification, approved claim or native entitlement.

Source reconciliation rule: public release counts, internal engineering counts, source digests, CI status and actual hosted availability must be stated separately. Any material technical claim made after this reference date should be revalidated before redistribution.

10. Technical decision register

Open decision	Required prerequisite
Exact official POUT source token identity	Owner-confirmed mint; independently verify Solana token program, decimals, authorities, supply and finalized chain state before migration.
Production service predicate	Independent observer model and coalition cost lower-bound hypothesis surviving adversarial controls.
Consensus weight	Precise formula, canonical ledger/horizon, safety/liveness and Sybil/collusion analysis.
POUT service utility	Actual service offering, supported wallet/payment settlement, refund/failure and accounting rules.
Conversion path	Planned conversion/integration workstream; choose an implementation only after source-finality, custody, double-circulation, wallet, security and legal reviews.
Native claims and reserve	Governance-approved entitlement policy, ratio, genesis reserve and no-double-circulation controls.
Release / independent audit	Frozen candidate source, independent verification, protected private IP where applicable, documented risks.