ENI6MA: A Peer-to-Peer Proof-of-Knowledge Ledger and Bank-Grade Stablecoin E6G (Eni6ma Gold)

Author
Dylan Rosario ˜ Founder Rosario Cybernetics Research Group (Rosario–Wang Proof)
research@eni6ma.co
www.eni6ma.com

Abstract

We introduce **ENI6MA**, a peer-to-peer electronic cash and settlement system that replaces proof-of-work/stake with **Proof-of-Knowledge (PoK)** grounded in the **Rosario–Wang Proof (RWP)**. Instead of expending energy or locking capital, each spend is authorized by an **ephemeral witness** tied to fresh entropy and the exact transaction context, and verified publicly with fast symmetric checks. Blocks are timestamped by **capsule commitments** and chained in a compact, threshold-signed **accumulator**, yielding seconds-class finality, constant-time input verification, strong replay resistance, and a clean, auditable security budget. ENI6MA further specifies **ENI6MA-G**, a **1-gram-of-gold per token** stablecoin with per-window, RWP-attested reserve proofs, enabling bank-grade issuance and redemption without long-lived private keys at rest or PoW externalities.

At the heart of authorization is a one-time witness computed by a sealed, symmetric **private morphism** $\mathcal{M}$ shared by prover and verifier (compiled "twins"). Fresh session entropy x, the canonical block/window time T, and the public transaction context q (referenced UTXOs, amounts, scripts, network ID, policy tags) are mapped to a transient orientation:

$$W \;=\; f_{\mathcal{M}}(x, T, q).$$

"W equals f sub M of x, T, and q."
Here $f_{\mathcal{M}}$ is a composition of domain-separated hashes and PRFs parameterized by $\mathcal{M}$; W never leaves the device and is unrecoverable out of window. Ephemer-

ality collapses the attack surface associated with static signing keys: there are **no reusable private keys** to phish, leak, or compel.

Public verification reduces to fixed-width XOR equalities over a deterministic probe set derived from (q, T, s_T), where s_T is the window's randomness beacon. The transcript τ carries masked responses ρ_i that cancel only if the spender knows the correct W:

$$\Lambda \;=\; \bigwedge_{i=1}^{h} \big(\,\mathrm{XOR}(p_i, \rho_i) = 0\big).$$

"Lambda equals the logical AND, for i from one to h, of p sub i exclusive-or rho sub i equals zero."
All symbols are public at verification time: p_i are challenges, ρ_i are responses, and h is the probe count. Acceptance is strictly conjunctive with policy:

$$\mathsf{ACCEPT} \iff \Lambda = 1 \;\wedge\; \mathsf{PolicyOK}(q).$$

"Accept if and only if Lambda equals one and Policy-O-K of q is true."
Thus, **soundness** (ephemeral knowledge) and **governance** (jurisdiction, KYC tags, caps) are cleanly separated yet jointly enforced.

Blocks bind time, randomness, and policy with **capsule commitments** and roll up content under a single **accumulator**. The header's environment capsule

$$\mathsf{com}_t \;=\; H\big(\mathsf{TAG[CAP]} \,\|\, T_t \,\|\, H(\mathsf{TAG[SEED]} \,\|\, s_t) \,\|\, \mathsf{aux}_t\big)$$

"com sub t equals hash of tag cap concatenated with T sub t, concatenated with hash of tag seed concatenated with s sub t, concatenated with aux sub t."
commits the canonical time T_t, the beacon seed s_t, and auxiliary flags aux_t. Content and policy are chained via

$$A_t \;=\; H\big(\mathsf{TAG[ACC]} \,\|\, A_{t-1} \,\|\, \mathsf{com}_t \,\|\, \mathsf{txh}_t \,\|\, \mathsf{pol}_t\big),$$

"A sub t equals hash of tag A-C-C concatenated with A sub t minus one, concatenated with com sub t, concatenated with t-x-h sub t, concatenated with pol sub t."
where A_{t-1} is the prior accumulator, txh_t summarizes the included transcripts, and pol_t is the enforceable rule word. A rotating beacon **committee** signs A_t with a threshold signature, certifying time/randomness anchoring and making fork choice trivial: extend the **longest valid, signed accumulator chain**. Because verification is hash/XOR plus one header signature, SPV clients achieve **full-node assurances for specific payments** from headers and a single Merkle branch.

Our security model replaces hash-power or stake-economic assumptions with two auditable premises: **entropy freshness** and **threshold honesty**. Single-input forgery for any PPT adversary without $\mathcal{M}$ is negligible in the security parameter λ; union bounds scale to many concurrent attempts with linear (and budgetable) risk. Replay and precomputation fail because τ binds to

(x, T, q) and probes recompute from (T, s_T, q). The hot path is **post-quantum friendly**—hashes, PRFs, XORs—while the committee's threshold signature is modular and upgradable to lattice or hash-based PQC without altering block structure or SPV flow. Crucially, there are **no private keys at rest** for spends or disclosures.

We operationalize a bank-grade stablecoin, **ENI6MA-G**, with one token per **gram of vaulted gold** and per-window reserve proofs. Each custodian publishes a **reserve attestation capsule**

$$\mathsf{res}_t \;=\; H\big(\mathsf{TAG[RES]} \,\|\, \mathsf{SKU} \,\|\, \mathsf{mass}_t \,\|\, \mathsf{vaultID} \,\|\, \mathsf{audit\_ref}\big),$$

"res sub t equals hash of tag R-E-S concatenated with S-K-U, concatenated with mass sub t, concatenated with vault I-D, concatenated with audit reference."
alongside an RWP transcript τ_t^{res} proving live control at time T_t. These capsules are bound into A_t, turning solvency into a **machine-checkable inequality** from headers alone. The "**gram code**" (1 g per token) provides a neutral unit for cross-currency valuation using public gold prices, enabling predictable FX routing and transparent treasury operations without oracle cartels or signature-key custodianship.

Empirically, ENI6MA achieves **seconds-class finality** (time-window pacing, pipeline threshold signatures), **O(1)** input verification cost (dominated by hash/XOR), compact headers for fast propagation, and **first-class SPV** on mobile. Economically, security is priced by explicit, predictable fees (bandwidth and block space), not external energy burn; committee service fees are small and policy-regulated; stablecoin reserves can even **subsidize** user fees via attestation-tied rebates. Privacy is practical by construction—pseudonymous scripts, short-lived transcripts, and selective disclosure proofs—while compliance is objective and replayable via pol_t.

In sum, ENI6MA generalizes Nakamoto's insight—**public, timestamped history**—by replacing *work* with *knowledge* as the scarce resource. The result is a ledger that is faster, cheaper, and cleaner; preserves decentralization where it matters (anyone can verify, anyone can transact); and elevates stability to a cryptographically auditable property through **per-window reserve proofs** for a **1-gram-backed** instrument that banks can operate today.

---

1. Introduction

The past decade proved that public ledgers can coordinate value transfer without central custody, yet mainstream commerce still leans on intermediaries to prevent double-spending and resolve disputes. This trust-based posture exports costs—chargebacks, data hoarding, compliance duplication—and imports privacy leakage as a feature, not a bug. Bitcoin's proof-of-work (PoW) reframed the problem: probabilistic finality through energy expenditure and open participation. But its strengths carry structural trade-offs: high latency tied to block

discovery variance, fee pressure from limited throughput, and a security budget that is literally burned.

ENI6MA takes a different path: it replaces *work* with *knowledge*. Each spend is authorized by a one-time, time-keyed **ephemeral witness** derived from fresh entropy and the exact transaction context, and verified publicly with fast, symmetric checks. Nothing reusable is exposed on chain; there are **no private keys at rest** to steal or coerce. The result is finality measured in seconds, a verification cost that is constant per input, and a security budget you can reason about with parameters rather than hash-rate economics.

At the center is the **Rosario–Wang Proof (RWP)**, a Proof-of-Knowledge primitive that binds authorization to the tuple (session entropy, canonical time, public context). The prover's device and validator share a sealed, symmetric morphism compiled from the same binary; it projects those inputs into a private manifold where membership can be checked through fixed-width XOR equalities. Because the witness is event-local and never reappears, transcripts cannot be replayed, malleated, or correlated across time.

Time itself is a first-class object. Blocks are paced by short **windows** (e.g., one second) rather than mining races. Each window publishes a **capsule commitment** that binds the canonical timestamp, an unbiased seed from a rotating beacon committee, and the active policy word. Transactions included in that window are summarized, and everything—time, randomness, content, policy—is rolled into a single **accumulator** value that the committee threshold-signs. Fork choice becomes deterministic: extend the **longest valid chain of signed accumulators**.

This structure compresses verifiers. A full node re-derives the public context for each input and runs a handful of XOR checks. A light client follows only headers and, for a specific payment, requests one Merkle branch plus the transcript, verifying locally with the same symmetric code path. In other words, SPV is not a second-class citizen; it is the default way ordinary devices achieve **full-node assurances for the transactions they care about**.

Privacy follows from non-reusability. Scripts are pseudonymous and rotate by default; the one-time transcript reveals no standing key. Linkage attacks are pushed back to the visible context (amounts, timing, change patterns), which wallets can shape with standard coin-selection policies. Where jurisdictions require disclosure, ENI6MA supports **selective proofs** anchored in headers: a payer can reveal exactly one inclusion fact without exposing unrelated history.

Governance is explicit and separate from soundness. The cryptographic core answers "*is this spend authorized for this time and context?*" while the policy word answers "*is this spend permitted under current rules?*" (jurisdiction bits, KYC tags, allow/deny lists, fee schedules, activation heights). Because policy is committed in the accumulator, enforcement is objective and replayable: any reviewer can re-validate decisions offline and reach the same outcome as the network did online.

Security assumptions shift from "honest majority of energy or stake" to two auditable premises: **entropy freshness** and a **threshold-honest** beacon committee. Both are measurable on chain—randomness proofs and signing partici-

pation—and both have direct mitigations: operator diversity, rotation, slashing, and conservative confirmation depths for high-value flows. The cryptographic hot path is hash/PRF/XOR only, making the system **post-quantum friendly** with a modular upgrade for the committee threshold signature.

Economic incentives are simple and legible. There is no block subsidy and no hardware race. Fees price bandwidth and block space; the beacon committee earns a small, policy-regulated service fee when it delivers seeds and signatures on time; and in the stablecoin context custodians can **rebate** user fees when they publish timely reserve capsules. Because verification cost is tiny and predictable, pricing maps closely to real resource consumption rather than to external energy markets.

Persistence is scalable by design. Blocks commit a Merkle root for all transcripts so nodes can safely **prune** bodies after a small, explicit finality depth while retaining headers, roots, and signatures. Full nodes maintain only the compact UTXO set and recent windows; light clients keep just headers. Archivists can store old transcripts off-chain without trust games: a later inclusion claim is either verified against the preserved root or rejected.

On top of the cash rail, ENI6MA specifies **ENI6MA-G**, a bank-grade stablecoin where one token equals one gram of vaulted, good-delivery gold. Custodians publish **reserve capsules** every window—SKU, mass, vault ID, audit reference—attested with RWP and bound into the accumulator. Solvency becomes a machine-checkable inequality from headers alone, not a quarterly PDF. The "**gram code**" then serves as a neutral unit for cross-currency settlement via public gold prices, giving banks transparent, programmatic FX without oracle cartels.

Operationally, institutions interact with three moving parts: code identity (the sealed morphism), entropy quality (local plus beacon), and header signatures (committee threshold). There are no long-lived issuer keys to escrow, rotate, or leak. Reconciliation shrinks to header checks and reserve-capsule diffs; audits become replay exercises over public data; incident response is parameterized in policy instead of improvised out of band.

Comparatively, ENI6MA preserves the decentralization that matters—anyone can verify; anyone can transact—while removing the resource asymmetries that encourage centralization in PoW (energy markets) or PoS (capital rents and complex liveness assumptions). Latency drops to human-friendly seconds, fees track bytes not joules, and the privacy surface improves because no durable identities anchor analysis.

None of this dismisses Nakamoto's insight; it **generalizes** it. Bitcoin taught the world that a public, timestamped history can resolve double-spends in an open network. ENI6MA keeps the public history and tightens the timestamps—then swaps the scarce resource: *knowledge* in the right moment and context instead of *work*. That swap unlocks faster settlement, stronger privacy, clearer audits, and a stablecoin whose reserves are visible each block.

What follows formalizes the RWP construction, the capsule and accumulator machinery, the network protocol and fork rule, the incentive model, and the reserve-attestation flow for ENI6MA-G. We present parameter choices, security

bounds under standard assumptions, and migration notes for post-quantum committees. The aim is practical: a ledger simple enough to verify on a phone, strong enough to settle institutional flows, and transparent enough that anyone can replay the rules from headers alone.

---

2. Transactions

UTXO model with RWP transcripts
In the ENI6MA ledger, value is tracked as discrete unspent transaction outputs (UTXOs). A transaction selects one or more UTXOs as inputs and produces a new set of UTXOs as outputs, preserving conservation of value. The distinctive departure from signature-based systems is that each input embeds an RWP transcript τ rather than a reusable public-key signature. The transcript certifies that the spender possessed the correct, time-keyed knowledge at the precise moment and for the precise context of this spend, without exposing a static credential that could be harvested or correlated. This yields strong replay resistance (the same transcript is invalid outside its time window or context) and collapses the attack surface associated with key storage, recovery seeds, and cross-protocol key reuse. From a validation standpoint, nodes treat τ as a self-contained proof object: mempools verify τ along with standard structural and value checks before forwarding, and block producers re-verify τ upon inclusion, achieving constant-time verification per input with only symmetric operations.

Sealed symmetric private morphism $\mathcal{M}$ and session binding
The prover's device and the network verifier share a sealed, symmetric private morphism $\mathcal{M}$ compiled from the same binary image ("twins"). $\mathcal{M}$ should be viewed as a private map: a compact, stateless program that deterministically projects fresh entropy and transaction context into a manifold where membership tests are efficient to verify yet infeasible to counterfeit without the binary. There are no long-lived keys to manage; security reduces to code identity and runtime freshness. Practical deployments harden $\mathcal{M}$ with binary self-checksums, control-flow integrity, anti-tamper markers, and offline verification modes to detect modification. Because $\mathcal{M}$ is symmetric and sealed, the attack model is shifted from "protect a secret scalar forever" to "protect code identity and entropy freshness at the time of use," which is operationally tractable in embedded, mobile, and server environments with standard attestation and measured-boot stacks.

Entropy, timestamp, and context q
Each session samples a high-entropy nonce x, binds it to a wall-clock timestamp T, and folds in a context tuple q. The context canonically includes the referenced UTXO identifiers, amounts, recipient locking script template, chain/network identifiers, and optional policy tags (e.g., jurisdiction, asset type). Domain-separation tags and fixed-width encodings prevent structural ambiguities. This binding thwarts precomputation (an adversary cannot guess x), forces freshness (the transcript is only valid near T), and guarantees non-malleability with re-

spect to the economic meaning of the spend (altering amounts, outputs, or chain flips the context and invalidates the proof). Because q is public and deterministic from the transaction body, all validators derive the same q and therefore run precisely the same checks.

Ephemeral witness definition and pipeline

$$W \;=\; f_{\mathcal{M}}(x, T, q).$$

"W equals f sub M of x, T, and q."
The function $f_{\mathcal{M}}$ is a composition of symmetric primitives parameterized by $\mathcal{M}$: a domain-separated hash of (x, T, q) feeds a pseudo-random function cascade and a small set of projection maps. The result W is an orientation in a private symbolic manifold that is deterministic for the tuple (x, T, q) yet computationally indistinguishable from random to anyone without $\mathcal{M}$. Intuitively, W is the "one-time secret" for this spend: it never leaves the device, is not stored at rest, and cannot be recomputed after the time window without the live entropy and code identity. Because $f_{\mathcal{M}}$ relies only on symmetric operations (hash, XOR, PRF), the construction is post-quantum friendly and fast on commodity hardware.

Transcript construction and public verifiability
The prover derives a transcript $\tau = \mathsf{Transcribe}(W, q)$ that encodes challenge-response material allowing third parties to verify membership of W in the correct equivalence class for context q without revealing W itself. Concretely, the verifier prepares h designated probes $\{p_i\}$ deterministically from q and block-local beacons; the transcript carries masked responses $\{\rho_i\}$ derived from W. The check reduces to a constant-time XOR equation per probe, avoiding group arithmetic, pairings, or large finite-field operations. The transcript is non-linkable beyond its explicit context because the masking depends on fresh entropy and T, and it is non-transferable because replay outside the window fails the recomputed probes.

Membership XOR test and security parameters

$$\Lambda \;=\; \bigwedge_{i=1}^{h} \left(\mathrm{XOR}(p_i, \rho_i) = 0\right).$$

"Lambda equals the logical AND, for i from one to h, of the statement: p sub i exclusive-or rho sub i equals zero."
Each equality acts as a small, independent predicate that a forger cannot satisfy better than random guessing when W is unknown. By setting the per-probe hardness (e.g., k effective bits) and the probe count h, the false-accept probability contracts as 2^{-kh}. Because the probes are derived from public context and time-beacon material, all honest verifiers arrive at the same $\{p_i\}$; because ρ_i are tied to W, only a holder of the correct ephemeral witness can cause the XORs to cancel. This yields an $O(h)$ verifier with tiny constant factors, making full-node and light-client validation equally practical at high throughput.

Acceptance rule and policy layer

$$\mathsf{ACCEPT} \iff \Lambda = 1 \ \wedge \ \mathsf{PolicyOK}(q).$$

"Accept if and only if Lambda equals one and Policy-OK of q is true." Validation is intentionally factored into cryptographic soundness and explicit policy. The Λ test certifies "this spend was authorized by live knowledge at the right time for this context," while $\mathsf{PolicyOK}(q)$ enforces non-cryptographic constraints such as asset-class rules, AML/KYC flags, jurisdictional whitelists, supply ceilings, and script versioning. Separating these layers preserves the minimality of the cryptographic core and allows chains, subnets, or institutions to evolve policy without touching $\mathcal{M}$ or $f_{\mathcal{M}}$. In mempools, transactions that fail either branch are dropped; in blocks, any failure invalidates the containing input and causes the block to be rejected.

Outputs, scripts, and composition

Outputs retain the familiar "locking script + amount" pattern. A recipient can require future spenders to present an RWP transcript satisfying a particular policy tag, combine RWP with multi-party conditions (e.g., threshold RWP across devices or cosigners), or embed time locks and covenants that constrain how descendants may spend the value. Change outputs are created exactly as in UTXO systems, enabling efficient coin selection and consolidation. Because the spending condition is a predicate over τ and q rather than a reusable public key, script templates naturally support privacy-preserving address rotation and policy upgrades by template hash, while remaining friendly to light clients through compact Merkle proofs and constant-time transcript verification.

---

3. Timestamp Server (Capsule Commitments)

$$\mathsf{com}_t \ = \ H(\mathsf{TAG[CAP]} \,\|\, T_t \,\|\, H(\mathsf{TAG[SEED]} \,\|\, s_t) \,\|\, \mathsf{aux}_t)$$

"com sub t equals hash of tag cap concatenated with T sub t, concatenated with hash of tag seed concatenated with s sub t, concatenated with aux sub t."

The capsule commitment com_t is the block's compact, tamper-evident summary of time and environment. Here, $H(\cdot)$ is a collision-resistant hash (e.g., SHA-3 or BLAKE3); $\|$ denotes an unambiguous, length-delimited concatenation; and $\mathsf{TAG[CAP]}$ is a domain-separation constant that prevents cross-protocol collisions. T_t is the canonical block time for interval t, encoded in a fixed width (e.g., 64-bit little-endian UNIX milliseconds) to eliminate parsing ambiguity. The inner hash $H(\mathsf{TAG[SEED]} \,\|\, s_t)$ binds a committee-derived seed s_t while separating the seed's domain from other hashed fields. aux_t is a structured policy word, flags and parameters that contextualize the block, such as network ID, ruleset version, and window length.

The inclusion of T_t makes the commitment a timestamp in the cryptographic sense: once com_t is published, any later attempt to claim a different time for the same block breaks preimage resistance. This anchors RWP transcripts to

a specific temporal window. Because RWP witnesses are ephemeral and time-keyed, binding time directly into com_t ensures that transcripts verified under this header cannot be replayed across windows, and that validators converge on the same notion of "now" for inclusion and finality.

The seed s_t is the block's entropy anchor. Practically, s_t is produced by a rotating beacon committee using an unbiased protocol (e.g., commit-reveal, VRF aggregation, or drand-style randomness beacons). Hashing s_t under $\mathsf{TAG[SEED]}$ prevents "seed substitution" attacks and ensures uniform mixing before it feeds downstream derivations, such as transcript probes and mempool selection randomness. Because RWP proofs depend on fresh entropy, an unpredictable s_t denies adversaries the ability to precompute transcripts for a future block context.

The policy word aux_t gives the ledger a clean separation between cryptographic soundness and governance. It can carry bitfields like $\mathsf{AUX.VER}$ (consensus ruleset version), $\mathsf{AUX.DELTA}$ (target window length), $\mathsf{AUX.NET}$ (network/chain ID), or toggles for experimental features. The cryptographic binding of aux_t inside com_t means any policy transition is immutably recorded and inheres in the block's identity; clients that do not recognize a new policy version will deterministically reject the block.

Domain separation via $\mathsf{TAG[CAP]}$ and $\mathsf{TAG[SEED]}$ is critical. Without tags, a malicious encoder could craft different tuples that collide after concatenation ("type confusion"). Tags ensure that even if two concatenations accidentally share byte patterns, they cannot be interpreted as belonging to another hash's domain. This property underwrites safe composability: the same hash function H is reused across the system without risking cross-component collision games.

Security of com_t is governed by standard hash properties. Collision resistance prevents an attacker from finding two different quadruples $(T_t, s_t, \mathsf{aux}_t)$ that yield the same com_t; second-preimage resistance prevents altering a previously published block's time or policy without detection; preimage resistance prevents reconstructing $(T_t, s_t, \mathsf{aux}_t)$ from com_t alone, which is useful if aux_t contains commitment-style subfields revealed later. Because RWP relies on symmetric operations, choosing a modern H also maintains a post-quantum posture.

Operationally, com_t unifies consensus pacing and anti-grinding. Pacing comes from T_t and windowing rules: producers can only assemble blocks whose timestamps lie within the permitted drift, and validators enforce it. Anti-grinding comes from s_t: block proposers cannot grind many candidate headers to game downstream selection logic because the unpredictable seed is injected after commit-and-sign by the beacon committee. This design removes the economic incentives that, in PoW systems, push miners to expend energy searching for "lucky" headers.

From an implementation viewpoint, com_t is a small, fixed-size digest that lives in the block header, travels well on the wire, and is simple to reconstruct by any validator. Clients that operate in SPV mode only need to fetch and hash these few fields to track time and policy transitions, while full nodes additionally use com_t as input to derive probe sets for RWP transcript checks, ensuring every validator recomputes the exact same verification challenges.

The capsule abstraction composes cleanly with future extensions. If later epochs introduce per-block auxiliary commitments, like reserve attestations for a stablecoin issuer, data-availability roots, or rollup summaries, those can be nested under aux_t as sub-commitments whose reveals occur over a bounded horizon. The outer $H(\cdot)$ keeps the header constant-size and verifiable with the same code paths, preserving the ledger's simplicity and throughput characteristics.

$$\mathsf{txh}_t \;=\; H(\mathsf{TAG[TXH]} \,\|\, \tau_1 \,\|\, \cdots \,\|\, \tau_K)$$

"t-x-h sub t equals hash of tag T-X-H concatenated with tau one through tau K."

The transaction hash txh_t is the block's content fingerprint for RWP transcripts. The values $\tau_1, \ldots, \tau_K$ are the validated, ordered RWP transcripts from the mempool selected for inclusion in block t, and K is the count of included transactions. $\mathsf{TAG[TXH]}$ is a domain-separation tag for transaction aggregation. This construction binds the exact multiset (indeed the exact sequence, if the serialization is order-sensitive) of transcripts to the block, so any post-hoc alteration of content breaks the hash and is rejected by validators.

The simplest aggregation is a straight, length-delimited concatenation of τ_i byte strings under a domain tag. This gives excellent performance and minimal overhead. In deployments that require inclusion proofs for light clients, txh_t can be defined as the Merkle root of $\{\tau_i\}$ with $\mathsf{TAG[TXH]}$ baked into the leaf hash; the rest of the header and accumulator math remains unchanged. The important property is that every validator derives the same txh_t from the same ordered set of transcripts.

Because each τ_i already encodes everything necessary to verify a spend, probe responses, context binding, and policy compliance, the aggregation hash has one job: make the block's content unforgeable in hindsight. If two producers attempt to equivocate by publishing different blocks for the same slot, the differing txh_t values make the fork explicit. Honest validators will only extend the branch whose header and committee signature are first to satisfy the acceptance rules; the other branch's txh_t serves as a forensic artifact showing exactly which transcripts diverged.

The design avoids heavy cryptography in the hot path. Computing txh_t is linear in the total transcript length and requires a single pass of H. Validation cost is dominated by per-input RWP checks, which are themselves constant-time per probe. This division keeps block creation and propagation latency low while leaving room for high transaction throughput without specialized hardware.

Ordering policy for $\{\tau_i\}$ has consensus implications. To prevent censorship-based grinding or mempool manipulation, producers can be required to sort transcripts by a deterministic key, such as $H(\mathsf{TAG[SORT]} \,\|\, \tau_i)$, or by fee-per-weight with a deterministic tiebreaker. Whatever the rule, it must be part of the consensus spec so that any two honest producers assembling the same candidate set produce the same txh_t.

In fraud investigation and audit scenarios, txh_t facilitates succinct account-

ability. A regulator or exchange can ask for the transcripts corresponding to a suspicious block; a light client can verify inclusion by requesting the transcript bytes plus a Merkle branch (if Merkle-ized) and recomputing txh_t. No signature keys or confidential material are needed to check the evidence, preserving the system's "public verifiability without private keys at rest" philosophy.

Because RWP transcripts are ephemeral and context-bound, txh_t also acts as a "forgetting boundary." Old blocks can prune transcript bodies while retaining txh_t (or the Merkle root) and the header. This allows archival nodes to compact storage without compromising verifiability: a future accuser can always reintroduce the transcript body from an external archive and demonstrate inclusion against the preserved txh_t.

Finally, $\mathsf{TAG[TXH]}$ ensures transcript aggregation cannot be confused with other header-level commitments. If a future upgrade adds a parallel commitment for data blobs, logs, or rollup states, the tag separation prevents adversaries from attempting cross-component hash replay attacks where a fake set of transcripts could masquerade as blob data or vice-versa.

$$A_t \;=\; H(\mathsf{TAG[ACC]} \,\|\, A_{t-1} \,\|\, \mathsf{com}_t \,\|\, \mathsf{txh}_t \,\|\, \mathsf{pol}_t)$$

"A sub t equals hash of tag A-C-C concatenated with A sub t minus one, concatenated with com sub t, concatenated with t-x-h sub t, concatenated with pol sub t."

The accumulator A_t is the ledger's one-way spine. It chains the entire history by hashing the previous accumulator A_{t-1} with the current capsule commitment com_t, the transaction hash txh_t, and a policy word pol_t. $\mathsf{TAG[ACC]}$ provides domain separation for the chain accumulator. Once a block is finalized, altering any constituent, time, seed, policy, or transcript set, changes A_t, which in turn changes every subsequent $A_{t+1}, A_{t+2}, \ldots$. This is the cryptographic root of immutability.

The previous accumulator A_{t-1} gives the construction its inductive strength. Any successful second-preimage on a past block must also collide the entire suffix of the chain, which is infeasible under standard assumptions for modern hash functions. This property mirrors the "block header chain" of PoW systems but without the need for difficulty targets or energy expenditure; security rests on the committee's signed time/seed and the hash chain's resistance to history rewrites.

pol_t is a consensus-layer policy word distinct from aux_t. It carries the rules that validators actually enforce when deciding to accept A_t. Examples include thresholds for the beacon signature scheme, allowed ranges for timestamp drift, fee schedule parameters, and activation heights for soft forks. By hashing pol_t into A_t, the block irrevocably records the rule context under which it should be interpreted, allowing future clients to re-validate history even across multiple governance eras.

The accumulator is signed by the beacon committee and placed in the block header. A threshold signature $\sigma_t = \mathsf{Sign}_{\mathcal{C}}(A_t)$ certifies that a qualified majority ($\geq t$ of n) agrees on the time window and seed for that interval. Validators check

σ_t before considering the block for extension. Because the signature covers A_t, it implicitly covers com_t and txh_t as well, tying the randomness and content to the committee's attestation.

Fork choice becomes "extend the longest valid, correctly signed accumulator chain." There is no difficulty metric; instead, validity hinges on cryptographic checks (RWP and signatures) and policy rules. If two branches exist for the same slot, only one can carry a valid committee signature, or only one will be extended as subsequent slots receive signed accumulators. Honest nodes automatically converge on the branch with continuous committee attestations, providing deterministic finality after a small number of windows.

The accumulator makes SPV simple and robust. A light client need only track the sequence of (A_t, σ_t) pairs and, for a payment, request the relevant txh_t inclusion proof plus the transcript. The client verifies the committee signatures, recomputes A_t locally, and checks the RWP equations. No heavy state is required, and no long-term public keys must be pinned other than the committee's rotating verification keys and hash-function parameters.

From a performance perspective, computing A_t is a single hash over fixed-size header fields plus the one digest txh_t. This keeps header sizes tiny and header-to-header validation constant-time. Combined with the RWP verifier's symmetric-only arithmetic, the ledger attains high throughput and low latency without specialized accelerators, and remains resilient against future quantum speedups that disproportionately threaten asymmetric primitives.

Finally, the accumulator structure is future-proof. If the network later adopts post-quantum threshold signatures (e.g., lattice-based) or adds auxiliary commitments (e.g., data-availability roots), those elements are either captured inside com_t or added as new tagged fields under pol_t, while the outer accumulator A_t and its validation logic remain unchanged. This modularity keeps consensus code compact, auditable, and amenable to formal verification, aligning with the ENI6MA design goal: minimal trusted state, maximal verifiability.

---

4. Proof-of-Knowledge in Lieu of Proof-of-Work

Where PoW burns energy to throttle block production and provide Sybil-resistance, ENI6MA uses **entropy-gated knowledge**. Security rests on three pillars:

1. **Ephemerality**: Witness W binds to (x, T, q). Replays at a different time or context fail.

2. **Statelessness**: No reusable private key or long-term secret exists outside $\mathcal{M}$ (sealed inside compiled twins).

3. **Symmetric verification**: Checks are hashes, XORs, and a compact committee signature.

We model single-input soundness as:

$$\Pr[\mathsf{ACCEPT}] \leq \mathsf{negl}(\lambda) \quad \text{for any PPT adversary without } \mathcal{M}.$$

"The probability of accept is at most negligible of lambda for any polynomial-time adversary without script M."

Multi-input soundness aggregates (union bound):

$$\Pr[\exists i \text{ forged}] \;\leq\; h \cdot \mathsf{negl}(\lambda).$$

"The probability that there exists an i that is forged is at most h times negligible of lambda."

4. Proof-of-Knowledge in Lieu of Proof-of-Work

$$\Pr[\mathsf{ACCEPT}] \leq \mathsf{negl}(\lambda) \quad \text{for any PPT adversary without } \mathcal{M}.$$

"The probability of accept is at most negligible of lambda for any polynomial-time adversary without script M."

This bound formalizes **single-input soundness** for an RWP spend. The event ACCEPT means a verifier, running only symmetric checks, would deem a forged transcript valid. The symbol λ is the **security parameter** (e.g., bit-lengths for hashing, masking, and probe hardness); increasing λ makes attacks exponentially harder. The function $\mathsf{negl}(\lambda)$ denotes any **negligible** function—one that shrinks faster than $1/\lambda^c$ for every constant c. "PPT adversary" means any **probabilistic polynomial-time** attacker with oracle access matching our threat model, but crucially **without** the sealed morphism $\mathcal{M}$ (the symmetric private map compiled into the prover/validator twins). Intuitively, unless an attacker controls the exact code identity and the live entropy/time/context tuple, their chance to counterfeit a valid transcript for one input vanishes with λ.

The equation's left side quantifies the **best possible** success chance of a forger against one input under adaptive strategies. It already accounts for the fact that the adversary can choose the context q adversarially, eavesdrop on network traffic, and schedule queries in time; the bound still holds because the witness W is bound to fresh entropy x, timestamp T, and the very q the adversary hopes to subvert. Even if the attacker predicts or influences q, they cannot reconstruct $W = f_{\mathcal{M}}(x, T, q)$ without both the live entropy and the code identity $\mathcal{M}$.

The clause "without $\mathcal{M}$" is more than bookkeeping; it captures the **statelessness** doctrine. Traditional signature schemes place a long-lived secret scalar at rest; compromising it collapses security forever. Here, the only privileged asset is **code identity**—the compiled morphism $\mathcal{M}$. If $\mathcal{M}$ isn't in the attacker's possession, transcripts are information-theoretically opaque with respect to W, and computationally unforgeable with respect to the XOR-membership tests and the hash commitments. This shifts operational risk from perfect key custody to standard software integrity (measured boot, attestation, reproducible builds).

Mechanistically, the negligible bound arises from two layers. First, **projection secrecy**: given τ and q, inferring W (or any predicate that helps pass a fresh probe set) is computationally infeasible. Second, **probe hardness**: for any fixed probe set derived from q and the block seed, the probability that a transcript computed **without** W passes all XOR equalities is bounded by a term that collapses with λ (e.g., 2^{-kh} with k effective hardness bits per probe and h probes). The reduction maps a forger for ACCEPT to either a predictor for the masked responses (breaking projection secrecy) or to a distinguisher that contradicts the assumed pseudorandomness of the masking and hash oracles.

Ledger-wise, this single-input bound is the guarantee that **one counterfeit cannot sneak in** despite full network observation. Mempools can admit only those transactions whose transcripts pass deterministic checks; block producers re-run exactly the same verifier. Because verification is symmetric-only and $O(h)$ per input, raising λ to shrink $\mathsf{negl}(\lambda)$ does not create a performance cliff. This property lets the ledger target bank-grade finality without sacrificing throughput or pushing costs to specialized hardware.

The time binding is central to defeating **replays and precomputation**. Even if an adversary records τ today, they cannot use it tomorrow: the probe set changes with the seed and time, and the verifier recomputes everything from public inputs. Likewise, precomputing a catalog of transcripts for possible q values is useless without the future entropy and timestamp. The negligible bound therefore encapsulates both computational unforgeability and **freshness** constraints in a single expression.

A natural question is what happens under **partial compromise**, e.g., device theft after a spend. The bound still protects **past** transactions (immutability) and **future** ones if the attacker cannot preserve the exact runtime environment, entropy source, and measured $\mathcal{M}$. Because there is no reusable private key to extract, post-compromise damage is sharply limited in time; revocation policies can simply reject transcripts whose device attestation has flipped since last measurement, again without weakening the negligible acceptance bound for honest inputs.

Finally, observe the **post-quantum posture** implicit in the inequality. The reduction relies on generic properties of cryptographic hashes, PRFs, and XOR masks—primitives for which quantum speedups do not create catastrophic asymmetries (Grover's gives at most square-root speedup, absorbed by a modest bump in λ). Thus, $\Pr[\mathsf{ACCEPT}]$ remains negligible even against quantum-capable PPT adversaries, provided parameter sizes are chosen accordingly and the committee signature covering the header is upgraded to a PQ threshold scheme.

$$\Pr[\exists i \text{ forged}] \;\le\; h \cdot \mathsf{negl}(\lambda).$$

"The probability that there exists an i that is forged is at most h times negligible of lambda."

This inequality is a **union bound** that scales single-input soundness to **many concurrent opportunities** for forgery. The event $\{\exists i \text{ forged}\}$ means

"at least one of the i under consideration was forged successfully." The symbol h here denotes the **number of independent forgery opportunities** the analysis is accounting for—this could be the number of inputs in a transaction, the number of adversarial queries allowed in a block interval, or any bounded set of parallel attempts. Because the same letter h also names the probe count elsewhere, implementers typically rename this population size to m or N in the formal spec to avoid confusion; the inequality itself remains valid for any such h.

The union bound is **worst-case conservative** and does not assume independence. Even if an adversary cleverly correlates their attempts—choosing contexts q_i adaptively after observing partial failures—the probability that **any** attempt succeeds is at most the sum of the per-attempt probabilities. Since each attempt is individually bounded by $\mathsf{negl}(\lambda)$ (by the first inequality), the total risk scales at most linearly with h. In practice, the actual risk is often **much lower** because attempts share randomness sources and policy constraints, but the union bound gives a clean, audit-friendly ceiling.

For ledger engineering, this inequality translates directly into **parameter selection**. Suppose a block can carry up to 10^4 inputs and policy treats all of them as adversarially chosen. If $\mathsf{negl}(\lambda) \approx 2^{-128}$, then $h \cdot \mathsf{negl}(\lambda) \leq 10^4 \cdot 2^{-128} \approx 2^{-114}$, still astronomically small. If the ecosystem anticipates orders of magnitude more parallel attempts (e.g., rollups verifying many proofs per slot), one simply increases λ or the per-probe hardness k to keep the aggregate risk below a fixed budget—without changing code paths or economic incentives.

The bound also informs **fee and DoS policy**. Because every additional adversarial attempt increases aggregate forging risk linearly, the network can price or rate-limit high-fan-out transactions or repeated failures in mempool admission. This keeps h within expected envelopes, making the residual risk predictable and letting operators certify quantitative security targets (e.g., "less than 10^{-18} probability of any forged input per day at peak load").

A subtle but important implication is **graceful degradation** under imperfect randomness. If the beacon seed or local entropy source experiences minor bias, single-input soundness might inflate the per-attempt bound by a small factor β, becoming $\beta \cdot \mathsf{negl}(\lambda)$. The union bound then inflates to $h\beta \cdot \mathsf{negl}(\lambda)$. Because the ledger monitors entropy quality (via self-tests, committee cross-checks, and on-chain measurements), it can respond dynamically—e.g., shrink per-block h, raise probe counts, or increase finality windows—keeping the aggregate below the target threshold without halting the chain.

In adversarial **multi-context** scenarios—say, a malicious exchange trying to double-spend across several merchants simultaneously—this inequality states that protecting each merchant individually at negligible risk suffices to protect **all of them jointly** at a comparably negligible level, given sane h. There's no combinatorial explosion in the analysis; the ledger's security budget composes linearly across concurrent verifications.

From a compliance and audit standpoint, the union bound is the bridge between **cryptographic guarantees** and **operational guarantees**. Risk officers can convert "negligible in λ" into concrete service-level numbers by plugging

in worst-case h for their institution's flow—per hour, per day, per peak event. Because both sides of the inequality are explicit, these assurances can be made contractual without revealing proprietary internals of $\mathcal{M}$ or the exact probe construction.

Overall, the union bound interacts cleanly with the **committee-honesty** and **entropy-freshness** assumptions that replace PoW's hash-power majority. Even if an adversary controls network scheduling or attempts to bias seeds within some failure probability ε per slot, the total probability that **either** a committee failure occurs **or** any of the h attempted forgeries slip through is at most $\varepsilon + h \cdot \mathsf{negl}(\lambda)$ (by another union bound). This additivity keeps the threat model modular: improvements in beacon resilience directly subtract from ε, while cryptographic parameter choices bound the second term—together yielding a transparent, composable security budget for the ledger.

---

5. Network

The network runs as follows:

1. New transactions broadcast to all nodes with transcripts τ.
2. Nodes validate inputs by running the RWP checks and policy.
3. Nodes assemble blocks for the current time window, binding com_t and txh_t.
4. The beacon committee signs A_t (threshold).
5. Nodes accept a block iff all included τ verify and the header signature is valid.
6. Nodes extend the longest valid accumulator chain A_t.

Forks resolve by **longest valid chain** measured in signed accumulator steps (no difficulty). Because blocks are paced by **time windows** (not PoW), parameters (e.g., 1–2 s cadence) can target low-latency settlement.

5. Network

1) New transactions broadcast to all nodes with transcripts τ.

A transaction enters the network as a compact object containing its inputs, outputs, and one RWP transcript τ per input. The transcript τ is the public, ephemeral proof artifact—no long-lived signature or public key is revealed—so onlookers learn only what is necessary to verify that the spend is authorized for this specific time and context. Because τ is time-keyed and entropy-keyed, the

object is useless to replay after its intended window; this dampens the incentive for mempool flooding with stale data.

The broadcast layer uses gossip with duplicate suppression and content addressing: nodes announce a transaction by its hash and serve the body on request. This keeps bandwidth bounded under contention and allows nodes under DDoS to handle "inv" messages faster than they handle payloads. The transaction hash is derived from a canonical serialization that includes the RWP transcripts in a fixed order so that two honest broadcasters always identify the same bytes with the same digest.

Admission control starts at the edge: before a node commits bandwidth, it performs a quick, constant-time precheck—payload size limits, structural sanity, and a shallow RWP screening that verifies domain tags and transcript framing. These checks are symmetric-only and take microseconds, which means hostile peers cannot force expensive work through malformed objects. If the precheck passes, the node requests the bytes and schedules full RWP and policy validation.

To resist eclipse and partition attacks, peers prefer diverse network paths and delay acceptance of "first seen" transactions that originate from a single neighbor. A randomized relay map plus per-peer rate limits deters one peer from becoming your entire view of the world. Nodes also apply rolling bloom filters per neighbor to detect repeated spam and to quarantine misbehaving peers without disconnecting healthy ones.

Privacy is preserved because τ does not reveal a reusable public key. Still, first-hop metadata can deanonymize broadcasters. To mitigate this, wallet software uses randomized broadcast delays, dandelion-like stem/flare routing, and optional relays through well-known "transaction fountains" to blur the source. None of these behaviors are trusted; they simply add noise to timing analysis without affecting verifiability.

Gossip remains robust in the face of packet loss. Announcements are cheap; full payloads are fetched lazily and retry on timeout. If a node hears about a transaction that fails RWP precheck, it records a short "seen-bad" memo keyed by the hash to avoid re-requesting the same junk from other peers. This memo expires quickly to avoid becoming an oracle for censorship.

Finally, fee signaling is embedded in the transaction body, and nodes maintain a mempool policy that weighs fee-per-weight and age. Because RWP verification is cheap, fee policy is less about paying for CPU and more about protecting scarce bandwidth and block space. The broadcast step therefore aligns economic incentives (fees) with operational costs (bandwidth and inclusion probability) without introducing asymmetries attackers can easily game.

2) Nodes validate inputs by running the RWP checks and policy.

Full validation begins with reconstructing, from the transaction's public fields, the exact context q used by the prover and deriving the designated probe set from (q, T_t, s_t). For each input, the node runs the XOR-membership predicates encoded in τ. These are constant-time checks over fixed-width words, which lim-

its timing side-channels and makes batching effective: validators can vectorize probe checks across inputs to amortize memory accesses.

Policy validation is intentionally separate from cryptographic validation. Once Λ evaluates true for an input (i.e., all XOR constraints cancel as expected), the node applies $\mathsf{PolicyOK}(q)$: supply caps for the relevant asset class, jurisdiction tags, AML patterns, and script-level constraints such as timelocks or covenant rules. This separation keeps the core proof minimal while giving chains and institutions freedom to evolve regulatory or business rules without touching $\mathcal{M}$ or the RWP verifier.

Double-spend protection inside the mempool marks UTXOs tentatively spent by newly validated transactions. Conflicts are resolved deterministically—typically by higher fee-per-weight and then by a tiebreaker derived from a domain-separated hash of the transaction. Deterministic conflict resolution eliminates "who saw it first" races and closes a class of censor-grinding attacks where adversaries try to force inconsistent views of mempools across the network.

RWP's ephemerality removes the class of "signature malleability" issues that plague some asymmetric schemes. There is nothing to tweak in τ that preserves validity while changing the hash; any byte change breaks at least one XOR equality. This simplifies transaction ID stability and reduces the surface for third-party malleation attacks that could otherwise strand dependent transactions in mempool limbo.

Nodes also perform resource accounting at validation time. If an input's transcript is overly large, exceeds the allowed probe count, or requires unsupported policy primitives, the transaction is rejected under consensus rules. Because the acceptable envelope is enforced identically at all honest nodes, an attacker cannot rely on a permissive minority to propagate pathological objects into blocks.

To defend against side-channel attempts that time the RWP verifier, implementations make the XOR checks uniform and avoid early-exit branching. Even if this costs a few extra cycles on invalid inputs, it prevents per-probe timing leaks that could reveal structure about W or the masking scheme. The result is a validation routine that is not only fast but also safe to expose to untrusted inputs at line rate.

Finally, validation emits clear diagnostics. Nodes annotate rejected transactions with reason codes—bad format, policy fail, double spend, expired window—without divulging internal seed material or probe sets. This helps honest users debug mistakes and gives operators data to tune mempool and fee policies, while preventing adversaries from learning anything useful for adaptive forgeries.

3) Nodes assemble blocks for the current time window, binding $\mathbf{com}_t$ and $\mathbf{txh}_t$.

A "time window" is the consensus pacing unit (e.g., one second). During a window, block producers collect validated transactions from their mempool and lay them out in a deterministic order (fee-based with a deterministic tiebreaker).

Determinism matters: if two producers see the same mempool at the same time, they should compute the same txh_t, which reduces accidental forks and simplifies replay safety for light clients.

Producers compute the capsule commitment $\mathsf{com}_t = H(\mathsf{TAG[CAP]} \,\|\, T_t \,\|\, H(\mathsf{TAG[SEED]} \,\|\, s_t) \,\|\, \mathsf{aux}_t)$. Here, T_t is the canonical block time for the window, s_t is the beacon seed released for this window, and aux_t carries flags such as ruleset version and network ID. Binding all three under domain-separated tags ensures that any later attempt to reinterpret the header or to swap seeds between windows is detected immediately by hash mismatch.

The content hash txh_t is computed over the ordered transcripts: either as a straight, length-delimited concatenation under $\mathsf{TAG[TXH]}$ for maximum speed or as a Merkle root if inclusion proofs are required. The chosen form is a consensus parameter; both preserve the property that changing even one transcript changes txh_t, which then changes the accumulator A_t and invalidates the block.

Assembling within a bounded window eliminates incentive to "grind" for lucky headers. Producers cannot iterate nonces looking for better lottery outcomes because there is no PoW field and the seed s_t is exogenous. This removes an entire class of miner extractable value behaviors tied to header space and forces competition where it belongs: in fee-aware, deterministic selection of well-formed transactions.

The block body remains compact. Because RWP transcripts are already minimal and self-verifying, there is no need for bulky signature witnesses or script interpreters in the hot path. This compactness increases the number of transactions that fit per window and reduces propagation time, which in turn reduces fork rates and improves the probability that the network converges on the same head quickly.

Producers pre-announce their candidate header (sans signature) to neighbors as soon as com_t and txh_t are computed. Early header relay primes validators to request the body and stage verification, tightening end-to-end latency. If multiple candidates appear for the same slot, the deterministic ordering plus committee signature (next step) resolves which one is admissible.

Finally, assembly enforces soft resource caps: maximum block weight, maximum transcripts per block, maximum per-sender footprint. These caps bound worst-case verification time and prevent a single actor from monopolizing a window. Because caps are encoded in aux_t and pol_t, changes are explicit on chain and enforced uniformly by all honest nodes.

4) The beacon committee signs A_t (threshold).

The accumulator $A_t = H(\mathsf{TAG[ACC]} \,\|\, A_{t-1} \,\|\, \mathsf{com}_t \,\|\, \mathsf{txh}_t \,\|\, \mathsf{pol}_t)$ becomes the header's digest of record. A threshold t-of-n committee—diverse, rotating members—produces a single, compact signature σ_t over A_t. Validators accept only headers carrying valid σ_t under the committee's current public verification key set, which is itself tracked on chain with explicit activation epochs.

Signing A_t rather than its parts binds time, randomness, content, and policy in one stroke. An adversary cannot mix-and-match a seed from one window with

transactions from another, nor can they salvage a valid signature over com_t to bless different content. The threshold reduces single-point risk: compromising fewer than t members does not allow header forgery, and honest members can refuse to sign if they detect policy or content violations.

Committee beacons generate s_t via unbiased protocols (e.g., verifiable random functions aggregated and revealed with commit-reveal). Members precommit to randomness, reveal after the window opens, and contribute signature shares on A_t. Slashing or ejection rules in pol_t penalize failures to participate, equivocation, or detectable bias, keeping incentives aligned and the randomness stream healthy.

The threshold scheme is upgradeable. Early networks might use classical BLS threshold signatures for compactness; later, a governance upgrade can switch to hash-based or lattice-based post-quantum thresholds. Because the signed message is just A_t, the surrounding validation code hardly changes; only the signature verification module and key management rotate, minimizing consensus churn.

Operationally, the committee is not a censorship oracle. Its mandate is to attest to time and randomness and to sign the accumulator if and only if the header matches the published s_t and satisfies consensus policy. Transaction inclusion remains in the hands of block producers; misbehavior by committee members is detectable (missing shares, inconsistent seeds) and punishable on chain.

To reduce latency, signature aggregation is pipelined with block propagation. Nodes begin distribution of the candidate header while the final signature is still assembling; as soon as the threshold is met, the compact σ_t is injected and the header becomes admissible. This pipelining keeps slot times tight without sacrificing the cryptographic guarantee that every accepted block is committee-attested.

The signed accumulator acts as a **time certificate** for light clients. A wallet that only follows headers can verify σ_t, check the chain of accumulators, and, when presented with a payment, request the transcript and a Merkle proof against txh_t. No heavy state, no trusted RPCs, and no long-lived public keys beyond the committee's rotating set are required for strong payment security.

5) Nodes accept a block iff all included τ verify and the header signature is valid.

Admission is strictly conjunctive: cryptographic soundness $\wedge$ policy compliance $\wedge$ valid committee signature. If any included transcript τ fails its XOR-membership checks, or if $\mathsf{PolicyOK}(q)$ fails for any input, the entire block is invalid. Similarly, if σ_t does not verify under the current committee key set, validators reject the block regardless of its contents or fees.

This "all-or-nothing" rule raises the cost of adversarial inclusion attempts. A block producer that tries to sneak in even one bad transcript risks losing the entire window's fees and reputation when validators discard the block. Because RWP verification is cheap, honest validators can afford to re-verify everything

before extending, making it infeasible for an attacker to rely on lazy peers.

Consensus code re-derives all header commitments locally. Validators recompute com_t from $(T_t, s_t, \mathsf{aux}_t)$ and rebuild txh_t from the ordered body. This defends against "header/body mismatch" attacks in which a producer assembles a well-formed header but ships a divergent body to split views. Any mismatch results in deterministic rejection and peer reputation downgrades.

To keep admission fast, nodes pipeline checks: they verify σ_t as soon as the header arrives, then stream the body and validate τ as bytes come in. If either stage fails, the node aborts further processing and informs peers of the rejection reason. This protects against bandwidth exhaustion attacks where adversaries try to force nodes to download large invalid bodies.

Admission also advances the node's **double-spend frontier**. Once a block is accepted, UTXOs consumed by its transactions are finalized in local state. Subsequent blocks that attempt to consume the same UTXOs are invalid by construction and can be rejected early. Because inputs are RWP-authorized, there is no ambiguity about ownership: either the ephemeral witness matched the context and the input is spent, or it did not and the input remains unspent.

Finally, acceptance triggers the node's archival and pruning logic. Depending on configuration, a node might retain only headers and txh_t roots after N confirmations, discarding transcript bodies while keeping the ability to verify inclusion proofs later. This keeps disk footprint predictable while preserving the public-verifiability ethos that any honest observer can reconstruct the state from compact data and shared code.

6) Nodes extend the longest valid accumulator chain A_t.

Fork choice is "longest **valid** chain by signed accumulator steps." Validity precedes length: a chain with more steps but an invalid signature or bad block anywhere is non-viable. Among valid chains, nodes choose the one with the greatest number of consecutively signed accumulators from genesis (or from the last finalized checkpoint). This rule is simple to implement and avoids tie-breakers based on stake or work.

Because windows are paced by time rather than by PoW, simultaneous blocks for the same window can occur. The fork choice rule handles this by waiting for subsequent windows to accumulate. The branch that continues to receive correctly signed accumulators wins; the other branch stalls. With a 1–2 second cadence and typical network latency, convergence happens quickly, and applications can pick a finality depth Δ (e.g., 3–5 windows) that balances risk and user experience.

Nodes maintain a small fork buffer that tracks competing heads for recent windows. For each candidate branch, they store the sequence of A_t values and signatures, along with minimal metadata. If a branch fails validation at any height, it is pruned immediately. This bounded buffer prevents memory blow-ups during adversarial bursts and ensures that honest nodes spend CPU only on branches with a real chance to become canonical.

Fork choice is resistant to grinding because there is no "difficulty" knob and

no nonce space to explore. The committee's seed s_t is exogenous and unbiased, and header commitments are deterministic functions of publicly known data. An adversary cannot improve their odds of winning a tie by iterating many header candidates; they can only try to produce valid content and propagate it quickly, which is exactly the behavior we want from honest producers.

In partitions, both sides can build locally longest valid chains, each with properly signed accumulators if the committee is also partitioned. When connectivity returns, the rule deterministically selects the branch with more consecutive signed steps. To minimize user impact, applications can require a slightly deeper Δ during suspected partitions (detected by missing peers or elevated orphan rates) and return to normal once the network stabilizes.

Light clients implement fork choice purely over headers and signatures. They do not need to download bodies to decide which head to follow, which is crucial for mobile and embedded devices. When a payment is observed, they fetch only the relevant transcript and a Merkle branch for the specific A_t that currently leads, verify the RWP checks, and present a confidence estimate based on Δ.

Finally, the rule composes with governance. If a new committee is appointed or keys are rotated, pol_t encodes the activation epoch. After that epoch, only signatures under the new key set are valid; any chain that continues under the old keys immediately becomes invalid regardless of its length. This clear boundary prevents ambiguous transitions and makes upgrades a matter of header verification, not social coordination.

Fork resolution and time windows (low-latency settlement).

Time-window pacing decouples security from energy and lets the network target human-friendly finality. With a one-second window, merchants can see a customer's payment land in a block within a second and reach practical finality after a handful of signed steps. Because validation is symmetric-only, nodes keep up with this pace even on modest hardware; bandwidth, not CPU, is the limiting resource, and the protocol keeps payloads compact.

Window discipline relies on a tolerable clock skew bound enforced by pol_t. Validators reject headers whose T_t differs from local time by more than the allowed drift, and producers are required to align their slotting to the same cadence. This creates a global rhythm without a central clock: the committee's signed headers serve as the canonical heartbeat that all honest nodes can verify.

Low latency does not mean low rigor. Each window's header still binds s_t and content; each block's transcripts are still fully checked. The only difference is that the network removes miner lotteries and replaces them with an attested metronome. In practice, this reduces the stale-block rate, improves throughput, and makes user experience predictable—payments settle in seconds, not minutes.

The residual reorg risk is pushed into explicit, tunable parameters. Applications set Δ according to value at risk and observed network health. Exchanges might require more windows for large deposits; consumer payments can clear after fewer. Because the fork choice rule is simple and the committee's behavior

is observable on chain, these policies can be transparently justified and audited.

Fairness improves when everyone plays to the same clock. Without PoW, there is no advantage to hoarding specialized hardware or cheap electricity. The scarce resource is inclusion bandwidth, sold via fees, and the scheduling rule is deterministic. This makes the system more geographically inclusive and lowers barriers to entry for validators and service providers.

Time windows also enable clean integration with real-world oracles and reserve attestations. Custodians can publish RWP-attested state each window, and anyone can verify that the reserve capsule for time T_t is bound into com_t and thus into A_t. This turns the ledger into a synchronized tape for both cash-like transfers and regulated disclosures, a combination difficult to achieve in PoW systems with variable block times.

Finally, because the protocol exposes minimal moving parts—headers with $\{A_t, \sigma_t\}$, compact bodies with τ, and a single fork rule—implementations are amenable to formal verification and differential fuzzing. This verifiability, coupled with low-latency operation and explicit security budgets, is what gives the RWP ledger its practical strength: it is simple to reason about, fast to use, and stubbornly hard to subvert.

---

6. Incentive

Without mining rewards, incentives derive from:

- **Transaction fees** paid to validators/relays that supply bandwidth, storage, and DoS protection.
- **Beacon committee service fees** (small, regulated).
- For ENI6MA-G (gold-backed stablecoin), **custodian rebates** can subsidize network fees when reserve attestations are posted, aligning economic incentives with regulated issuers.

Fee markets are explicit and predictable; no energy cost externality is required for security.

ENI6MA's incentive model is intentionally minimal and transparent because security does not rely on burning energy or amassing hash power. The three explicit sources are transaction fees, a small service fee for the beacon committee, and—when operating ENI6MA-G—a custodian-funded rebate that subsidizes user fees when reserve attestations are posted. Each of these values is visible on-chain and parameterized in the policy word pol_t, so operators and auditors can reconstruct who paid what, for which service, in which block window. The absence of a mining reward eliminates the arms race for specialized hardware, shifting competition to efficient validation, propagation, and honest operation.

Transaction fees compensate the actors who actually incur costs: validators and relays that provide bandwidth, storage, and DoS filtering. Fees are as-

sessed on a predictable **fee-per-weight** schedule where "weight" is a consensus-measured proxy for bytes on the wire and verification work (primarily the size of transcripts τ and the number of inputs). Because RWP verification is symmetric-only and constant-time per probe, "work" is tightly bounded and public, making price discovery far easier than in systems where verification cost varies wildly with script complexity. Users can estimate costs ahead of time; validators can provision capacity with confidence.

Fee distribution is aligned with network health. A block producer receives the majority share for assembling a valid block in its window, but a configurable fraction may be earmarked to upstream relays that can present lightweight forwarding receipts proving they helped disseminate the included transactions promptly. This **relay share** deters selfish propagation and rewards nodes that keep the network well-connected under load. Implementation-wise, receipts are just short hash attestations rolled into the next block's auxiliary fields and paid out deterministically based on pol_t.

The beacon committee's service fee is deliberately small and **regulated by policy**. Its purpose is to fund the generation of unbiased seeds s_t, threshold signing of accumulators A_t, and the operational security needed to keep members diverse and honest. Because the committee's job is attestive—not discretionary—compensation is tied to objective delivery: a fee is earned only when the committee publishes the seed and a valid threshold signature σ_t for the window. Missed or equivocated duties invoke slashing or forfeiture as encoded in pol_t, ensuring the fee line item translates into reliability, not rent-seeking.

For ENI6MA-G, **custodian rebates** create a virtuous loop between reserve transparency and user costs. When a qualified custodian posts a reserve attestation capsule in a window (bound via com_t), a precommitted rebate budget offsets that window's transaction fees for all users, or for transactions involving the stable asset, depending on policy. This ties economic advantage to verifiable disclosure: the more consistently custodians prove reserves, the lower the effective fees the ecosystem experiences, accelerating adoption without compromising security or neutrality.

Fee markets remain explicit and predictable because **no energy cost externality** needs to be recovered by the protocol. The network can adopt a base-fee + tip design (akin to EIP-1559) where a block-to-block base fee responds smoothly to observed utilization, and an optional tip decides ordering among equally priced transactions. With deterministic ordering and bounded verification cost, fee volatility is dampened compared to PoW systems: users aren't competing against stochastic block times or hash-rate shocks, but against a fixed window cadence and capacity targets published in aux_t.

From a security perspective, fees directly price **DoS and Sybil attempts**. Every broadcast consumes scarce bandwidth and block space; every validation is a small, bounded cost. By requiring even small fees for inclusion—while allowing zero-fee gossip to be rate-limited—attackers must spend real value to sustain floods, and honest nodes can tune acceptance thresholds without risking liveness. Because admission policies and caps are uniform and on-chain parameterized, there is little room to exploit permissive islands to smuggle oversized

payloads.

Finally, the incentive layer is governable without ambiguity. All parameters that affect economics—fee caps, relay share, committee fee schedule, rebate rules—are versioned in pol_t and therefore hashed into A_t. Changes are visible, auditable, and replayable by light clients from headers alone. This keeps the **economic contract** between users and operators crisp: services rendered are measurable; fees paid are justified by verifiable work; and security derives from well-priced, bounded resources rather than external subsidies or opaque privileges.

---

7. Reclaiming Disk Space

Transactions in each block are Merkle-ized for pruning while preserving header hashes:

$$\mathsf{mrk}_t \;=\; \mathsf{MerkleRoot}(\tau_1, \ldots, \tau_K).$$

"m-r-k sub t equals the Merkle root of tau one through tau K."

Older blocks can safely prune transaction bodies, retaining $(A_t, \mathsf{com}_t, \mathsf{mrk}_t)$ and committee signatures. Light clients keep headers only.

$$\mathsf{mrk}_t \;=\; \mathsf{MerkleRoot}(\tau_1, \ldots, \tau_K).$$

"m-r-k sub t equals the Merkle root of tau one through tau K."

The value mrk_t is the compact fingerprint of a block's transaction transcripts. Each τ_i is the RWP proof object for a specific input; K is the number of included transactions (or inputs, depending on serialization). The **Merkle root** is computed over domain-separated leaf hashes (e.g., $H(\mathsf{TAG[LEAF]} \,\|\, \tau_i)$) and internal node hashes (e.g., $H(\mathsf{TAG[NODE]} \,\|\, L \,\|\, R)$), ensuring that neither a transcript nor a tree shape can be confused for a different object or level. By placing mrk_t in the header (via the accumulator A_t), the system binds the exact set and order of transcripts to the block forever.

A Merkle tree gives **logarithmic inclusion proofs**. To convince a light client that a particular τ_j was part of block t, a prover supplies the transcript bytes and a path of sibling hashes from the leaf up to the root. The light client recomputes the leaf hash, iteratively hashes with each sibling in the correct left/right position, and checks equality with mrk_t from the header. The proof size is $O(\log K)$, and verification is a handful of hash calls—perfectly matched to ENI6MA's symmetric-only philosophy.

Once a block has matured past a policy-defined depth Δ, nodes can **prune** the transaction bodies while retaining just $(A_t, \mathsf{com}_t, \mathsf{mrk}_t)$ and the committee signature σ_t. This collapses storage from "all transcripts forever" to "headers + roots," yet preserves full auditability: any party can later present a transcript

and its Merkle branch to re-establish inclusion. Because mrk_t and A_t are chain-hashed, pruning is provably safe—changing even one bit in history would cascade into mismatched accumulators.

Light clients thrive in this model. They track only headers—essentially the sequence of (A_t, σ_t) plus com_t and mrk_t—and, upon receiving a payment, request the minimal data needed: the relevant τ and a Merkle branch. They then verify the committee signature for the header, recompute the branch up to mrk_t, and finally run the RWP XOR checks locally. No full state, no trusted gateways, and no background sync are required to achieve strong assurances about a specific spend.

Pruning does not mean losing state. Full nodes maintain the **UTXO set** (or equivalent spendable state) as a compact key-value structure updated at each block. This set is all that's needed for ongoing validation; historical transcripts are only needed for audits or disputes. Nodes may periodically snapshot the UTXO set and discard intermediate history beyond Δ; new nodes can bootstrap by verifying headers and then ingesting a recent, checkpointed UTXO snapshot signed by the committee, or by replaying from genesis if they wish to trust nothing but code and math.

Archivists and regulated entities can operate **history warehouses** that store pruned transcripts off-chain. Because authenticity is anchored by mrk_t, any transcript pulled from an archive is either authentic (its inclusion proof verifies) or useless. This separation of concerns lets the base network stay lean while still supporting rich compliance and forensics workflows: subpoenas, dispute resolution, and academic analysis can proceed without burdening every validator with indefinite data retention.

Data-availability during the fresh window is ensured by propagation rules and **bounded pruning delay**. Producers must make full bodies available when the block is published; validators refuse to extend headers whose bodies they cannot fetch and verify. Only after Δ signed steps—when the probability of reorg is negligible—do nodes begin pruning. If the ecosystem requires stronger guarantees, erasure-coded sidecar blobs can be committed under a tagged root in aux_t, preserving the same pruning mechanics with verifiable retrieval.

The Merkle commitment is future-proof and **post-quantum friendly**. Its security reduces to the collision and second-preimage resistance of the underlying hash; quantum adversaries gain at most a square-root advantage (Grover's), absorbed by increasing digest size. Moreover, mrk_t composes cleanly with advanced structures like Merkle-Mountain Ranges (MMRs) for efficient append-only proofs, or with vector commitments if later upgrades want sub-logarithmic proofs for specific indices—all without changing the accumulator A_t or the RWP verifier.

By making pruning a first-class, consensus-aware behavior—anchored by mrk_t, carried in A_t, and guarded by σ_t—ENI6MA achieves the trifecta of scalability, verifiability, and longevity. Validators run fast with small disks; light clients verify like full nodes for the transactions they care about; and historians can reconstruct any piece of the past with cryptographic certainty.

---

8. Simplified Payment Verification (SPV)

A light client downloads the header chain $\{A_t\}$ and verifies the committee signatures. For a payment, it requests the Merkle branch proving inclusion of τ in mrk_t and verifies the RWP checks for that input (the checks are public, symmetric operations). Thus SPV requires **no full node** and achieves constant-time verification per input.

A light client follows only the signed headers, i.e., the sequence $\{A_t\}$ with their threshold signatures $\{\sigma_t\}$. Here, A_t is the block's accumulator digest (the one-way spine that commits to time, randomness, policy, and content), and σ_t is the beacon committee's threshold signature over A_t. By verifying each σ_t against the currently active committee verification keys (advertised in policy and rotated on-chain), the light client gains a succinct, trust-minimized view of time and ordering—without fetching full blocks. "Curly-brace A sub t is the header chain; sigma sub t is the committee signature over A sub t."

For a specific payment, the sender provides τ (the RWP transcript for the spent input), the block height t, and a Merkle branch path that connects τ to the block's transactions root mrk_t. The light client recomputes the leaf hash for τ, folds it up the branch, and checks equality with the mrk_t recorded in the corresponding header. This step answers "was this exact transcript included in block t?" with logarithmic work.

$$\mathsf{VerifyMerkle}(\tau, \mathsf{path}, \mathsf{mrk}_t) \to \mathsf{true}.$$

"Verify-Merkle of tau, path, and m-r-k sub t returns true."

After inclusion, the light client locally re-derives the context q (UTXO identifiers, amounts, recipient script, chain ID, etc.) from the transaction body and recomputes the designated probe set tied to q, the block seed s_t, and the block time T_t already embedded in the header's capsule commitment. It then runs the **public, symmetric** RWP checks—fixed-width XOR equalities—over τ. Because the verifier is symmetric-only and constant-time per probe, this step is computationally tiny and side-channel resistant even on mobile devices.

Acceptance at the input level uses the same predicate full nodes use; SPV does not weaken the rule.

$$\mathsf{ACCEPT} \iff \Lambda = 1 \ \wedge \ \mathsf{PolicyOK}(q).$$

"Accept if and only if Lambda equals one and Policy-OK of q is true."
Here, Λ is the logical AND of all probe-wise XOR checks for this input, and $\mathsf{PolicyOK}(q)$ applies any policy flags enforced at that height (jurisdiction tags, asset caps, timelocks, etc.), all of which are committed in A_t via pol_t.

To decide **how final** a payment is, the light client counts a small number Δ of consecutive, properly signed headers after height t. Because fork choice is "longest valid chain by signed accumulator steps," the probability of a reorg that dislodges the payment decays rapidly with Δ. The client can expose this as a simple meter—e.g., "2/5 confirmations"—without maintaining a mempool or running heavy reconciliation logic. The values involved are explicit: t (the

block height containing τ), Δ (the confirmations required), and $\sigma_{t+1}, \ldots, \sigma_{t+\Delta}$ (the successive committee signatures).

SPV's privacy posture is materially improved by RWP. Because τ contains **no reusable public key** or signature seed, correlating spends across time by key fingerprint is impossible. Light clients can further randomize who they query for path and body bytes (dandelion-like routing, randomized gateways) without trusting those relays: every byte returned is immediately checked against mrk_t and A_t.

Robustness comes from layered checks that a malicious server cannot fake. A dishonest full node cannot convince an SPV client that a non-included τ was included, because any invented branch fails the equality to mrk_t. It cannot convince the client to accept a header on a dead branch, because σ_t must verify under the current committee keys; nor can it make an invalid RWP transcript seem valid, because the XOR-membership predicates are recomputed locally from public data. The only remaining levers—network partition and eclipse—are mitigated by multi-peer header sampling and signature gossip.

The model is **post-quantum friendly** end-to-end. SPV relies on hash preimage and collision resistance (for mrk_t and A_t) and on the committee's threshold signature scheme; both can be instantiated with PQ algorithms (e.g., hash-based or lattice-based) without changing the SPV flow. If a PQ upgrade occurs, the only change a light client sees is the header's signature type and keys; all other values and checks remain identical.

Finally, the economic footprint is minimal and predictable. A wallet only fetches headers and, on demand, a handful of body bytes for the payments it cares about. No archival duties, no perpetual sync. Yet the assurances match full-node logic for that payment because the client verifies the exact same predicates over τ, mrk_t, and A_t. This alignment—**full security for local facts, with tiny bandwidth**—is the core strength of SPV in an RWP ledger.

---

9. Combining and Splitting Value

Transactions support multiple inputs and outputs. Standard coin-selection consolidates dust; change outputs return to the sender's script. Fan-out is not problematic because validity is local to the included inputs' τ and the Merkle proof of inclusion.

A transaction may consume multiple inputs and produce multiple outputs; the ledger enforces value conservation with a simple invariant. Let $\{\mathrm{in}_j\}_{j=1}^{m}$ be input amounts and $\{\mathrm{out}_k\}_{k=1}^{n}$ be output amounts, then

$$\sum_{j=1}^{m} \mathrm{in}_j \;=\; \sum_{k=1}^{n} \mathrm{out}_k \;+\; \mathrm{fee}.$$

"Sum of inputs equals sum of outputs plus fee."
Every input carries its own RWP transcript τ_j proving authorization for this

transaction's context q_j, which binds the referenced UTXO, amounts, recipient script(s), and chain identifiers. Because each τ_j is time- and context-bound, the act of combining or splitting value introduces no new attack surface: each input stands or falls on its own XOR-membership checks.

At the transaction level, acceptance composes conjunctively across inputs and policy.

$$\mathsf{ACCEPT}_{\mathrm{tx}} \iff \bigwedge_{j=1}^{m} \left(\Lambda_j = 1 \wedge \mathsf{PolicyOK}(q_j)\right) \wedge \mathsf{ConserveVal}.$$

"Accept transaction if and only if, for every j from one to m, Lambda sub j equals one and Policy-OK of q sub j is true, and the conserve-value rule holds." Here, Λ_j is the probe-wise AND for input j, and $\mathsf{ConserveVal}$ encodes the sum equation above. This structure makes **fan-in** (many inputs) and **fan-out** (many outputs) straightforward: validation work scales linearly, and no cross-input algebra beyond value conservation is required.

Standard coin-selection strategies (largest-first, knapsack, branch-and-bound) can be used to consolidate dust into fewer, larger outputs—lowering future fees and shrinking the UTXO set. RWP strengthens privacy in this process because there is no persistent public key that links the consolidated inputs; each authorization is a fresh, per-context transcript. Policy can set a dust threshold $d_{\min}$ to ban outputs below an economic minimum, defending against UTXO set bloat. "d sub min is the minimum dust value allowed for an output."

Change handling remains familiar: any excess input value becomes a change output to a fresh script controlled by the sender. With RWP, that script can require future spenders to present an RWP transcript satisfying particular policy tags (e.g., device attestation class, jurisdiction). Since τ is non-reusable, address rotation is natural and correlation-resistant; wallets can default to one-time scripts without key-management pain.

Complex payouts—salary batches, airdrops, channel openings—benefit from **deterministic ordering** of outputs and inputs under a domain-separated tag (e.g., sort by $H(\mathsf{TAG[SORT]} \,\|\, \text{entry})$). Determinism minimizes accidental forks if a block producer reconstructs the same transaction from mempool fragments and ensures that txh_t and mrk_t are stable under re-serialization. The values that matter here are the serialization rules and the tag used for sorting; both are consensus parameters recorded in policy.

From a DoS and resource perspective, policy caps on m and n (inputs and outputs per transaction) prevent pathological fan-outs. Because RWP verification is cheap, the main scarce resources are bandwidth and Merkle-path depth. Caps are published in pol_t so every node enforces the same limits; fee-per-weight ensures very large, but still legal, transactions pay commensurately for the block space they occupy.

Multi-party and covenant-style constructions compose well with RWP. For example, a payout can require **threshold RWP** where k of r distinct devices each supply a τ bound to the *same* transaction context q. The acceptance predicate simply accumulates a larger conjunction: all k chosen inputs must

satisfy $\Lambda = 1$. Covenants can constrain descendants (e.g., "must pay to a script with tag X") by embedding rules into q and $\mathsf{PolicyOK}$, which are both committed in the header chain via A_t. No new signature scheme or opcodes are needed; the ledger's existing variables—τ, q, A_t, and pol_t—carry the semantics.

Fee-bumping and replacement policies (e.g., Replace-By-Fee) remain available with clear, consensus-visible rules. A replacement transaction must reference the same inputs, offer a higher fee, and pass the same RWP checks for each q_j. Because τ_j binds tightly to the transaction's amounts and outputs, an attacker cannot "tweak" a victim's transaction into paying a different destination while preserving validity: any change to q_j forces a completely fresh τ_j, which only the rightful prover can produce within the window.

Finally, the compositionality of combining and splitting is friendly to light clients. SPV needs only the Merkle branches for the specific inputs the client cares about, plus the accumulator headers that make the chain canonical. Whether a payment arrived through a single-output send or a complex batched transaction is immaterial: the client's local checks over τ_j, mrk_t, and A_t are identical. This uniformity—**one simple acceptance logic everywhere**—keeps the implementation surface small while scaling to complex, real-world payment flows.

---

10. Privacy

Unlike bank rails that centralize linkage, ENI6MA preserves privacy by:

- **Pseudonymous scripts** per transaction, rotating addresses by default.
- **Ephemeral witnesses**: no static signature key appears on chain.
- **Short-lived transcripts**: outside their time window, they are useless for correlation.

Optional mixers or payment codes can further obscure linkages while remaining compatible with RWP verification.

ENI6MA's privacy model starts from the observation that most leakage on conventional rails comes from durable identifiers—account numbers, long-lived public keys, or static device fingerprints—being reused across time. In the RWP ledger, a spend is authorized by a **transcript** τ that proves knowledge for a single event only, and never exposes a reusable signing key. The variables at play are τ (the proof artifact), x (fresh session entropy), T (the block/window time), and q (the public transaction context: referenced UTXOs, amounts, recipient script, chain/network ID). The binding relation "τ is valid for (x, T, q)" eliminates a global handle; in "tau binds to x, T, and q for this spend alone."

Pseudonymous scripts ensure that recipients don't publish an enduring address. Each output's locking script is a one-off predicate that will later require an RWP transcript matching its template. Because there is no long-lived public key inside the script, wallets safely **rotate by default**—each payment request

generates a fresh script hash, and each future spend will produce a fresh τ. The value pol_t (policy at time t) can even require rotation by consensus, preventing accidental reuse that could aid linkage.

Ephemerality of the witness flows from the sealed morphism $\mathcal{M}$ and the function $W = f_{\mathcal{M}}(x, T, q)$: the witness W never leaves the device and never repeats, and τ reveals only masked responses that cancel under verifier-side XOR checks. An observer who sees τ and later sees another τ' gains no statistical purchase to link them unless the public contexts q themselves correlate (e.g., identical amounts and change patterns). Even then, the privacy risk is **contextual**, not cryptographic; users can mitigate it with coin-selection and output-amount shaping policies that are visible in pol_t.

Short-lived transcripts blunt network-level surveillance. A gossip adversary who records τ at time T cannot replay it, and therefore can't "probe" the network later to find where it is accepted; any such probes will fail the recomputed probe set derived from s_T (the block seed) and T. The values s_T and T are committed in the header via com_T, making the verifier's challenge set common knowledge after the fact but useless for linkage going forward.

Because SPV verification is symmetric-only, light clients can retrieve data **anonymously** from multiple peers and verify locally. A wallet asks for the Merkle branch for a candidate τ under mrk_t and checks inclusion against the accumulator A_t. No trusted indexer is required, and the wallet never needs to reveal which branch of the tree it ultimately cares about. The variables mrk_t (the Merkle root for block t) and A_t (the block's accumulator) are the only anchors it needs, both broadcast to everyone.

Optional mixers and **payment codes** plug in without modifying the RWP verifier. Mixers are simply transactions whose outputs are deliberately uniform in value and script template; because τ is one-time and scripts are pseudonymous, a well-designed mixer reduces the correlation of inputs to outputs with no custom cryptography. Payment codes (e.g., derivation schemes that let a recipient publish a single code yet receive to unlinkable scripts) are implemented at the script-derivation layer and show up in chain data only as ordinary one-time scripts, still spending with fresh τ.

Change outputs are a persistent source of heuristic linkage in UTXO systems. ENI6MA counters by encouraging wallets to send **deterministically indistinguishable** change: randomize exact amounts within small ranges, rotate scripts, and, when feasible, aggregate change with later payments through coin selection. Since authorization never reuses a public key, even change that returns to the same owner arrives under a **fresh predicate**, reducing the success rate of change heuristics that rely on key reuse or script templating.

Finally, compliance-oriented privacy is enabled rather than obstructed. If a user opts into selective disclosure, a wallet can reveal a minimal tuple—(t, τ, path)—proving receipt or payment inclusion under mrk_t and A_t, without revealing anything about unrelated activity. Because τ cannot be repurposed and path is a local Merkle proof, the verifier learns only "this one transfer happened." This one-fact proof pattern is a direct consequence of encoding authorization via ephemeral knowledge rather than durable identity.

11. Security Calculations

11.1 Single-Window Forgery

Assume the XOR-membership test uses h probes with per-probe false-pass probability 2^{-k} under random guessing. Then:

$$\Pr[\Lambda = 1 \mid \text{guess}] \;=\; 2^{-kh}.$$

"The probability that Lambda equals one given guessing equals two to the power of minus k times h."

Selecting $k = 8$ and $h = 8$ yields 2^{-64} per input; combined with hash binding and policy checks, the overall bound is dominated by the negligible term $\mathsf{negl}(\lambda)$ from breaking the transcript construction.

11.2 Double-Spend Race

Let Δ be the **finality window** in blocks (time windows). An attacker who issues two conflicting transactions must create two valid transcripts τ, τ' for disjoint contexts q, q' over distinct windows (or sub-windows). Without access to $\mathcal{M}$ and live entropy, the best strategy is **precomputation or replay**, both of which fail by design. The residual risk is committee capture within Δ:

$$\Pr[\text{reorg} \geq \Delta] \;\leq\; \sum_{j=t}^{n} \binom{n}{j} \varepsilon^j (1-\varepsilon)^{n-j},$$

"The probability of a reorganization at least Delta is at most the sum for j from t to n of binomial n choose j times epsilon to the j times one minus epsilon to the n minus j,"

where ε is the per-member compromise probability and t the threshold. Parameterizing n and t yields exponentially small reorg odds.

11.3 Liveness

Because blocks follow wall-clock pacing, expected confirmation time is linear in Δ. With a 1-second block window and $\Delta = 5$, typical practical finality is ~ 5 seconds, absent network partitions.

12. ENI6MA-G: A Bank-Grade Stablecoin (1 g Gold per Token)

ENI6MA-G is a fiat-onramp-friendly stable instrument defined as **one token per gram of vaulted gold** held by qualified custodians.

Issuance & Redemption.
Custodians mint/burn on customer instruction. Each block window, custodians publish a **reserve attestation capsule**:

$$\mathsf{res}_t \;=\; H(\mathsf{TAG[RES]} \,\|\, \mathsf{SKU} \,\|\, \mathsf{mass}_t \,\|\, \mathsf{vaultID} \,\|\, \mathsf{audit\_ref}),$$

"res sub t equals hash of tag R-E-S concatenated with S-K-U, concatenated with mass sub t, concatenated with vault I-D, concatenated with audit reference."

and an RWP transcript τ_t^{res} showing the attestor's live control over the data source(s) at time T_t. The capsule is included in txh_t and bound into A_t. Anyone can verify that total circulating supply $\leq$ attested reserves (within tolerance).

Why banks can adopt.

- No reusable HSM keys to leak; **no private keys at rest**.
- Public, **continuous reserve attestations** with cryptographic binding.
- Symmetric-only verification (hash/XOR/threshold-sig), **post-quantum friendly**.
- Fine-grained compliance policies embedded via pol_t (jurisdictions, KYC tags, etc.).

12. ENI6MA-G: A Bank-Grade Stablecoin (1 g Gold per Token)

Overview and monetary model
ENI6MA-G defines the **unit of account** as exactly one (1) gram of vaulted, good-delivery gold per token, with redemption rights administered by qualified custodians under published policies. The monetary base is therefore not an algorithmic promise but a **physically collateralized ledger state** whose adequacy can be verified by anyone through reserve capsules committed every block window. Let $G = 1$ gram/token be the conversion constant; this "gram code" functions as the **stable code**—a neutral, apolitical base unit that maps deterministically into any currency via public gold prices. Because the ledger's correctness depends only on symmetric verification and capsule arithmetic (not on proprietary price oracles), users can price, hedge, and settle across currencies without trusting opaque keyholders or energy-based lotteries.

Cross-currency valuation and settlement
The gram code establishes a canonical bridge between token quantity and any fiat currency $c \in \{\mathrm{USD}, \mathrm{EUR}, \mathrm{JPY}, \ldots\}$. Let $P_c(t)$ denote the price of **one gram** of gold in currency c at time t (sourced from any regulated venue or benchmark). Then the fair value $V_c(t)$ of q tokens is

$$V_c(t) \;=\; q \cdot G \cdot P_c(t).$$

"V sub c of t equals q times G times P sub c of t."
Because G is constant and public, bilateral FX flows reduce to **two observable prices**: the gram price in the source currency and the gram price in the destination currency. Routing payments through grams avoids triangular arbitrage complexity and makes bank treasuries' VaR transparent—no secret pegs, only metal.

Programmatic FX fences
Banks may prefer to smooth intraday volatility. Define a time-weighted benchmark $\overline{P}_c$ over a policy window Θ (e.g., 30 minutes):

$$\overline{P}_c \;=\; \frac{1}{|\Theta|} \sum_{t \in \Theta} P_c(t).$$

"P bar sub c equals one over the size of theta times the sum over t in theta of P sub c of t."
Using $\overline{P}_c$ to quote retail conversions while settling interbank at real-time P_c yields predictable customer experiences without hiding basis risk; all parameters sit in the on-chain policy word pol_t, so auditors can replay decisions.

Risk partitions and bankruptcy remoteness
Reserves are held in segregated, bankruptcy-remote trusts. The ledger records **which vault and SKU** backs circulating supply; redemption flows are routed by $\mathsf{vaultID}$ and SKU, so a custodian-level issue cannot contaminate the global peg. Banks can hold multiple SKUs (e.g., 400 oz bars vs. 1 kg bars), with on-chain conversions executed by the custodian only when bar lotting requires it. The ledger's invariants do not change: tokens map to grams; grams map to SKUs; SKUs map to vault balances.

Operational clarity for banks
From a core-banking perspective, ENI6MA-G behaves like a **real-time, bearer certificate of deposit** with metal backing and instantaneous, final settlement. Customer KYC/AML status rides in tag bits inside pol_t; transfer predicates require RWP-PASS **and** compliant tags, ensuring that bank policy is enforced at spend time without revealing identity on chain. Treasury sees a one-line balance per legal entity; reconciliation reduces to header checks and reserve capsule diffs; privacy is preserved because no long-lived signing keys ever appear.

Fraud surfaces retired by construction
There are **no private keys at rest** to exfiltrate from HSMs, no static issuer signatures to forge, and no miner liveness assumptions. Every authorization is an ephemeral RWP transcript, and every reserve disclosure is a committed capsule under the signed accumulator A_t. Attackers cannot "practice" forgeries in dark pools because transcripts expire with the window and probes change with the seed. The bank's security operations center measures three things—not a thousand: code identity, entropy freshness, and committee signature validity.

Regulatory auditability
All economically relevant promises are machine-verifiable from headers: circulating supply, reserve mass per vault, SKU composition, policy version, and signature quorum. External auditors can replay from genesis using the pub-

lic code and **no secrets**, recomputing the same accept/reject decisions that honest nodes did. This produces defensible SOC-1/SOC-2 evidence, shortens audits, and allows regulators to build independent monitors that watch for drift between supply and reserve capsules.

Issuance & Redemption and the reserve capsule

$$\mathsf{res}_t \;=\; H(\mathsf{TAG[RES]} \,\|\, \mathsf{SKU} \,\|\, \mathsf{mass}_t \,\|\, \mathsf{vaultID} \,\|\, \mathsf{audit\_ref})$$

"res sub t equals hash of tag R-E-S concatenated with S-K-U, concatenated with mass sub t, concatenated with vault I-D, concatenated with audit reference."

The **reserve capsule** res_t is a tamper-evident digest of a custodian's state at window t. The symbol SKU identifies the class of metal (e.g., 995/1000 gold) and its lotting format; mass_t is the gram-accurate total mass under that SKU currently unencumbered; $\mathsf{vaultID}$ identifies the storage location; and $\mathsf{audit\_ref}$ binds an external attestation artifact (report hash, timestamp, or chain of custody). Domain separation via $\mathsf{TAG[RES]}$ prevents cross-component collisions. Any post-hoc attempt to revise inventory or point the same digest at different documents breaks second-preimage resistance and is immediately detectable.

The custodian must also prove **live control** of the data source at T_t. They publish an RWP transcript τ_t^{res} bound to $(x, T_t, q_{\mathrm{res}})$, where q_{res} includes the human-readable disclosure (SKU, mass, vault, audit ref) and the current accumulator A_{t-1}. Validators verify τ_t^{res} exactly like a payment transcript—symmetric XOR checks only—so no special issuer keys or custom logic exist. If the custodian's device identity drifts (attestation hash changes), τ_t^{res} fails; the bank ops team sees it in the very next window.

Once res_t and τ_t^{res} are included, they are pulled into the block's content hash txh_t and then into the accumulator A_t. This yields a public inequality that anyone can evaluate from headers alone. Let Supply_t be total tokens outstanding for the SKU family at height t, and let Mass_t be the sum of mass_t across all custodians after policy haircuts h (e.g., insurance deductibles, transport buffers). The safety condition is

$$\mathsf{Supply}_t \;\le\; \frac{\mathsf{Mass}_t}{G} \cdot (1 - h).$$

"Supply sub t is less than or equal to Mass sub t divided by G times one minus h."

Violation is objective: either the inequality holds or it does not. No FOIA requests, no phone calls.

Mint and burn discipline

Minting requires a **preimage** of future reserve: a custodian (or consortium) first posts res_t reflecting incoming metal; only then may they mint tokens up to the allowed slack. Burning mirrors the process: tokens are destroyed on chain; the next res_{t+1} reflects the newly unencumbered mass. This order prevents "mint

now, promise later" drift and turns redemption into a two-line ledger diff visible to counterparties and regulators without revealing PII.

Tolerance bands and SKU rotation

Physical logistics induce small timing mismatches. Policy encodes a tolerance band ϵ and SKU rotation rules:

$$\left| \mathsf{Supply}_t - \frac{\mathsf{Mass}_t}{G} \right| \leq \epsilon.$$

"Absolute value of Supply sub t minus Mass sub t over G is less than or equal to epsilon."

Breaches outside ϵ trigger rate-limited mint/burn locks and on-chain incident tags, not ad-hoc emails. SKU rotation (e.g., melting and recasting) is expressed as a zero-sum move between SKUs within the **same** $\mathsf{vaultID}$, leaving Mass_t and inequality checks intact.

Redemption workflows

Retail redemption can be physical (deliver metal) or financial (wire in fiat). The transaction predicate checks RWP-PASS plus KYC tag bits; upon redemption, the custodian burns the customer's tokens and issues a redeem receipt whose hash is included in the next res_t via $\mathsf{audit\_ref}$. Institutional redemption can be bar-for-bar: tokens map to grams; grams map to bar counts rounded by SKU; any remainder stays as token change. Every step is **cryptographically linked** to headers, so disputes are resolvable with a single Merkle proof and a receipt.

Multi-custodian aggregation

Banks rarely rely on a single venue. The ledger naturally aggregates res_t across many custodians and vaults; Mass_t is a sum, not a statement of faith. Diversity reduces idiosyncratic risk and allows **jurisdictional matching**—a European bank can accept only EU vault IDs; an Asian bank can require APAC vaults—without forking the asset or changing code paths. Everything is visible in policy and headers.

Why banks can adopt (security & compliance deep dive)

No private keys at rest

Traditional tokenization leans on long-lived issuer keys inside HSMs—attractive, breachable targets. ENI6MA-G uses **ephemeral RWP witnesses** for both payments and disclosures. The only privileged object is the morphism $\mathcal{M}$, i.e., code identity verified via measured boot and remote attestation. Even if a device is seized, there is no master signing key to extract; past transcripts remain unforgeable; future ones require fresh entropy and the correct code hash. This collapses legal exposure around key custody and simplifies SOC controls.

Continuous reserve attestations

Reserves are not quarterly PDFs; they are **per-window capsules**. Each res_t is hashed, tagged, and chained under the accumulator A_t signed by the committee. External audit reports and inventory movements are linked by $\mathsf{audit\_ref}$; their hashes become part of the public tape. A regulator can subscribe to headers,

compute the inequalities in real time, and set automated alerts for drift or missing capsules—no privileged API required.

Symmetric-only, post-quantum friendly verification
All hot-path checks are hashes and XORs; the only asymmetric element is the committee's threshold signature over A_t, which is **swappable** for lattice- or hash-based PQ schemes without changing block structure. Banks gain long-horizon cryptographic assurances and avoid retooling every time standardization bodies update PQ recommendations. Parameter raises (e.g., longer digests) are easy and low-risk.

Fine-grained compliance in pol_t
Jurisdiction, asset class, risk ratings, travel-rule hints—these live as **policy tags** in pol_t that spending predicates must satisfy alongside RWP-PASS. Because pol_t is hashed into A_t, enforcement is **objective** and replayable: a transfer either satisfied the policy bits that were active at height t or it didn't. Banks can update allow-lists, add sanctions designators, or raise KYC levels with explicit activation heights; light clients can still verify with headers only.

Bank-grade fault domains and segregation
Custodian disclosure is segregated by $\mathsf{vaultID}$ and SKU; mint/burn windows are rate-limited; and tolerance ϵ is enforced on chain. These variables make operational risk **quantized** and analyzable: a single vault's error cannot invalidate global solvency, and breaches are machine-flagged with timestamps. Treasury functions can attach operational KPIs (e.g., maximum drift minutes, SLA on capsule timeliness) to $\mathsf{vaultID}$ and compare custodians on public data.

Cross-currency rails without reinventing FX
Because the stable code is the gram, the bank's FX desk simply manages $P_c(t)$ curves it already understands. Retail apps can show balances in local currency while keeping settlement in grams; corporate treasuries can hedge with standard gold forwards. Nothing in the ledger asks banks to trust a new oracle cartel; **any** legally acceptable price source can be used, and discrepancies resolve via arbitrage and redemption, not by governance edict.

Operational simplicity and audit trail
Daily ops reduce to three verifications: header signatures σ_t, reserve capsules res_t vs. supply, and local policy compliance at spend time. Every event leaves a cryptographic breadcrumb retraceable with (A_t, mrk_t) and a single Merkle branch. Investigations don't require subpoenaing private logs first; the public tape narrows scope to specific windows and capsules before private records are even consulted.

Customer protection and recourse
If a custodian is late or equivocal, **rate-limiters** and **automatic locks** in pol_t can freeze mint/burn capabilities above de-risked thresholds until valid capsules resume, without halting peer-to-peer transfers. Consumers remain liquid; institutions see an explicit state flag on headers; and remediation steps are written into policy rather than improvised, reducing legal ambiguity.

Additional equations (valuation, solvency, conversions)

Valuation across currencies

$$V_c(t) \;=\; q \cdot G \cdot P_c(t), \qquad G = 1 \text{ gram/token}.$$

"V sub c of t equals q times G times P sub c of t; G equals one gram per token."

Aggregate solvency across custodians

Let $\mathcal{C}$ be the set of custodians, each with capsule mass $\mathsf{mass}_t^{(j)}$ and haircut $h^{(j)}$.

$$\mathsf{Supply}_t \;\le\; \sum_{j \in \mathcal{C}} \frac{\mathsf{mass}_t^{(j)}}{G} \cdot \big(1 - h^{(j)}\big).$$

"Supply sub t is less than or equal to the sum over j of mass superscript j sub t divided by G times one minus h superscript j."

Tolerance band policy

$$\left| \mathsf{Supply}_t - \frac{\mathsf{Mass}_t}{G} \right| \;\le\; \epsilon, \quad \mathsf{Mass}_t = \sum_j \mathsf{mass}_t^{(j)}.$$

"Absolute value of Supply sub t minus Mass sub t over G is less than or equal to epsilon; Mass sub t equals the sum over j of mass superscript j sub t."

Attestation binding

$$\mathsf{res}_t \;=\; H(\mathsf{TAG[RES]} \,\|\, \mathsf{SKU} \,\|\, \mathsf{mass}_t \,\|\, \mathsf{vaultID} \,\|\, \mathsf{audit\_ref}), \quad \tau_t^{\text{res}} = \mathsf{Transcribe}(f_{\mathcal{M}}(x, T_t, q_{\text{res}}),\, q_{\text{res}}).$$

"res sub t equals the hash of tag R-E-S, S-K-U, mass sub t, vault I-D, audit reference; tau superscript res sub t equals Transcribe of f sub M of x, T sub t, q sub res with q sub res."

Conversion routing between currencies $a \to b$

$$\text{Pay}_b \;=\; \text{Recv}_a \cdot \frac{P_b(t)}{P_a(t)}.$$

"Pay sub b equals Receive sub a times P sub b of t divided by P sub a of t."

These identities—and their explicit, forms—are not "advice"; they are mechanical relations that any participant or regulator can recompute from public data, ensuring that ENI6MA-G's peg, solvency, and FX conversions are **transparent, verifiable, and bank-operable**.

---

13. Discussion: Threats and Posture

- **Relay/Replays:** Fail outside the time window; τ cannot be replayed because it binds to (x, T, q).

- **Precomputation:** Entropy x is fresh and unpredictable; $\mathcal{M}$ is sealed.
- **Committee Capture:** Mitigated by threshold schemes, diverse operators, and on-chain rotation with slashing.
- **Quantum Adversaries:** Hash and XOR checks remain robust; committee signatures can use lattice or hash-based PQC.
- **DoS:** Fees price scarce resources; non-verifying floods are dropped before inclusion.

Relay/replay threats are neutralized by the binding $\tau \leftrightarrow (x, T, q)$. A captured transcript is only valid for the unique triple it was minted against; any attempt to relay it later (different T) or elsewhere (different q) fails when validators recompute the probe set from (T, s_T, q) and find that the XOR equalities do not cancel. In "tau is accepted only if x, T, and q match the original; otherwise, the probe set changes and tau fails." This property turns the network's own clock and randomness into a built-in replay firewall without tracking nonces or keeping per-identity state.

Precomputation is blocked by **fresh entropy** x and the sealed morphism $\mathcal{M}$. Even if an adversary could predict q (say, an invoice amount and recipient script), they cannot predict the private orientation $W = f_{\mathcal{M}}(x, T, q)$ because x is sampled at session start and $\mathcal{M}$ is not extractable. Attempts to build rainbow tables over plausible q values or over future seeds s_t are futile: the combination with x and the device-bound code identity produces a one-time witness that does not collide across events. The only viable path would be to compromise the code identity $\mathcal{M}$ itself—addressed by measured boot, attestation, and self-checksums—and even then, the damage is time-limited because past transcripts do not help forge future ones.

Committee capture is modeled explicitly via threshold parameters (n, t) and a compromise probability ε. The header's signature σ_t is valid only if at least t out of n independent operators participate honestly to sign the accumulator A_t. Mitigations stack: diversify operators across jurisdictions and infrastructures; rotate keys and membership on-chain; require public commitments for randomness beacons; and enforce **slashing** for equivocation or absenteeism encoded in pol_t. In risk terms, the residual chance that a captured committee can backdate or bless a conflicting history within an application's finality window Δ is bounded by a tail term that decays with operator diversity and increases with Δ; systems that need stronger guarantees simply raise Δ or the threshold t.

Quantum adversaries change the calculus for asymmetric cryptography, but RWP's verifier is **symmetric-only**. The hash $H(\cdot)$ that underpins com_t, A_t, and mrk_t remains secure against quantum attack with a modest increase in digest size (to offset Grover's square-root speedup). The XOR-membership checks retain their hardness as long as masks and probes derive from pseudorandom functions instantiated over post-quantum-safe hashes. The only component that must be upgraded proactively is the committee's threshold signature; the design allows swapping to lattice- or hash-based schemes without touching block structure or RWP validation, because σ_t signs a single value A_t either way.

Denial-of-service (DoS) pressures are priced rather than wished away. Every transaction must pay a **fee-per-weight** where weight is a consensus-defined function of bytes on the wire and the number of per-input probes verified. Because RWP verification is constant-time and small, the fee schedule maps closely to the real resource under contention: bandwidth and block space. Non-verifying floods—packets that fail structural prechecks or domain tags—are dropped at the edge before they consume validation cycles; nodes record short-lived "seen-bad" notes to avoid re-requesting the same junk. Rate limits by peer and per-subnet prevent a single network vantage point from saturating a victim's view.

Network-level threats like eclipse and partition are mitigated by **multi-homing** header sampling and the fork rule "longest valid chain by signed accumulator steps." Nodes maintain diverse peer sets and refuse to accept unbroken sequences of headers sourced from a single neighbor without cross-checks; headers carry compact signatures σ_t that can be validated in isolation. During partitions, both sides may extend valid local chains; when connectivity returns, the branch with more consecutive signed steps wins. Applications can adapt by increasing the confirmation depth Δ during suspected partitions, a value that is explicit and easy to reason about.

Side-channel and malleability classes are narrowed by design. The verifier for τ executes fixed-width XORs with constant-time equality checks, avoiding early exits and data-dependent branching. There is no signature malleability: any bit flip in τ breaks at least one predicate, and hence the transaction ID is stable once formed. Scripts are pseudonymous and minimal, reducing the surface for covert channels embedded in exotic opcode sequences; if richer programmability is desired, it is gated behind policy flags in pol_t so that all nodes agree on which features exist at which heights.

Operational posture recognizes that **entropy quality** is security-critical. The network continuously monitors beacon health—e.g., checking unpredictability of s_t across windows and requiring commit-reveal proofs from committee members, with slashing for bias or failure. Endpoints run self-tests on local x sources and fall back to mixed sources (device RNG plus beacon-derived salt) to keep the effective min-entropy high. If measurements detect degradation, policy can respond automatically: raise probe counts, shrink per-window limits, or temporarily increase Δ until randomness quality is restored, all recorded on-chain via pol_t.

Lastly, the posture is **auditable**. Every assumption that matters—committee keys and rotation epochs, fee schedules, randomness beacons, pruning horizons, finality depth recommendations—is either committed in headers $(A_t, \mathsf{com}_t, \mathsf{mrk}_t)$ or encoded in on-chain policy pol_t. External reviewers can replay history from genesis with only public code and these values, recomputing RWP checks and fork choices exactly as honest nodes did. That transparency is the ultimate hedge: even if a new threat emerges, the community can measure its impact precisely and adjust parameters within the same, small, verifiable envelope.

---

14. Conclusion

We propose a peer-to-peer electronic cash and settlement system that **replaces PoW with Proof-of-Knowledge** anchored in time-keyed entropy and sealed symmetric morphisms. Transactions are validated by **ephemeral witnesses** and aggregated in a signed **accumulator chain**; double-spending is thwarted without requiring energy-intensive mining. The design preserves pseudonymity, enables light-client verification, and offers bank-grade reserve attestations for a **1 g gold-backed** stablecoin. In short, ENI6MA delivers the benefits of trust-minimized digital cash with **lower cost, faster finality, and stronger operational privacy**, and without burning watts for security.

ENI6MA replaces energy-dominated consensus with **Proof-of-Knowledge (PoK)** anchored in time-keyed entropy and sealed, symmetric morphisms. In practice, every spend is authorized by an **ephemeral witness**—a per-event statement that never becomes a reusable secret—while blocks are chained by a compact, threshold-signed **accumulator** that binds time, randomness, policy, and content. This yields a ledger where correctness is checked with hashes and XORs, not megawatts and nonces, and where **security budgets are explicit** in parameters rather than implied by fluctuating hash markets.

The architecture separates cryptographic soundness from governance via **policy words** embedded in headers. Cryptography answers "is this spend correct for this time and context?"; policy answers "is this spend permitted under current rules?" This clean factoring lets regulators, custodians, and networks evolve operational rules without disturbing the proof system or re-keying the world, preserving both stability and upgrade velocity.

Because verification is **symmetric-only and constant-time** per input, light clients achieve full-node assurance for the payments they care about with tiny bandwidth: a header chain, a Merkle branch, and a local run of the RWP checks. That makes **secure mobile and embedded experiences** a default, not an afterthought, and broadens participation by eliminating heavyweight node requirements and centralized RPC trust.

Privacy derives from **non-reusability**. No long-lived signature key ever appears on chain; scripts are pseudonymous and rotate by default; transcripts expire outside their windows. The result is operational privacy that resists common chain-analysis heuristics without relying on opaque mixers or proprietary cryptography, while still enabling **selective disclosure** with single-fact proofs anchored in public headers.

The system's threat model is **transparent and tunable**. Where Bitcoin assumes an honest majority of CPU power, ENI6MA assumes an honest threshold of beacon committee members and the freshness of entropy. Both assumptions are auditable on chain (signing participation, randomness proofs), and both have **direct mitigations**: rotation, diversity, slashing, and parameter shifts (confirmation depth, probe counts) that tighten guarantees without redesign.

By committing **reserve capsules** each window, ENI6MA-G turns solvency into a **machine-checkable inequality** rather than a PDF promise. Anyone can recompute circulating supply against committed mass and policy hair-

cuts from headers alone. This creates a credible, **bank-operable stablecoin**—priced in grams, settled in seconds, and reconciled deterministically—without hidden oracles or key-custody cliffs.

Economically, incentives align with real costs: bandwidth, storage, and availability are paid by **simple, predictable fees**; the beacon committee earns a small, regulated service fee for producing unbiased seeds and signed accumulators; and custodians can **rebate** fees when they publish timely reserve capsules. No energy cost externality is required for security, and no advantage accrues to specialized hardware—lowering barriers and improving geographical fairness.

In short, ENI6MA delivers trust-minimized digital cash with **lower cost, faster finality, and stronger operational privacy**, while enabling a **bank-grade, 1-gram-backed stablecoin** whose solvency is public, continuous, and cryptographically bound. It builds on well-understood primitives, keeps the verification surface compact and post-quantum friendly, and makes both payments and disclosures verifiable by **anyone, anywhere, from headers alone**.

Contributions of ENI6MA Stablecoin E6G to the Industry

E6G introduces the **gram code**—one token equals **one gram of vaulted gold**—as a neutral, apolitical unit of account that maps deterministically into any fiat via public gold prices. This removes ambiguity around pegs, replaces opaque "stability mechanisms" with physical collateral, and gives banks and treasuries a **universal conversion rail** that does not depend on a single sovereign currency or a cartelized oracle.

E6G operationalizes **continuous solvency**. Each window, custodians publish a **reserve capsule** (SKU, mass, vault ID, audit reference) plus an RWP transcript proving live control of sources. The capsule is committed into the block's content and accumulator, so supply-versus-reserves can be checked in real time by wallets, auditors, and supervisors. Industry pain points—quarterly lag, spreadsheet drift, and unverifiable claims—are replaced by a **public, header-bound tape**.

E6G eliminates **private keys at rest** for issuance and disclosure. Instead of HSM-guarded issuer keys that can leak or be coerced, mint/burn and attestations are authorized by **ephemeral witnesses** bound to code identity and time. Compromise does not cascade across epochs; past transcripts do not help forge future ones; and legal/compliance exposure around key custody shrinks dramatically.

E6G embeds **compliance as code** via on-chain policy words: jurisdiction tags, KYC levels, allow/deny lists, travel-rule hints, fee schedules, mint/burn tolerances, and committee membership/thresholds. Because policy sits under the same accumulator and signatures as transactions, enforcement is **objective and replayable**: a transfer either satisfied the active policy at height t or it didn't, and any reviewer can confirm that outcome independently.

E6G provides **seconds-class settlement** with deterministic fork choice ("longest valid chain by signed accumulators"), enabling card-like user experiences without probabilistic six-block waits. Finality depth is an explicit param-

eter that businesses can right-size by risk tier; consumer payments can clear in a few windows, while large treasury moves can wait deeper—**one mechanism, tunable assurances**.

E6G upgrades **SPV to first-class**. A smartphone can verify a received payment or a reserve claim with only headers and a Merkle path, performing the very same XOR/hash checks as a full node. This collapses the trust surface around hosted nodes and proprietary indexers and enables **consumer-grade self-custody** without sacrificing safety or compliance.

E6G offers **programmatic FX** across currencies by routing through grams. Pricing, hedging, and settlement become simple transformations on public data (price of a gram in each currency), allowing banks to integrate E6G into existing risk and treasury systems without inventing new valuation frameworks. Customer apps can present local-currency balances while settling in grams under the hood.

E6G advances **sustainability and fairness**. With no PoW and no stake-rent feedback loops, there is no incentive to centralize around energy or capital monopolies. Security flows from fresh randomness, code identity, and diverse committees, which are **auditable and rotate by rule**. The cost drivers—bandwidth and availability—are paid by users who consume them, rather than by society through external energy burn.

E6G streamlines **audit and supervision**. Supervisors can follow headers, verify committee signatures, read policy, and evaluate solvency inequalities in near real time—no privileged APIs, NDAs, or human-driven reconciliations. This **self-auditing** posture shortens regulatory cycles, reduces uncertainty, and raises the bar for all tokenized money: if it isn't public and replayable, why trust it?

E6G preserves **user privacy with accountability**. Pseudonymous, rotating scripts and short-lived transcripts resist linkage by default, yet selective disclosure is easy: a payer can reveal exactly one inclusion proof and nothing else. This is the **"minimum necessary"** approach regulators increasingly encourage—effective oversight, minimal data exhaust—made possible by the ledger's cryptographic structure.

Advantages over Bitcoin/ETH's and PoW/PoS

ENI6MA reduces **settlement latency** from minutes (Bitcoin's ˜10-minute blocks and multi-block confidence) or probabilistic epochs (many PoS chains) to **seconds-class finality** with a small, explicit confirmation depth. Users see payments land quickly; merchants can release goods with high confidence; and institutional flows can be tiered by value at risk without changing networks or assets.

ENI6MA **slashes operating cost**. Verification is a handful of hashes and XORs per input; headers are tiny; SPV requires no archival sync. There is **no energy externality** to recoup with fees, and no signature-verification storms under complex scripts. This yields predictable fee/throughput curves and makes high-integrity validation feasible on ordinary hardware, widening geographic and socioeconomic participation.

ENI6MA is **resilient to key loss and theft**. Because no long-lived signing keys appear on chain and authorization is event-bound, the blast radius of endpoint compromise is sharply time-limited. In contrast, leaks of private keys in PoW/PoS systems often imply irreversible asset loss or global re-key events. The measurable surface in ENI6MA—code identity, entropy, and committee signatures—is easier to monitor and remediate.

ENI6MA replaces the **hash-power honesty assumption** (PoW) and **economic majority/stake liveness assumptions** (PoS) with two orthogonal premises: **entropy freshness** and **threshold honesty** in a rotating committee. Both are **observable** (beacon proofs, signing participation) and parameterized (threshold t, committee size n, confirmation depth Δ), enabling formal security budgets: aggregate forgery probability bounded by $h \cdot \mathrm{negl}(\lambda)$ plus a tail for committee failure that decays with diversity and rotation.

ENI6MA is **post-quantum friendly by construction**. Its hot-path checks use symmetric primitives whose security degrades gracefully under Grover-class speedups (absorbed by larger digests). The only asymmetric element—the committee threshold signature—is modular and swappable for lattice or hash-based schemes without touching ledger semantics or client flows. Bitcoin and many PoS systems, by contrast, embed long-horizon dependence on ECDSA/EdDSA or pairing-based constructions that require deeper redesign for PQ migration.

ENI6MA improves **privacy without secrecy theater**. There are no static keys to correlate, no signature malleability to exploit, and no on-chain public keys that become heuristics glue. Short-lived transcripts bound to (x, T, q) give replay resistance and unlinkability "for free," while still allowing one-fact proofs for compliance. PoW/PoS systems often depend on key rotation hygiene and complex mixers to approach similar outcomes.

ENI6MA offers **cleaner liveness under load**. Deterministic ordering, fixed windows, and small headers minimize propagation variance and stale-block rates. PoW suffers naturally from variance in block discovery; PoS often faces networking and view-synchrony challenges tied to validator set size and message complexity. ENI6MA's fork choice ("longest valid by signed accumulators") and pipeline signing keep the control plane compact.

ENI6MA's **soundness reductions** are tight and composable. Single-input acceptance is negligible in λ for any PPT adversary without $\mathcal{M}$; union bounds scale to many concurrent attempts; and replay/precompute are nullified by entropy and time binding. PoW's game-theoretic incentives are powerful but indirect; PoS's slashing calculus is expressive but complex. ENI6MA's calculus is explicit: hashes, XORs, and thresholds with verifiable on-chain evidence.

ENI6MA-G delivers a **stable, bank-operable instrument** with continuous, cryptographically bound reserves—something neither PoW nor PoS supplies natively. Bitcoin provides bearer settlement but not solvency assurances for a fiat-priced peg; PoS tokens can represent claims but rarely make reserves public and machine-checkable each block. E6G's gram code plus reserve capsules closes this gap and **bridges regulated finance with verifiable crypto**.

ENI6MA trades PoW's "permissionless energy race" for a **governed randomness/signing service**. That introduces a new attack surface—committee

capture—that is mitigated (not eliminated) by size, diversity, rotation, slashing, and public proofs. The crucial difference is **observability**: when the committee fails, the failure is on chain; when hash-power colludes, the signal is the longest chain itself. ENI6MA embraces this trade with explicit parameters and live telemetry so institutions can **measure, insure, and hedge** the residual risk.

Taken together, the record points the same way: ENI6MA/RWP provides **faster, cheaper, cleaner, and more auditable** settlement while preserving decentralization where it matters (anyone can verify; anyone can transact) and enabling **bank-grade stability** through public, per-window reserve proofs. It does not discard Bitcoin's insight—timestamped public history—it **generalizes** it: time is still the spine, but *knowledge* replaces *work* as the scarce resource.

Appendix X — RWP for E6G (ENI6MA-G)

A.1 Overview (what the proof is proving)

For E6G, every economically relevant action is authorized by a **Proof-of-Knowledge (PoK)** instance derived from the Rosario–Wang framework with **HoloMorphic Entropy–based Witness Accumulation**. Two PoK families are used:

- **Payment PoK** (spend of UTXO value): proves the payer possessed the *right ephemeral knowledge* for the precise time and transaction context—without revealing any reusable secret.
- **Reserve PoK** (custodian attestation): proves a custodian had *live, immediate control* over reserve data (SKU, mass, vault) at the block's time and bound it to the header chain.

Both are verified publicly with symmetric operations only (hashes, XORs, length-delimited concatenations, and a threshold signature over the block accumulator). **Verifier time is linear in the number of probes, i.e.,** $O(n)$, where n is the probe count per input (or round count in the multi-round linguistic model). Memory is $O(1)$.

A.2 Notation (bridging the "language" model to E6G)

- **Alphabets & morphisms (language view):** Distinct alphabets Σ_i and bijective morphisms f_{ij} prevent ambiguity and let us "shuffle" representations without loss. In E6G, these encode *how* device state and public context are projected into transcript space.
- **Session values (systems view):**
  - x: high-entropy session nonce sampled by the prover device.
  - T: canonical block/window time.

– q: public context tuple; for **payments** q_{pay} includes referenced UTXO(s), amounts, recipient script, chain ID, policy tags; for **reserves** q_{res} includes $\mathsf{SKU}, \mathsf{mass}_t, \mathsf{vaultID}, \mathsf{audit\_ref}, A_{t-1}$.

- **Sealed morphism:** $\mathcal{M}$ (code identity) is the symmetric, sealed "private morphism" shared by prover and verifier implementations (compiled twins).
- **Ephemeral witness:**

  W ;=; f_{\mathcal M}(x, T, q)

  $$
  * **Transcript (one-time proof object):**

$$\tau \;=\; \mathsf{Transcribe}(W, q)$$

- **Designated probes (multi-round checks):** From (q, T, s_T) the verifier deterministically derives n probes $\{p_i\}_{i=1}^{n}$; the transcript carries masked responses $\{\rho_i\}$.
- **Membership test (accumulator in the linguistic sense):**

$$\Lambda \;=\; \bigwedge_{i=1}^{n} \big(\mathrm{XOR}(p_i, \rho_i) = 0\big)$$

- **Acceptance (per input):**

$$\mathsf{ACCEPT} \iff \Lambda = 1 \;\wedge\; \mathsf{PolicyOK}(q)$$

A.3 Construction (how the prover forms τ)

1. **Bind time & context:** Canonicalize q (length-delimited fields; domain-separated tags), fetch the **beacon seed** s_T for the current window, and fix T within the allowed drift.
2. **Sample entropy:** Draw $x \leftarrow \{0,1\}^{\kappa}$ from a device RNG (optionally salted with s_T).
3. **Project to witness:** Compute $W = f_{\mathcal{M}}(x, T, q)$ via a small cascade of domain-separated hashes/PRFs and lightweight permutations (the "holomorphic" projection).
4. **Derive probes:** Verifier and prover independently derive $\{p_i\}_{i=1}^{n}$ from $H(\mathsf{TAG[PRB]} \| T \| s_T \| q \| i)$.
5. **Mask responses:** Form $\rho_i = g_{\mathcal{M}}(W, i) \oplus p_i$, where $g_{\mathcal{M}}$ is a PRF keyed by W (or a derived subkey). Only the ρ_i values go on chain.

6. **Emit transcript:** τ encodes: version, context digest $H(q)$, probe count n, $\rho_1, \ldots, \rho_n$, and small integrity bytes (e.g., a transcript MAC under $H(W)$ domain-separated, *not* revealing W).

Intuition: With the right W, the prover can cancel each p_i via ρ_i. Without W, ρ_i is indistinguishable from random, and the chance that *all* XORs cancel is negligible.

A.4 Verification (algorithm and $O(n)$ complexity)

Input: τ, public q, window parameters (T, s_T), policy pol_T.
Steps (per input):

1. Recompute q from the transaction (or reserve capsule) bytes and policy tags.
2. Derive probes $\{p_i\}_{i=1}^{n}$ from (q, T, s_T) exactly as the prover did.
3. For each $i = 1..n$ do a fixed-width XOR and equality check: test $\mathrm{XOR}(p_i, \rho_i) = 0$.
4. Compute $\Lambda = \bigwedge_i$ of the results (constant-time AND accumulation).
5. Check $\mathsf{PolicyOK}(q)$ (jurisdiction bits, timelocks, supply caps, SKU integrity, etc.).

Cost: Each probe is a constant-time XOR/equality over a fixed word size; deriving the probe stream is one hash/PRF per index. Hence, **time is** $O(n)$ per input (linear in the number of probes/rounds); **space is** $O(1)$ (streaming derivation; no large tables).
Whole-transaction cost: For m inputs with probe counts n_j, the verifier cost is $\sum_{j=1}^{m} O(n_j)$.
SPV: Add $O(\log K)$ hashes for the Merkle branch; PoK checking remains $O(n)$.

A.5 Correctness, soundness, zero-knowledge-like properties

- **Completeness:** Honest provers possessing $W = f_{\mathcal{M}}(x, T, q)$ produce ρ_i that cancel the derived p_i for all i, so $\Lambda = 1$ and ACCEPT holds (assuming policy).
- **Soundness (single input):** For any PPT adversary without $\mathcal{M}$ and x,

$$\Pr[\mathsf{ACCEPT}] \leq \mathsf{negl}(\lambda) \;\approx\; 2^{-kn}$$

  where k is the effective hardness (bits) per probe and n the probe count.
- **Soundness (many opportunities):** For h concurrent attempts (e.g., many inputs or many forged disclosures),

$$\Pr[\exists \text{ forged}] \leq h \cdot \mathsf{negl}(\lambda)$$

- **Zero-leakage of reusable secrets:** No long-lived key appears; τ reveals masked responses only, bound to (x, T, q). Replays fail because probes change with T and s_T.

A.6 Reserve PoK for E6G (custodian side)

Reserve capsule (public commitment):

$$\mathsf{res}_t \;=\; H\big(\mathsf{TAG[RES]} \,\|\, \mathsf{SKU} \,\|\, \mathsf{mass}_t \,\|\, \mathsf{vaultID} \,\|\, \mathsf{audit\_ref}\big)$$

Reserve transcript: $\tau_t^{\mathrm{res}} = \mathsf{Transcribe}(f_{\mathcal{M}}(x, T_t, q_{\mathrm{res}}), q_{\mathrm{res}})$,
with $q_{\mathrm{res}} = (\mathsf{res}_t, A_{t-1}, \mathsf{issuerID}, \mathsf{policy\_bits})$.

Verification: Identical $O(n)$ PoK checks as for payments, then policy: (i) signer is an authorized custodian for $\mathsf{vaultID}$; (ii) mass_t changes are within tolerance ϵ; (iii) full-supply inequality holds once included. Include $(\mathsf{res}_t, \tau_t^{\mathrm{res}})$ under the block's content hash and accumulator.

Security angle: Without W for this *current* T_t and q_{res}, a stale or synthetic attestation fails the probe set. The PoK gives **freshness** and **control**; the capsule gives **binding**.

A.7 Payment PoK for E6G (payer side)

For each input u consumed by a transaction, form q_{pay} (UTXO id, amounts, recipient script, asset tag = E6G, chain ID, policy bits, etc.), compute W, then τ^{pay}. Nodes verify each input's PoK in $O(n)$, and enforce $\mathsf{PolicyOK}(q_{\mathrm{pay}})$ (e.g., jurisdiction tags, dust limits, timelocks). Value conservation and fee rules are standard UTXO arithmetic, independent of PoK.

Composability: Multi-input transactions simply conjoin per-input $\Lambda_j = 1$. Batching increases total cost linearly in $\sum n_j$, with excellent cache locality because XOR/equality are tight loops.

A.8 Probe generation and holomorphic accumulation

Probes implement the "multi-round linguistic verification" using a **holomorphic accumulation** of entropy and context:

$$p_i \;=\; H\big(\mathsf{TAG[PRB]} \,\|\, i \,\|\, T \,\|\, s_T \,\|\, H(q)\big), \qquad \rho_i \;=\; \mathsf{PRF}_W(i) \oplus p_i.$$

Equivalently, in the language view, each round R shuffles the active alphabet by a morphism f_R and checks membership of the "symbol" induced by W in the appropriate subset; the AND over rounds is Λ. The **computational footprint remains** $O(n)$ because both p_i and ρ_i are stream-derivable.

A.9 Complexity summary (payments, reserves, SPV)

- **Per input (PoK only):** $O(n)$ time, $O(1)$ space.
- **Per block (K inputs total):** $\sum_{j=1}^{K} O(n_j)$ for PoK + $O(1)$ for header checks + $O(K)$ to hash the body or $O(K \log K)$ if Merkle-building dominates (parallelizable).
- **SPV verification:** $O(n)$ for PoK + $O(\log K)$ for the Merkle branch + $O(1)$ for the header signature.
- **Parameterization:** Choose n and per-probe hardness k to achieve target $\approx 2^{-kn}$ false-accept probability while keeping per-input cost small (e.g., $n = 8..16$, $k = 8..16$).

A.10 Security reductions (outline)

1. **From forgery to PRF/Hash distinguishers:** A strategy that passes Λ without W yields either (i) a predictor for PRF_W outputs, or (ii) a correlation that violates the assumed pseudorandomness of H and probe derivation.
2. **Union bound to many attempts:** For h attempts within a window, total success probability increases at most linearly: $h \cdot 2^{-kn}$.
3. **Freshness vs replay:** Because probes depend on (T, s_T), any replay in a different window faces an independent probe set; success collapses to the negligible bound again.
4. **Reserve integrity:** Without control of q_{res} at time T, a custodian cannot make an old mass claim pass new probes; the capsule binds disclosures to headers, preventing equivocation.

A.11 Algorithms (concise pseudocode)

Prover (payments or reserves):

```
x ← RNG()
q ← CanonicalContext(...)
T,sT ← CurrentWindow()
W ← f_M(x, T, q)
for i in 1..n:
    pi  ← H(TAG_PRB || i || T || sT || H(q))
    r_i ← PRF_W(i) XOR pi
 ← Encode(version, H(q), n, r_1..r_n, integrity_bytes)
emit
```

Verifier (per input, O(n)):

```
q ← RebuildContextFromTxOrCapsule(...)
T,sT ← HeaderWindow()
acc ← 1
for i in 1..n:
    pi  ← H(TAG_PRB || i || T || sT || H(q))
    acc ← acc AND ( (pi XOR r_i) == 0 )
return (acc == 1) AND PolicyOK(q)
```

A.12 E6G-specific invariants and equations

Supply $\leq$ Reserves (by headers alone):

$$\mathsf{Supply}_t \;\leq\; \sum_{j\in\mathcal{C}} \frac{\mathsf{mass}_t^{(j)}}{G}\cdot(1-h^{(j)}), \quad G=1 \text{ gram/token}.$$

Valuation across currencies via the gram code:

$$V_c(t) \;=\; q_{\text{tokens}}\cdot G\cdot P_c(t).$$

Final acceptance rule (block): all included inputs satisfy $O(n)$ PoK checks and policy; header carries a valid threshold signature over

$$A_t \;=\; H(\mathsf{TAG[ACC]} \,\|\, A_{t-1} \,\|\, \mathsf{com}_t \,\|\, \mathsf{txh}_t \,\|\, \mathsf{pol}_t).$$

A.13 Practical tuning (banks & wallets)

- **Probe count n:** Choose n for target risk per input (e.g., $n=12, k=8 \Rightarrow 2^{-96}$); auditors can certify $h\cdot 2^{-kn}$ daily risk budgets.
- **Windowing:** 1–2 s cadence; SPV wallets see confirmations in Δ windows; all math is constant-time per probe.
- **Implementation:** Keep XOR/equality constant-time; derive probes streaming; pin domain tags; version transcript framing; log reason codes for policy failures (without revealing W structure).

A.14 Takeaway

For E6G, the Rosario–Wang Proof turns *fresh entropy + time + public context* into a one-time witness whose public verification is $O(n)$, side-channel-hard, and post-quantum friendly. Payments and reserve attestations share the same verifier, so the network's "cash rail" and "solvency rail" are secured by one compact primitive—easy to audit, easy to accelerate, and hard to cheat.

Appendix: Complete RWP/ENI6MA Equation Sheet (with TTS)

Below is a compact, self-contained set of definitions and equations drawn from the ENI6MA / E6G spec, each followed by a short text-to-speech (TTS) readout.

Notation & Operators

- $H(\cdot)$: collision-resistant hash.
  "H of dot, a collision-resistant hash."
- $\operatorname{PRF}_K(\cdot)$: pseudorandom function keyed by K.
  "P-R-F sub K of dot."
- $|$: unambiguous, length-delimited concatenation.
  "concatenate."
- $\oplus$: XOR over fixed-width words.
  "exclusive-or."
- $\bigwedge$: logical AND across a set.
  "logical and."
- Domain tags $\textsf{TAG[*]}$: domain-separation constants.
  "tag star."
- Equality checks are constant-time unless otherwise noted.
  "constant-time equality."

Core Variables

- Session entropy $x \in \{0,1\}^{\kappa}$ (fresh per spend/attestation).
  "x is kappa-bit fresh entropy."
- Canonical window time T.
  "capital T is the canonical time."
- Beacon seed s_T.
  "s sub T is the seed for time T."
- Public context q (UTXO ids, amounts, scripts, chain ID, policy tags; for reserves: SKU, mass, vault, audit ref, prior accumulator).
  "q is the public context tuple."

- Sealed symmetric morphism (code identity) $\mathcal M$.
  “script M is the sealed morphism.”

Ephemeral Witness & Transcript

1. **Witness binding**

$$W \;=\; f_{\mathcal{M}}(x, T, q).$$

“W equals f sub M of x, T, and q.”

2. **Probe derivation**

$$p_i \;=\; H\big(\mathsf{TAG[PRB]} \,\|\, i \,\|\, T \,\|\, s_T \,\|\, H(q)\big).$$

“p sub i equals hash of tag P-R-B, concatenate i, T, s sub T, and hash of q.”

3. **Masked responses**

$$\rho_i \;=\; \mathrm{PRF}_W(i) \;\oplus\; p_i.$$

“rho sub i equals P-R-F sub W of i exclusive-or p sub i.”

4. **Transcript object**

$$\tau \;=\; \mathsf{Encode}\big(\mathsf{ver},\, H(q),\, n,\, \rho_1, \ldots, \rho_n,\, \mathsf{int}\big).$$

“tau equals encode of version, hash of q, n, rho one through rho n, and integrity bytes.”

Verifier Predicate & Acceptance

5. **Probe-wise XOR membership**

$$\Lambda \;=\; \bigwedge_{i=1}^{n} \big(\, p_i \oplus \rho_i = 0 \,\big).$$

“Lambda equals the logical and over i from one to n of p sub i exclusive-or rho sub i equals zero.”

6. **Input acceptance**

$$\mathsf{ACCEPT} \iff \Lambda = 1 \ \wedge \ \mathsf{PolicyOK}(q).$$

"Accept if and only if Lambda equals one and Policy-O-K of q."

7. **Transaction-level acceptance (multi-input)**

$$\mathsf{ACCEPT}_{\mathrm{tx}} \iff \Big(\bigwedge_{j=1}^{m} \big(\Lambda_j = 1 \wedge \mathsf{PolicyOK}(q_j)\big)\Big) \ \wedge \ \mathsf{ConserveVal}.$$

"Transaction accept if and only if, for every j, Lambda sub j equals one and Policy-O-K of q sub j, and conserve value holds."

8. **Value conservation**

$$\sum_{j=1}^{m} \mathrm{in}_j \ = \ \sum_{k=1}^{n} \mathrm{out}_k \ + \ \mathrm{fee}.$$

"Sum of inputs equals sum of outputs plus fee."

Block Header Commitments

9. **Capsule commitment**

$$\mathsf{com}_t \ = \ H\big(\mathsf{TAG[CAP]} \,\|\, T_t \,\|\, H(\mathsf{TAG[SEED]} \,\|\, s_t) \,\|\, \mathsf{aux}_t\big).$$

"com sub t equals hash of tag cap, T sub t, hash of tag seed with s sub t, and aux sub t."

10. **Transactions fingerprint (concat form)**

$$\mathsf{txh}_t \ = \ H\big(\mathsf{TAG[TXH]} \,\|\, \tau_1 \,\|\, \cdots \,\|\, \tau_K\big).$$

"t-x-h sub t equals hash of tag T-X-H concatenated with tau one through tau K."

11. **Merkle root (pruning form)**

$$\mathsf{mrk}_t \ = \ \mathsf{MerkleRoot}(\tau_1, \ldots, \tau_K).$$

"m-r-k sub t equals the Merkle root of tau one through tau K."

12. **Accumulator**

$$A_t \;=\; H\big(\mathsf{TAG[ACC]} \,\|\, A_{t-1} \,\|\, \mathsf{com}_t \,\|\, \mathsf{txh}_t \,\|\, \mathsf{pol}_t\big).$$

“A sub t equals hash of tag A-C-C, previous A, com sub t, t-x-h sub t, and pol sub t.”

13. **Committee certificate**

$$\sigma_t \;=\; \mathsf{Sign}_{\mathcal{C}}(A_t).$$

“sigma sub t equals committee sign of A sub t.”

14. **Fork choice rule**

$$\mathsf{BestChain} \;=\; \arg\max_{\text{valid}} \; \#\{(A_t, \sigma_t)\}.$$

“Best chain equals the valid branch with the most signed accumulators.”

SPV & Inclusion

15. **Merkle inclusion check**

$$\mathsf{VerifyMerkle}(\tau, \mathsf{path}, \mathsf{mrk}_t) = \mathsf{true}.$$

“Verify Merkle of tau, path, and m-r-k sub t returns true.”

16. **SPV payment acceptance**

$$\mathsf{ACCEPT}^{\mathrm{SPV}} \iff \big(\mathsf{VerifyMerkle} \wedge \Lambda = 1 \wedge \mathsf{PolicyOK}(q)\big).$$

“S-P-V accept if and only if Merkle verifies, Lambda equals one, and Policy-O-K of q.”

Stablecoin E6G (Reserves & Valuation)

17. **Reserve capsule**

$$\mathsf{res}_t \;=\; H\big(\mathsf{TAG[RES]} \,\|\, \mathsf{SKU} \,\|\, \mathsf{mass}_t \,\|\, \mathsf{vaultID} \,\|\, \mathsf{audit\_ref}\big).$$

“res sub t equals hash of tag R-E-S, S-K-U, mass sub t, vault I-D, and audit reference.”

18. **Reserve transcript**

$$\tau_t^{\mathrm{res}} \;=\; \mathsf{Transcribe}\big(f_{\mathcal{M}}(x, T_t, q_{\mathrm{res}}),\, q_{\mathrm{res}}\big).$$

"tau superscript res sub t equals transcribe of f sub M of x, T sub t, q sub res with q sub res."

19. **Solvency inequality (multi-custodian)**

$$\mathsf{Supply}_t \;\le\; \sum_{j \in \mathcal{C}} \frac{\mathsf{mass}_t^{(j)}}{G} \cdot \big(1 - h^{(j)}\big), \qquad G = 1 \text{ gram/token}.$$

"Supply sub t is less than or equal to the sum over custodians of mass sub t divided by G times one minus haircut; G equals one gram per token."

20. **Tolerance band**

$$\left|\mathsf{Supply}_t - \frac{\mathsf{Mass}_t}{G}\right| \;\le\; \epsilon.$$

"Absolute value of supply minus mass over G is less than or equal to epsilon."

21. **Fiat valuation via the gram code**

$$V_c(t) \;=\; q_{\mathrm{tok}} \cdot G \cdot P_c(t).$$

"V sub c of t equals token quantity times G times P sub c of t."

22. **Currency conversion through grams**

$$\mathrm{Pay}_b \;=\; \mathrm{Recv}_a \cdot \frac{P_b(t)}{P_a(t)}.$$

"Pay sub b equals receive sub a times P sub b of t over P sub a of t."

Security Bounds & Freshness

23. **Per-probe false pass**

$$\Pr[p_i \oplus \rho_i = 0 \mid \text{guess}] \;=\; 2^{-k}.$$

"Probability of a probe passing by guess equals two to the minus k."

24. **Single-input soundness**

$$\Pr[\mathsf{ACCEPT}] \;\le\; 2^{-kn} \;=\; \mathsf{negl}(\lambda).$$

“Acceptance probability is at most two to the minus k n; negligible in lambda.”

25. **Union bound over h attempts**

$$\Pr[\exists\,\text{forgery}] \;\le\; h \cdot 2^{-kn}.$$

“Probability of any forgery is at most h times two to the minus k n.”

26. **Reorg tail (committee failure over Δ)**

$$\Pr[\text{reorg} \ge \Delta] \;\le\; \sum_{j=t}^{n} \binom{n}{j} \varepsilon^{j} (1-\varepsilon)^{n-j}.$$

“Probability of a reorg at least delta is at most the sum from j equals t to n of n choose j times epsilon to the j times one minus epsilon to the n minus j.”

27. **Replay failure (window change)**

$$\Pr[\Lambda = 1 \text{ on } (T', s_{T'}) \neq (T, s_T)] \;\approx\; 2^{-kn}.$$

“Probability Lambda equals one in a different window is about two to the minus k n.”

Complexity & Resource Bounds

28. **Per-input verifier**

$$T_{\text{verify}}(\text{input}) \;=\; O(n), \quad S_{\text{verify}}(\text{input}) \;=\; O(1).$$

“Verify time per input is big O of n; space is big O of one.”

29. **Per-block verification (K inputs)**

$$T_{\text{verify}}(\text{block}) \;=\; \sum_{j=1}^{K} O(n_j) \;+\; O(1).$$

“Block verify time equals the sum over inputs of big O of n sub j, plus constant.”

30. **SPV cost**

$$T_{\text{SPV}} \;=\; O(n) \;+\; O(\log K) \;+\; O(1).$$

“S-P-V time is big O of n plus big O of log K plus constant.”

Pruning & Archival

31. **Header-only retention invariant**

$$\forall t > t_0 + \Delta : \text{ retain } (A_t, \sigma_t, \mathsf{com}_t, \mathsf{mrk}_t), \text{ prune bodies.}$$

"For blocks deeper than delta, retain A sub t, sigma sub t, com sub t, and m-r-k sub t; prune bodies."

Deterministic Ordering & Conflict Rules

32. **Deterministic ordering key**

$$\mathsf{orderKey}(\tau_i) \;=\; H\big(\mathsf{TAG[SORT]} \,\|\, \tau_i\big).$$

"Order key of tau i equals hash of tag sort concatenated with tau i."

33. **RBF admissibility (same inputs, higher fee)**

$$\mathsf{Admit}_{\mathrm{RBF}} \iff \mathrm{Inputs\_identical} \;\wedge\; \mathrm{fee}' > \mathrm{fee}.$$

"Admit replace-by-fee if and only if inputs identical and new fee greater than old fee."

Time & Window Discipline

34. **Window pacing constraint**

$$|\, T_t - T_{\mathrm{local}} \,| \;\le\; \mathsf{Drift}_{\max}.$$

"Absolute difference between T sub t and local time is at most drift max."

35. **Probe seed binding to window**

$$p_i \text{ depends on } (T_t, s_t, H(q)) \;\Rightarrow\; \text{no cross-window reuse.}$$

"Probes depend on T sub t, s sub t, and hash of q; no cross-window reuse."

References

Foundational cryptography & proofs of knowledge

1. Goldwasser, S., Micali, S., Rackoff, C. "The Knowledge Complexity of Interactive Proof Systems." *STOC '85 / SIAM J. Comput.* (1989). (ResearchGate)

2. Goldreich, O., Goldwasser, S., Micali, S. "How to Construct Random Functions." *CRYPTO '84.* (SciSpace)

3. Fiat, A., Shamir, A. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." *CRYPTO '86.* (NDSS Symposium)

4. Schnorr, C.-P. "Efficient Identification and Signatures for Smart Cards." *CRYPTO '89 / J. Cryptology 1991.* (jcr.cacrnet.org.cn)

Timestamping, Merkle trees, and early e-cash

5. Haber, S., Stornetta, W.S. "How to Time-Stamp a Digital Document." *Journal of Cryptology*, 1991. (Bitcoin Optech)

6. Merkle, R.C. "Protocols for Public Key Cryptosystems." *IEEE S&P*, 1980. (King's College London)

7. Back, A. "Hashcash – A Denial of Service Counter-Measure." 2002. (GitHub)

8. Nakamoto, S. "Bitcoin: A Peer-to-Peer Electronic Cash System." 2008. (timroughgarden.org)

Accumulators, headers, SPV, and UTXO practices

9. Bitcoin whitepaper (SPV section; block headers as hash chain). (See #8). (timroughgarden.org)

10. Todd, P. "Merkle Mountain Ranges." OpenTimestamps write-up, 2016. (ResearchGate)

11. Bünz, B., Fisch, B., Szepieniec, A. "FlyClient: Super-Light Clients for Blockchains." *IEEE S&P 2020 (preprint 2019).* (ACM Digital Library)

Randomness beacons, seeds, and VRFs

12. Micali, S., Rabin, M., Vadhan, S. "Verifiable Random Functions." *FOCS '99.* (Trusted Computing Group)

13. RFC 9381 (IRTF CFRG). "Verifiable Random Functions (VRFs)." 2023. (tex2e.github.io)

14. NIST Interoperable Randomness Beacons: NISTIR 8213 (draft) "A Reference for Randomness Beacons." 2019. (NIST Publications, NIST CSRC)

15. NIST Beacon 2.0 posters/slides (architecture & pulse format). 2018–2020. (NIST CSRC)

16. drand / League of Entropy: distributed randomness beacon (paper, docs, and repo). 2019–2021. (NIST Publications, GitHub, Cloudflare)

17. SoK: "Decentralized Randomness Beacon Protocols." 2022. (ResearchGate)

Symmetric hashing & domain separation (post-quantum friendly path)

18. NIST FIPS 202. "SHA-3 Standard: Permutation-Based Hash and XOFs." 2015. (Trusted Computing Group)

19. O'Connor, J. et al. "BLAKE3: One Function, Fast Everywhere." 2020. (NIST Publications)

Fee markets & mempool policy

20. Buterin, V. et al. "EIP-1559: Fee market change for ETH 1.0 chain." 2019/2021.

21. Corallo, M. et al. "BIP-125: Replace-By-Fee." 2016.

Network privacy & transaction relay

22. Fanti, G. et al. "Dandelion: Redesigning the Bitcoin Network for Anonymity." 2017. (NIST CSRC)

23. Venkatakrishnan, S.B. et al. "Dandelion++: Lightweight Cryptocurrency Networking with Formal Anonymity Guarantees." *ACM SIGMETRICS/Performance 2020* (preprint 2019). (cmapscloud.ihmc.us)

Threshold signatures, DKG, and committee attestation

24. Shamir, A. "How to Share a Secret." *Communications of the ACM*, 1979. (MIT CSAIL)

25. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T. "Secure Distributed Key Generation for Discrete-Log Systems." *EUROCRYPT '99.* (IETF Datatracker)

26. Boneh, D., Lynn, B., Shacham, H. "Short Signatures from the Weil Pairing (BLS)." *ASIACRYPT 2001.* (Harvard Dash)

27. Boldyreva, A. “Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme.” *PKC 2003*. (IETF Datatracker)

Post-quantum cryptography (upgradable committee signatures)

28. NIST FIPS 203. “ML-DSA (based on CRYSTALS-Dilithium).” 2024. (Trusted Computing Group)

29. NIST FIPS 205. “SLH-DSA (SPHINCS+).” 2024. (NIST CSRC)

30. RFC 8391. “XMSS: eXtended Merkle Signature Scheme.” 2018. (cyber.gouv.fr)

31. Grover, L.K. “A Fast Quantum Mechanical Algorithm for Database Search.” *STOC ’96*. (arXiv)

Trusted execution / measured boot (for sealed morphism deployments)

32. TCG. “TPM 2.0 Library Specification.” v1.59 (2019). (iacr.org)

33. NIST FIPS 140-3. “Security Requirements for Cryptographic Modules.” 2019. (iacr.org)
