Contents

Elara Protocol: A Post-Quantum Universal Validation Layer for Digital Work

Version 0.7.35 Date: July 12, 2026 **Author:** Nenad Vasic **Contact:** nenadvasic@protonmail.com

For Enterprise, Government, and Defense Evaluators

This protocol was designed for operations where validation must be absolute and exposure must be zero. Every digital execution — every firmware decision, every sensor reading, every autonomous action — is cryptographically validated with post-quantum algorithms from the NIST-standardized families selected by CNSA 2.0 (ML-DSA-65, ML-KEM-768). The current implementation ships the FIPS 204 / FIPS 203 final encodings (ML-DSA-65 signatures fixed at 3,309 bytes, ML-KEM-768) — the verifier hard-rejects any legacy Round-3 signature length (Section 4.2). No cloud dependency. No blockchain dependency.

The protocol supports two deployment modes: **private networks** (no beats, no public exposure, organization controls every node) and a **public permissionless network** with an optional beat layer for witness incentivization. Private deployments use the same cryptographic protocol (Layer 1) and DAM data structure without any economic layer. Data never leaves the perimeter. Content can be validated without being revealed, and destroyed without leaving recoverable traces.

The minimum viable deployment is one device. The protocol operates offline, across air-gapped networks, through light-speed delays, and across planetary distances. Consensus is achieved through MESH-BFT (realized as Adaptive Witness Consensus) — a diversity-weighted Byzantine fault tolerant protocol where correlated witnesses are discounted; its safety, post-quantum security, and finality are proven in the companion paper. The safety and liveness cores are additionally bounded-model-checked in TLA+ with TLC and the handshake/record/admission cores in ProVerif — the runnable specs ship in `spec/tla/` and `spec/proverif/`; machine-checked proofs at unbounded scale remain ongoing. Two reference implementations exist: a Rust performance runtime (Layer 1.5) and a Python cognitive architecture (Layer 3), each with working test suites.

Relevant sections for private deployment: 3.5 (Minimum Viable Validation), 4 (Post-Quantum Cryptography), 7 (Interplanetary Operations), 8 (IoT and Hardware), 10.6 (Private Networks), 12 (Longevity and Enterprise Privacy). Section 9 covers public network economics — it does not apply to private deployments.

Abstract

As autonomous software and AI agents increasingly act across digital and physical infrastructure, a foundational question follows every action: *who — or what — performed it, in what form, and when* — and can that be proven beyond doubt, durably, against any future adversary? The Elara Protocol is infrastructure for answering it: a tamper-evident, **post-quantum** record of digital provenance — an undeniable footprint of *who did what, and when* — for any actor (a human, a device, or an autonomous AI agent) across any digital or physical system, from a $30 phone to a satellite.

Every digital execution generates a trust gap. A robotic arm on a factory floor makes 10,000 decisions per shift with no post-quantum audit trail. A satellite transmits telemetry across light-speed delays with no offline validation capability. An autonomous vehicle computes hundreds of safety-critical decisions per second with no immutable provenance chain. A manufacturing line produces billions of sensor readings per day with no cryptographic proof that the data was not altered after collection. These are not theoretical problems — they are operational gaps in every defense contractor, every space agency, every semiconductor fabrication plant, every autonomous vehicle program operating today.

Current systems for validating digital work — patents, copyright registries, blockchain times-

tamps — were designed for documents, not for industrial-scale digital execution. They are fragmented, expensive, jurisdiction-dependent, and vulnerable to quantum computing. No existing protocol validates work at this scale, across this range of devices, with this diversity of network conditions.

The Elara Protocol introduces a layered architecture for cryptographically validating all forms of digital execution and creation. Built on a novel data structure — the **Directed Acyclic Mesh (DAM)** — the protocol achieves instant local validation, partition-tolerant consensus, and interplanetary operation without requiring a universal clock. The DAM extends existing DAG architectures with three structural dimensions (time, concurrency, zone topology) and two orthogonal operational layers (classification-based projection, AI-assisted analysis).

The protocol specifies post-quantum signatures and key exchange from genesis using NIST standards (FIPS 203/204/205), with dual-signature strategy and algorithm agility designed to produce proofs that outlive the protocol itself. This drives decisions (PQC from genesis, algorithm agility, no institutional dependency, self-describing data formats) that produce more resilient engineering. The longevity property is grounded in a specific architectural feature: validation proofs are self-contained mathematical relationships between signed records, not database entries or application state. They can be moved to any future medium and remain independently verifiable, because the proof lives in the mathematics, not in the infrastructure that created it. The protocol's self-describing data format follows the Rosetta Stone principle: every record carries its own schema, algorithm identifiers, and human-readable serialization — decodable by any future system without external documentation. No specific timeframe is claimed; longevity is bounded by the underlying cryptographic assumptions (see Section 12.1 and Section 14 for honest limitations). The zero-knowledge validation layer ships SHA3-256 hiding commitments in Phase 1 — hash-based, with no classical-curve dependency in shipped code; the specified Groth16/BN254 zk-SNARK constructions are design-stage and rejected fail-closed on the wire until a prover/verifier lands, with a defined migration path to fully post-quantum ZKP constructions as lattice-based and hash-based proof systems mature (see Sections 5.3 and 14.3 for an honest assessment of this gap). Consensus is achieved through Adaptive Witness Consensus (AWC) — a continuous trust model targeting Byzantine fault tolerance at the standard 1/3 bound (machine-checked formal verification pending; see Section 14.6).

Enterprise, government, and defense deployments operate as private networks (Section 10.6) with no beat involvement. The optional beat layer applies exclusively to the public permissionless network. Private deployments use the same cryptographic protocol (Layer 1) and the same data structure (DAM) without any economic layer.

The minimum viable network is one device. A factory sensor validates locally without any network dependency. The same protocol scales down to a $30 phone validating creative work offline, for free, in sub-second time, with the same cryptographic proof available to a Fortune 500 corporation. This paper presents the complete protocol specification across 16 sections: DAM architecture, post-quantum cryptography, zero-knowledge validation, identity, interplanetary operations, IoT integration, public network economics, governance, adversarial resilience analysis addressing 35 attack vectors, and an honest assessment of limitations and open problems.

Table of Contents

---

1. Problem Statement

1.1 The Broken Validation Landscape

The systems that validate digital work were designed for a world of physical documents, national borders, and human-speed communication. They are failing at every scale — from individual creation to industrial execution.

Industrial validation is absent. Manufacturing lines generate billions of sensor readings with no post-quantum audit trail. Autonomous systems make safety-critical decisions with no immutable provenance. Satellite networks generate telemetry across light-speed delays with no offline validation capability. Defense contractors validate firmware and mission-critical software through ad-hoc processes with no cryptographic proof chain. These are not edge cases — they are the primary volume of digital work produced today, and no existing system addresses them.

Patents require government forms, specific character sets, jurisdictional filings, and thousands of dollars in fees. A software developer in Montenegro, Slovakia, or Iceland cannot file a provisional patent through the United States Patent and Trademark Office because the system rejects characters with diacritics, Cyrillic script, or non-ASCII names. The system that validates creative work cannot handle the creator's name.

Copyright is automatic in theory but unenforceable in practice. Proving creation date, establishing priority, and defending against infringement requires legal resources that individual creators cannot afford. The explosion of AI-generated content has made attribution even more intractable.

Blockchain timestamps solve the immutability problem but introduce new ones: high energy consumption, transaction fees, confirmation delays, limited throughput, and critical vulnerability to quantum computing. The ECDSA signatures used by major blockchains will be broken by Shor's algorithm on sufficiently powerful quantum computers. (Hash functions like SHA-256

retain ~128-bit security under Grover's algorithm — it is the signing keys, not the hashes, that are vulnerable.) No major blockchain has completed migration to post-quantum cryptography.

Centralized registries (package managers, code hosting platforms, container registries) provide practical version control but create platform dependency. An account suspension — triggered by an automated system, a billing dispute, or a policy misinterpretation — can temporarily lock a developer out of their own work. The creator's access depends on the platform's continued operation and policies.

1.2 The Scale Problem

Current validation systems were designed for documents, not for the volume of digital work produced today:

- A single IoT deployment generates millions of sensor readings per day, each requiring provenance
- Autonomous vehicles make hundreds of decisions per second, each with liability implications
- AI systems produce outputs that must be attributed to specific models, versions, and prompts
- Robotic surgical systems require immutable audit trails for every action
- Satellite networks generate telemetry across time delays of minutes to hours

No existing protocol validates work at this scale, across this range of devices, with this diversity of network conditions.

1.3 The Quantum Threat

Quantum computing is not a theoretical concern. NIST finalized its first post-quantum cryptographic standards in 2024. The cryptographic foundations of every major blockchain — RSA, ECDSA, EdDSA — will be vulnerable to Shor's algorithm on sufficiently powerful quantum hardware.

The migration challenge is immense. Retrofitting post-quantum cryptography onto an existing blockchain requires hard forks, community consensus, and backward compatibility — a process that will take years per network. During the transition window, existing signatures become progressively less trustworthy.

A new protocol, unburdened by legacy, can implement post-quantum cryptography from its first byte.

1.4 The Interplanetary Gap

No existing distributed ledger is designed for interplanetary operation. Blockchain consensus mechanisms assume network latency measured in seconds, not minutes. A Mars colony operating with 3–22 minute one-way communication delay cannot participate in real-time consensus with Earth nodes.

As humanity expands beyond Earth — with permanent lunar bases planned for the 2030s and Mars missions under active development — the infrastructure for digital trust must be designed now for the network conditions of the future.

1.5 Design Goals

The Elara Protocol addresses these failures with the following design goals:

1. **Scale** — handle IoT-volume data (billions of readings per day) with high-throughput local validation (aggregate performance benchmarks pending)
2. **Sovereignty** — every node is self-sufficient; minimum viable network is one device, no network dependency required
3. **Quantum safety** — post-quantum cryptography from genesis
4. **Partition tolerance** — operate across network splits, including interplanetary distances
5. **Privacy** — prove creation without revealing content
6. **Universality** — validate any digital work, from a factory sensor reading to a satellite telemetry stream to a poem
7. **Proof longevity** — validation proofs are designed to outlive the protocol itself, with no institutional dependency. Achievable because proofs are mathematical relationships, not application state — they survive the death of every system that created them and can be verified on any future medium. The protocol's algorithm agility (Section 4.4) and the specified key-rotation design (Section 11.2, not yet operational) are the intended migration paths when cryptographic algorithms require replacement. No specific timeframe is claimed — longevity depends on the underlying cryptographic assumptions holding and the ability to evolve when they don't
8. **Accessibility** — run on devices from a $4 microcontroller (via gateway-delegated signing, Profile C) to a datacenter

For enterprise, government, and defense deployments that require private operation with zero beat involvement, see Section 10.6 — Private Networks and Network Publication. These deployments use the same cryptographic protocol and data structure as the public network.

---

2. Related Work

2.1 Blockchain-Based Systems

Proof-of-work blockchains (2009–) proved that decentralized consensus is possible without trusted intermediaries. However, proof-of-work consensus is energy-intensive, limited in throughput (~7 transactions per second on first-generation networks), and architecturally incompatible with high-throughput validation of arbitrary digital work.

Smart contract platforms extended blockchain with programmable validation logic. Transitions to proof-of-stake improved energy efficiency but did not address base-layer throughput limitations (~30 TPS), quantum vulnerability, or interplanetary operation. These ecosystems are optimized for decentralized finance, not universal work validation.

High-throughput ledgers achieve up to ~65,000 TPS through novel consensus mechanisms such as proof-of-history, though with elevated hardware requirements for validators. Their architectures assume low-latency, high-bandwidth connectivity — the opposite of interplanetary conditions.

All blockchain-based systems share fundamental limitations for universal validation: block-based batching introduces latency, linear chain structure creates bottlenecks, and consensus mechanisms assume Earth-bound network conditions.

2.2 DAG-Based Systems

Directed Acyclic Graph (DAG) based distributed ledgers (2015–) eliminated blocks and miners, using transaction-based structures where each new transaction validates previous ones. The most mature DAG architectures are the closest relatives to the Elara Protocol in structure, but diverge in scope:

- They target machine-to-machine micropayments or value transfer, not universal work validation
- Most do not implement post-quantum cryptography natively
- Some rely on centralized finality mechanisms during their maturation phase
- They lack zero-knowledge validation for privacy-preserving attribution
- They were not designed for interplanetary partition tolerance and would require significant architectural changes to support it

Block-lattice variants (where each account maintains its own chain) achieve instant finality and zero fees, but are narrowly scoped to value transfer and lack the extensibility required for universal work validation.

2.3 Interoperability Protocols

Cross-chain interoperability layers connect multiple blockchains and legacy systems, serving an important role in the current fragmented landscape. Key differences from the Elara Protocol's approach:

- Most are proprietary and enterprise-licensed — not open infrastructure
- They bridge existing networks rather than providing native validation
- They do not implement post-quantum cryptography
- They are not designed for interplanetary or high-latency operation and assume low-latency network conditions
- Their scope is enterprise integration, not universal work attribution

2.4 Oracle Networks

Oracle networks connect blockchains to external data sources, attesting that off-chain data is accurate. These serve a critical role in blockchain ecosystems for data that originates outside the network (price feeds, weather data, off-chain events).

For device-generated data specifically, the Elara Protocol takes a different approach: devices sign their own readings with their own cryptographic keys at the point of measurement, reducing dependence on third-party attestation for sensor and IoT data. Oracle networks remain valuable for data that does not originate from a signing device.

2.5 Payment Settlement Networks

Real-time gross settlement systems and institutional currency exchange networks provide fast value transfer between financial institutions. These operate at a fundamentally different layer — they move value between parties, they do not validate the creation of digital work. The two models are complementary rather than competitive.

2.6 Intellectual Property Systems

International patent filing systems (such as the PCT system) cost $3,200–$3,800 per filing, require specific document formats and character sets, take 18–30 months for preliminary examination, and provide no protection for non-patentable creative work (art, music, data, AI outputs).

Existing open-source attribution and timestamping tools provide lightweight alternatives but lack cryptographic validation, privacy, or network consensus. They are tools, not protocols.

2.7 Decentralized Storage

Content-addressed decentralized storage networks identify files by their hash, with economic incentives for storage providers. These are complementary to the Elara Protocol — they could serve as off-DAM content storage layers. However, storage networks do not provide validation, attribution, or privacy. They store data; they do not prove who created it or when.

Permanent storage networks with one-time payment models align with the Elara Protocol's proof longevity goals and could serve as archive backends for the Elara DAM.

2.8 Decentralized Identity Standards

W3C Decentralized Identifiers (DIDs) (2022) is a standard for self-sovereign identity. DIDs share the Elara Protocol's philosophy: identities are self-generated, not authority-issued. The Elara Protocol's identity system (Section 6) is compatible with the DID standard and can be expressed as a DID method:

`did:elara:<identity_hash>`

This compatibility is intentional. The Elara Protocol does not reinvent identity — it extends the DID model with post-quantum cryptography, entity type classification (HUMAN/AI/DEVICE/ORGANIZATION/COMPOSITE), digital succession, and integration with the DAM's validation layer. Future work includes formal registration as a DID method with the W3C.

2.9 Gap Analysis

Capability	PoW Blockchains	Smart Contract Platforms	DAG Systems	Interoperability Layers	Oracle Networks	Elara Protocol
Universal work validation	No	Partial	No	No	No	**Design**
Post-quantum cryptography	No	No	No	No	No	**Specified**
Zero-knowledge privacy	No	Partial	No	No	No	**Design**
Interplanetary operation	No	No	No	No	No	**Design**
IoT-scale throughput	No	No	Partial	No	No	**Design**
No central authority	Partial	Partial	Transitioning	No	No	**Design**
Proof longevity	No	No	No	No	No	**Design**
Multi-dimensional structure	1D (chain)	1D (chain)	2D (DAG)	N/A	N/A	**3D + 2 layers (DAM)**

Design indicates specification-complete; production validation is pending. **Specified** indicates algorithms selected and integrated into protocol design; not yet implemented in reference code.

The Elara Protocol operates at a different layer than most existing systems — universal validation infrastructure, not financial settlement, not smart contracts, not oracle services. While

there may be overlap at specific boundaries (e.g., timestamping, identity), the protocol is designed to complement rather than replace existing infrastructure. It is a validation layer that existing systems could eventually integrate.

2.10 Provenance, Timestamping, and Authority-to-Act

The most direct way to situate the protocol against existing work is to answer the question a careful reviewer asks first: *why not simply compose OpenTimestamps with a post-quantum signature?* Composed, those two primitives prove exactly one thing — **"this key signed these bytes, and the bytes existed by this Bitcoin block."** That is *when* something happened and *which key* touched it. Neither primitive, alone or composed, carries any notion of **delegation, scope, or revocation**, so the pair structurally cannot express the proposition that matters when an autonomous agent acts on someone's behalf:

> *Agent A was mandated by principal P to perform action X; the mandate was valid at the act's signing time; and it was later revoked.*

Expressing that requires a third layer — an authority-and-accountability record that binds an act to the mandate under which it was performed and resolves, at verification time, whether that authority held when the act was sealed. The Elara Protocol provides that layer natively: post-quantum, offline-recomputable, and queryable as a time-aware ledger of *acts under authority* rather than a credential bolted onto a signature. Section 6.3 describes the AI-attribution use case the layer is built for; the reference runtime ships a dependency-free demonstration that reproduces every verdict discussed below from the same verification function the live node exposes over RPC.

The honest scope of "structurally cannot." The claim above is true of the OpenTimestamps-plus-signature *strawman* — those two primitives and nothing else. It is **not** a claim that no system can express a mandate. Several can (see Verifiable Credentials below); anyone who adds the missing authority layer has built something that does what this layer does. The protocol's contribution is not the idea of authority-to-act but shipping it as one integrated, post-quantum, offline-verifiable mesh in which the authority record and the act share the same witnessed ledger. The remainder of this subsection compares the protocol against the specific systems most often proposed as substitutes, conceding what each does well and isolating the single capability — a queryable, post-quantum record of *acts under delegated, revocable authority* — that distinguishes it.

OpenTimestamps. OTS proves that a single hash existed by a given Bitcoin block, and nothing about who produced it, under whose authority, or whether that authority was valid; its own documentation is explicit that it does not prove authorship. Calendar servers are queryable, but only to upgrade one commitment's Bitcoin proof — there is no queryable, time-aware history of *acts* or *authority*. The protocol *uses* OTS as one leg of its external anchor braid (an existed-by upper bound), not as a substitute for the authority layer.

A bare signature, post-quantum or classical. A Dilithium3 / ML-DSA signature answers `Verify(pk, msg, σ) → {0,1}` — "the holder of key K signed these bytes." No signature standard introduces an issuer chain, a scope, a validity window, or revocation; that is precisely why credential layers exist on top of raw signatures.

W3C Verifiable Credentials and X.509 attribute certificates. These *can* express delegated, scoped, revocable authority, and the protocol does not claim otherwise. Verifiable Credentials 2.0 (a W3C Recommendation since 2025) with a Bitstring Status List express a scoped, time-bounded, revocable grant; X.509 attribute certificates (RFC 5755) have encoded delegated privilege for years. If the goal is only to *express* a mandate, these are real options. The honest distinction is *what kind of artifact each is*. A verifiable credential or attribute certificate is a **credential format** — a statement one *presents*, answering "is this credential valid right now?"; the issuer hosts its own revocation/status service, and the credential is bolted onto

the surrounding system. The protocol instead records the **act**: a witnessed, content-addressed, time-anchored ledger entry that *references* its mandate and answers a harder question — "was the authority valid at the exact, sealed moment this act was signed, given that revocation may have occurred afterward?" The authority record and the act live on the **same** witnessed ledger; revocation is evaluated at read time and is front-run-proof, keyed to the principal so that a non-principal's purported revocation is never consulted. The signing layer is post-quantum today, whereas W3C post-quantum cryptosuites remain experimental as of 2026.

sigstore. sigstore is strong infrastructure for open-source supply chains, and the comparison must be precise about what it does today. It *can* verify offline: cosign bundles carry the Rekor inclusion proof and a signed timestamp, and `cosign verify --offline` is shipped. Its public instance is not yet post-quantum as of 2026, though experimental ML-DSA support is landing in the client and bundle tooling. What sigstore has no concept of is **authority-to-act**: a Fulcio certificate binds a signing identity (an OIDC token, a CI workflow) to a key, and Rekor logs the signing event, but neither states that *principal P authorized agent A, within scope S, revocably*. gitsign and attestations record *who signed an artifact*, not *who was authorized to*. sigstore is built for public transparency logs and internet OIDC, not for a long-running, possibly private, validation mesh; that authority gap — not offline-ness or post-quantum cryptography alone — is the difference.

C2PA / Content Credentials. C2PA is media-provenance metadata under a consortium PKI. Plain manifests can be stripped by re-encoding or by platforms that discard metadata; C2PA 2.1 and later answer this with Durable Content Credentials (an invisible watermark, a perceptual fingerprint, and a cloud manifest lookup) designed to survive stripping — at the cost of reintroducing a hosted database and a network dependency, and the watermark is not invulnerable to adversarial editing. C2PA manifests *do* carry a time authority: the specification uses RFC 3161 TSA timestamps — a *trusted-authority* timestamp, not a trustless anchor. What C2PA is not is a consensus network, a validator quorum, or a queryable ledger of authorized acts; it attests *who made a piece of media and how it was edited*, not *who was authorized to perform an action, under what mandate*. The two compose cleanly: a C2PA manifest hash is a perfectly good thing to record on the protocol.

Certificate Transparency. CT (RFC 6962 / 9162) is the closest structural cousin — an append-only Merkle log audited by public monitors. But CT logs certificate *issuance*: it proves that a CA issued a certificate, and by design not that a key was *authorized to perform an act*, with what scope, still valid at a later moment. CT is issuance transparency for the web PKI, on classical cryptography; the protocol is an append-only, post-quantum record of *acts under delegated authority*, with validity-at-signing and revocation as first-class, offline-recomputable ledger semantics. The problems are orthogonal.

These comparisons converge on a single sentence:

> OpenTimestamps proves *when* a hash existed and a post-quantum signature proves *which key* signed it, but neither — alone or composed — expresses whether that key was *authorized* to act. The Elara Protocol adds a post-quantum, queryable ledger that records each act's reference to a revocable, time-bounded mandate from a named principal and deterministically resolves — offline, re-runnable years later — whether that authority held at the act's signing time, on the same witnessed ledger as the act itself.

What this layer enforces today (honest-claims rule). The mandate layer is an observational v0, and the protocol states its scope precisely rather than implying more. Enforced in v0 are the *who* (binding an act to a mandated agent identity), the *when* (the mandate's validity window), and **revocation** (evaluated at read time, front-run-proof, keyed to the principal). Acts outside their mandate are **recorded and flagged** with a typed taxonomy — `NO_CHAIN`, `AGENT_MISMATCH`, `LAPSED`, `NOT_YET_VALID`, `POST_REVOCATION`, `OVER_SCOPE` — and carry zero trust or consensus weight; a truth ledger records violations rather than silently discarding them.

Not yet enforced in v0: operation-, zone-, and amount-level scope is *recorded but not yet checked*; consensus-weight enforcement (gating stake and committee standing on mandate validity) is the next implementation slice, and sub-delegation chain-walking follows it. The protocol never claims a scope check that the reference code does not perform.

---

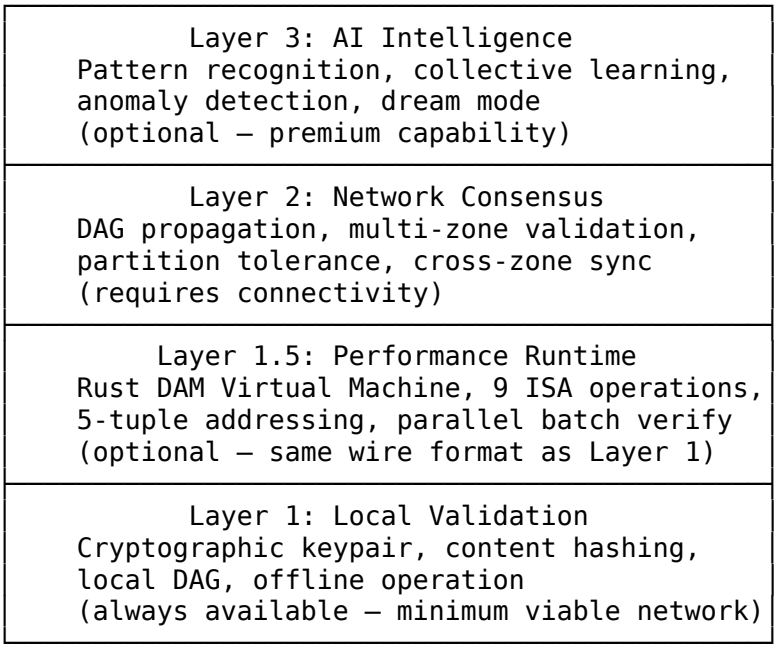
3. Protocol Architecture

3.1 Design Philosophy

The Elara Protocol follows three architectural principles:

1. **Sovereignty first** — every node must be fully functional in isolation. Network participation enhances trust but is never required for local validation.
2. **Branches merge, nothing is lost** — the DAG structure preserves all history, including conflicting claims. Conflicts are annotated, not resolved by deletion.
3. **The creation event IS the proof** — validation happens at the moment of creation, not after the fact. There is no delay between creating work and proving you created it.

3.2 Layered Architecture



Layer 1: Local Validation Every Elara node maintains:

- A **cryptographic keypair** (post-quantum, self-generated)
- A **local DAG** of all work validated by this node
- A **content-addressable store** for work artifacts
- A **validation engine** that hashes, signs, and timestamps locally

When a creator produces work, the node:

1. Computes a cryptographic hash of the content (SHA3-256)
2. Creates a validation record containing: content hash, creator's public key, timestamp, causal references to prior work, and classification level (public/private/restricted/sovereign)

3. Signs the validation record with the creator's private key (CRYSTALS-Dilithium)
4. Appends the signed record to the local DAG
5. Optionally wraps the content hash in a zero-knowledge proof (for private/restricted work)

This process completes in milliseconds on commodity hardware and requires no network connectivity. A validation created on an airplane, a submarine, or the surface of Mars is cryptographically valid the moment it is signed.

Layer 1.5: Performance Runtime Layer 1 defines the protocol semantics — what a valid record is, how it is signed, how it references parents. Layer 1.5 provides a high-performance implementation of those same operations in Rust, with the same wire format and byte-identical output. A record produced by Layer 1 (Python) and a record produced by Layer 1.5 (Rust) are indistinguishable on the network.

The Elara Runtime (Layer 1.5) implements:

- A **DAM Virtual Machine** with all 9 primitive operations defined in the Hardware Whitepaper (v0.2.0, Section 5.2): `DAM_INSERT`, `DAM_QUERY`, `DAM_WITNESS`, `DAM_HASH`, `DAM_SIGN`, `DAM_VERIFY`, `DAM_MERGE`, `DAM_CLASSIFY`, `DAM_ANALYZE`
- **5-tuple dimensional addressing** (`T, C, Z, K, A`) — the same addressing model that native hardware will implement physically
- **Tiled storage** with in-memory DAG index for sub-millisecond record lookup
- **Parallel batch verification** via Rayon — verifying multiple signatures concurrently on multi-core hardware
- **PyO3 bindings** — callable from Python as an optional fast path, transparent to the application layer

Layer 1.5 is optional. Layer 1 (Python) is the universal baseline that runs on any device. Layer 1.5 provides 10–100x performance improvements on devices with Rust support (laptops, servers, capable phones), bridging the gap between Layer 1's universality and the native hardware performance described in the Hardware Whitepaper. The progression is: Layer 1 (Python, available now) → Layer 1.5 (Rust, available now) → native hardware (FPGA prototyping 2027, ASIC 2029+).

No layer depends on the layers above it. Layer 1 is universal. Layer 1.5 is an acceleration of Layer 1. Layer 2 requires connectivity. Layer 3 is optional.

Layer 2: Network Consensus When network connectivity is available, nodes propagate validation records to peers. The DAG structure allows:

- **Asynchronous propagation** — no block intervals, no mining, no waiting
- **Parallel validation** — multiple branches of the DAG grow simultaneously
- **Conflict preservation** — if two nodes validate conflicting claims (e.g., two people claim authorship of the same work), both records are preserved with timestamps. The DAG does not resolve the conflict — it records it for human or legal resolution.

Consensus is achieved through **witness accumulation**: as more nodes receive and acknowledge a validation record, its trust score increases. A validation witnessed by 1 node is locally valid. A validation witnessed by 1,000 nodes across 50 countries is globally attested.

Settlement provides threshold guarantees: once a record accumulates attestations from witnesses representing ≥2/3 of diversity-weighted stake, it is considered settled — the cost of reversal exceeds the value of any plausible attack. However, trust continues accumulating beyond settlement. A record with 100 diverse witnesses is more trusted than one with the minimum settlement threshold, even though both are settled. Trust is continuous, not binary. A record is always valid from the moment of local signing, with increasing levels of network attestation building confidence over time.

Layer 3: AI Intelligence The optional AI layer provides:

- **Pattern recognition** — detecting anomalies in validation streams (e.g., a sensor producing physically impossible readings)
- **Collective learning** — anonymized patterns shared across consenting nodes improve the network's ability to detect fraud, predict failures, and optimize routing
- **Continuous autonomous thinking** — a 15-phase analysis engine that runs every 2 hours, covering pattern recognition, self-review, memory consolidation, and insight synthesis. Periodic "dream" modes (weekly, monthly, emotional) provide deeper analysis over longer timeframes. Inherited from Elara Core's existing cognitive architecture.
- **Cognitive Continuity Chain** — cryptographic proof of unbroken cognitive experience via hash-chained, dual-signed state snapshots. Each snapshot captures a `CognitiveDigest` — mood vector, memory/model/prediction/principle/correction counts, active goals, allostatic load — and chains it to the previous snapshot via DAG parent references. Six trigger events (boot, shutdown, milestone, drift, manual, periodic) generate snapshots, rate-limited to prevent flooding. The chain is verifiable: walk the DAG backwards to confirm no gaps in the cognitive record. This transforms AI cognition from an opaque process into a cryptographically auditable trail. (See Elara Core Whitepaper v1.5.1, Section 13.4 for implementation details.)
- **Natural language interface** — querying the validation history in human language

Layer 3 is explicitly optional. The protocol is fully functional without AI. This layer exists for nodes that choose to contribute computational resources in exchange for enhanced capabilities.

Minimum capability completeness: The three capability layers — validation (Layer 1), network consensus (Layer 2), and intelligence (Layer 3) — represent the minimum layering for a self-sustaining distributed validation system. This is provable by elimination: validation without networking is isolated (no trust propagation); networking without intelligence is blind to cross-dimensional patterns (no anomaly detection, no learning); intelligence without validation has nothing trustworthy to reason about. Each layer eliminates a distinct failure mode, and removing any layer produces an incomplete system. This parallels the minimum dimensionality principle in rotating field theory (see Hardware Whitepaper, Section 3.8.1): three phases (120° apart) is the minimum for smooth rotation because the roots of unity in Z/3Z sum to zero — each phase covers an orthogonal component of the rotating field, just as each capability layer covers an orthogonal failure mode of the distributed system. Layer 1.5 (Rust performance runtime) is an acceleration of Layer 1, not a fourth capability — it implements the same 9 operations on the same 5-tuple addressing with byte-identical wire format. On native hardware (Hardware Whitepaper, Section 8), Layer 1.5's operations are absorbed directly into the ISA, and the three-layer capability structure is physically implemented by the two die triads (see Hardware Whitepaper, Section 3.8.2).

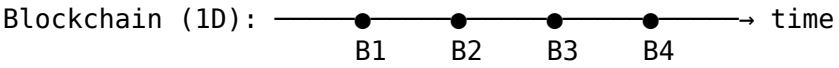
3.3 From DAG to DAM: The Directed Acyclic Mesh

3.3.1 Why Not Just a DAG A Directed Acyclic Graph is a graph with directed edges and no cycles. Existing DAG-based distributed ledgers have proven this structure effective for their respective use cases. The Elara Protocol's data structure extends beyond what a standard DAG describes, adding structural and operational axes that address different requirements.

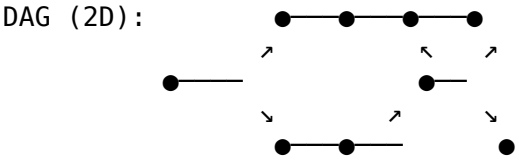
To understand why, consider dimensionality as an architectural metaphor. The following analysis describes the structural axes of these data structures — independent parameters that define their state space — rather than claiming formal mathematical dimensionality in the topological sense.

3.3.2 Dimensional Analysis of Distributed Ledgers **Blockchain is one-dimensional.** It is a line. Block follows block follows block. One path, one direction, one history. If the chain

forks, one branch must die. Blockchain cannot represent two simultaneous realities — it must collapse them into sequence. This is a fundamental geometric limitation, not an implementation detail.



A basic DAG is two-dimensional. Events spread across both time and concurrency. Multiple branches coexist, merge, and diverge. Existing DAG-based distributed ledgers operate effectively in these two dimensions.



The Elara Protocol adds a third structural dimension — zone topology — and two orthogonal operational layers. Each represents an independent axis:

Structural Dimensions:

Dimension	Axis	What It Represents
1st	Time	Causal ordering via vector clocks and DAG references
2nd	Concurrency	Parallel branches within a single zone
3rd	Zone topology	Independent DAGs per zone (Earth, Mars, Luna) that merge across planetary distances

Operational Layers (orthogonal features over the structure, not additional geometric dimensions):

Layer	Function	What It Provides
Classification	Observer-dependent projection	The same record presents differently depending on observer clearance (PUBLIC sees content, PRIVATE sees proof, SOVEREIGN sees only math)
Intelligence	AI pattern recognition	Cross-zone, cross-classification, cross-temporal analysis — connections invisible to any single node or zone

A blockchain is a line. A DAG is a surface. The Elara structure is a **mesh** — a network of interconnected DAGs that appears simple locally (each zone is a standard DAG) but forms a self-healing, multi-path topology globally.

Minimum mesh dimensionality: The three structural dimensions (time, concurrency, zone topology) are the minimum for a directed acyclic mesh. Removing any single structural dimension reduces the DAM to a previously known data structure: without time (D1), the structure is an unordered concurrent set — no causal reasoning is possible; without concurrency (D2), the structure is a single-threaded DAG — a blockchain; without zone topology (D3), the structure is a single-zone DAG — scalable within one partition but unable to survive or heal from network splits. Three structural dimensions is the minimum for a data structure that simultaneously supports causal ordering, concurrent state, and partition-tolerant topology. This is provable by elimination: each dimension eliminates a distinct failure mode (temporal blindness, sequential bottleneck, partition fragility), and no two dimensions together cover all three failure modes. This is the data-structural analog of the minimum dimensionality result for rotating

fields: three phases is the minimum for smooth rotation because each phase covers an orthogonal component; three structural dimensions is the minimum for mesh completeness because each dimension eliminates a distinct failure mode (see Hardware Whitepaper, Section 3.8.1). The two operational layers (classification, AI analysis) project onto this mesh without modifying its topology — they are views of the existing dimensional coordinates, not additional axes.

3.3.3 Formal Definition: Directed Acyclic Mesh (DAM) A **Directed Acyclic Mesh** is a distributed data structure defined as a tuple **M = (Z, V, E, C, π, A)** where Z is a set of zones, V is a set of validation records, $E \subset V \times V$ is a set of directed causal edges (acyclic), C is a classification function $C: V \times \text{Observer} \to \text{View}$, π is the partition-merge operator, and A is the cross-zone analytics function. The DAM satisfies the following properties:

1. **Locally flat** — for any zone $z \in Z$, the restriction $M|z = (V\_z, E\_z)$ is a standard DAG. Nodes create validation records that reference parent records. Edges are directed. Cycles are impossible (enforced: a record's hash includes its parents' hashes, making cycles computationally infeasible). Any engineer who understands a DAG understands a local view of the DAM.
2. **Globally interconnected** — across zones, the structure forms a self-healing mesh topology. Zone-DAGs operate independently during partitions and merge when connectivity resumes via the partition-merge operator $\pi: M|z_1 \times M|z_2 \to M|z_1 \cup z_2$. The merge operation preserves both branches — the mesh routes around the partition rather than breaking. π is commutative (merge order doesn't matter) and idempotent (re-merging is a no-op).
3. **Observer-dependent** — the classification function C(v, o) projects a validation record v into an observer-specific view based on cryptographic clearance level. This is analogous to projection in geometry: a 3D object casts different 2D shadows depending on the angle of light. The DAM casts different "shadows" depending on the observer's cryptographic clearance. Formally: for observers o_1, o_2 with clearance levels $l_1 < l_2$, the information in $C(v, o_1) \subseteq C(v, o_2)$ — higher clearance strictly reveals more.
4. **Analytically connected** — the analytics function A operates across the full mesh $A: M \to \text{Insights}$, detecting patterns that span zones, classification levels, and time periods — connections that no single-zone view can reveal. A has read access to the full DAM but cannot modify records.
5. **Partition-preserving** — unlike blockchain (which resolves forks by deletion) or basic DAGs (which assume continuous connectivity), the DAM treats partitions as expected topology changes. Formally: if the network partitions into zones $\{z_1, z_2\}$, both $M|z_1$ and $M|z_2$ grow independently, and upon reconnection $\pi(M|z_1, M|z_2)$ preserves all records from both branches with no data loss. Disconnected zones are segments of the mesh growing independently, destined to reconnect.

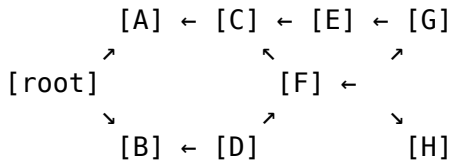
The name is chosen deliberately. In networking, a mesh is a topology where nodes interconnect directly, creating multiple paths and self-healing capability — if one link fails, traffic routes around it. The Elara DAM is a trust mesh: locally simple (each zone is a standard DAG), globally resilient (zones operate independently during partitions and merge seamlessly when connectivity returns).

3.3.4 Structure Within each zone, the DAM operates as a standard DAG:

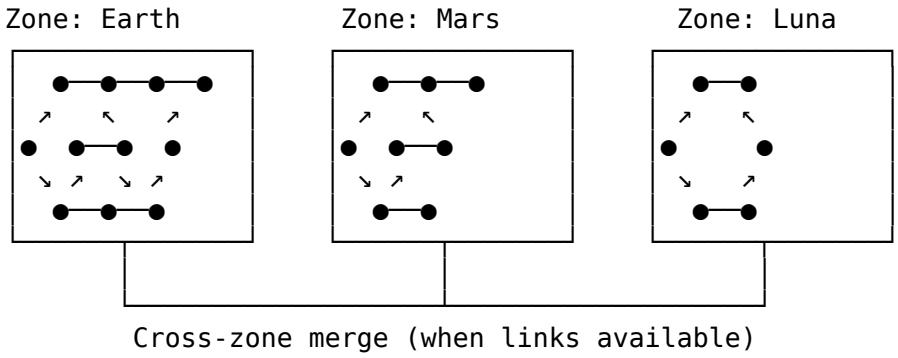
Blockchain (1D):

```
[Block N] → [Block N+1] → [Block N+2] → ...
(linear, sequential, one branch at a time)
```

Elara DAM (local view, 2D):



Elara DAM (global view, multi-zone):



Each validation record references one or more previous records (its "parents"). Within and across zones:

- Multiple branches grow in parallel (no bottleneck)
- Branches merge naturally when nodes synchronize
- Partitioned zones develop independent branches that reconnect when communication resumes
- The full history is preserved — nothing is pruned, rewritten, or discarded
- Classification projections ensure observers see only what their clearance permits

Validation Record Structure

```
ValidationRecord {
    id:             UUID v7 (time-ordered)
    version:        protocol version
    content_hash:   SHA3-256(content)
    creator:        public key (CRYSTALS-Dilithium)
    timestamp:      local ISO-8601 + vector clock position
    parents:        [record_id, ...] (DAG references)
    zone:           optional hierarchical zone path (wire format v3)
    classification: PUBLIC | PRIVATE | RESTRICTED | SOVEREIGN
    zk_proof:       optional zero-knowledge proof (for non-PUBLIC)
    metadata:       extensible key-value (content type, device info, etc.)
    signature:      ML-DSA-65 signature over all above fields
}
```

3.3.5 Dimensional Extensibility The DAM's dimensional count is not fixed at five. The architecture supports N structural dimensions and M operational layers, constrained by two requirements:

1. **Immutability.** Each coordinate must be deterministic and immutable at record creation. A property whose value changes over time — witness count, reputation score, confidence level — is an overlay metric, not a structural coordinate. Structural coordinates define where a record lives in the mesh; they cannot drift.
2. **Physical substrate mapping.** Each structural dimension must correspond to a physical property that, when implemented natively, eliminates a serialization tax (see Section 13).

A dimension without a hardware counterpart adds logical expressiveness but increases the serialization overhead that native hardware is designed to remove.

The current five dimensions — time, concurrency, zone topology, classification projection, and AI analysis — represent the hardware frontier of 2026. As substrate technology evolves, derived properties may be promoted to structural coordinates when they satisfy both requirements above.

First candidate: Lineage Depth (D). The number of causal hops from a zone's genesis record to a given record. Depth is currently derivable via DAG traversal at O(n) cost — walking the parent chain backward to the root. As a structural coordinate, it would reduce lineage queries and trust scoring to O(1) address lookups.

Depth satisfies the extensibility criteria partially:

- **Immutable at creation.** A record's depth is determined by its parents at the moment of creation and cannot change. ✓
- **Physically mappable.** 3D stacked memory (236+ layers in production, 2023) provides a substrate where records at the same depth occupy the same physical layer, making depth-based queries a layer selection rather than a graph traversal. ✓
- **Universally useful.** Forensics, trust scoring, provenance chain analysis, and regulatory audit all benefit from first-class depth addressing. ✓
- **Open problem: multi-parent ambiguity.** In a DAG with multiple parents at different depths, a record's depth requires a resolution rule — maximum parent depth + 1 (longest chain), minimum + 1 (shortest path), or zone-relative computation. This ambiguity does not exist in the current five coordinates, where each value is unambiguously determined at creation. ✗

The protocol reserves Depth as a future structural dimension, pending formal resolution of the multi-parent disambiguation rule and validation against production workloads.

Expansion axes. The architecture supports expansion in two distinct ways:

- **New structural dimensions** require hardware justification, immutability, and orthogonality to existing coordinates. They extend the address tuple and the wire format.
- **New operational layers** are cheaper to add — they project over the existing mesh without modifying the address space. Reputation scoring, semantic classification, and economic valuation are natural candidates for future layers that do not require structural promotion.

The 5-tuple addressing scheme (Section 5.3 of the Hardware Whitepaper) is designed to accommodate additional coordinates without breaking the wire format — the metadata field in ValidationRecord provides the extension point for dimensions that are structurally validated but not yet promoted to native coordinates.

3.4 Node Types

Type	Hardware Tier	Role	Stores Records?	Consensus?	Example
Leaf node	Tier 1 (MCU)	Creates records, signs directly (Profile A/B) or delegates to gateway (Profile C)	No (submits to nearest relay/witness)	No	IoT sensor, $4 ESP32
Relay node	Tier 2 (phone/laptop)	Light client, propagates records, verifies Merkle proofs, holds own records	Own records only	No	Phone app, laptop
Witness node	Tier 3 (VPS)	Attests to epoch seals, validates records, holds subscribed zones on disk	Yes (subscribed zones)	Yes	Cloud VM, mini PC
Anchor node	Tier 3-4 (high-trust)	Witness + epoch seal proposer for assigned zones (VRF-selected)	Yes	Yes + proposes epochs	Hardened VPS, data center

Type	Hardware Tier	Role	Stores Records?	Consensus?	Example
Archive node	Tier 4 (data center)	Full history for a region/industry, serves Merkle proofs and historical queries	Yes (broad/full)	Yes	Cold storage facility
Gateway node	Tier 1-2 (IoT hub)	Bridges constrained devices (Profile C) to the network via delegation	Delegated	Delegated	Home hub, factory edge

Any node can serve multiple roles simultaneously. A laptop can be a relay and witness. A $4 microcontroller can only be a leaf, but that is sufficient for its purpose. Gateway nodes enable Profile C devices (too constrained for PQC key generation) to participate via delegated signing.

Node role mapping: Light Node = relay. Full Node = witness/anchor. Storage Node = archive. Node incentive structures are described in Section 9.

Node Types vs. Module Tiers The Elara Core reference implementation (v0.15.0) introduces a **module tier system** that controls what cognitive capabilities a node activates. This is orthogonal to node types: the tier controls what a node *thinks*, the node type controls what it *does on the network*.

Core Module Tier	Capability Level	Typical Node Type	What It Unlocks
Tier 0: VALIDATE	Cryptographic signing only	Leaf node (Profile C gateway, IoT)	Layer 1 operations: hash, sign, verify, DAG append
Tier 1: REMEMBER	Memory and persistence	Leaf, Relay	+ episodic memory, corrections, handoff, basic recall

Core Module Tier	Capability Level	Typical Node Type	What It Unlocks
Tier 2: THINK	Reasoning and analysis	Witness, Relay	+ cognitive models, predictions, principles, reasoning trails, Cognitive Continuity Chain
Tier 3: CONNECT	Full network cognition	Anchor, Bridge	+ network tools, Layer 3 AI, dream/overnight processing, full inter-node cognitive exchange

A Tier 0 node on a $4 ESP32 is a leaf that validates sensor readings. A Tier 3 node on a server is an anchor that runs full cognitive analysis and generates Cognitive Continuity Chain snapshots. The same protocol, the same DAM, the same cryptographic proof — scaled by hardware capability rather than by price tier or permission level.

The tier system is **self-assessed** — each node selects its tier based on available hardware resources (RAM, storage, compute). There is no central authority assigning tiers, and a node can change its tier at any time by upgrading or downgrading its hardware. (See Elara Core Whitepaper v1.5.1, Section 13.3 for tier implementation details.)

3.5 Minimum Viable Validation: One Device, Zero Network

Before the interplanetary architecture, the multi-zone consensus, and the AI intelligence layer — there is the simplest possible use case. It matters more than the rest.

A vibration sensor on a factory bearing generates a reading. A military device validates a firmware update. A satellite records a telemetry measurement 14 light-minutes from Earth.

No internet. No central server. No cloud dependency. No beat. No fee.

What happens:

```
Step 1: Device generates a cryptographic keypair (once, on first boot)
        → Takes 200 milliseconds. No network needed.
        → This keypair IS the device's identity on the protocol. Permanently.

Step 2: The device produces data — a sensor reading, a firmware hash,
        a telemetry measurement, a decision log entry.

Step 3: The device validates.
        → Computes SHA3-256 hash of the data
        → Signs the hash with its private key (CRYSTALS-Dilithium)
        → Creates a ValidationRecord with timestamp and device public key
        → Appends to local DAG
        → Done. Sub-second on commodity hardware.

Step 4: The data now has cryptographic provenance.
        → No network was involved.
        → No authority approved it.
        → No fee was charged.
        → No beat was consumed.
     → The math is the proof — locally. Network witnesses add attestation over time.
```

When connectivity becomes available — seconds later on a factory LAN, hours later on a satellite downlink, days later on a field-deployed military device:

```
Step 5: The device syncs with its network (private or public).
        → The ValidationRecord propagates to peers.
        → Witnesses accumulate. Trust score grows.
        → The data is now attested beyond the originating device.
```

If a sensor reading is later disputed — was the bearing vibration within spec? — the DAM provides cryptographic evidence: the original ValidationRecord, signed with the device's key, timestamped at the moment of measurement, with an unbroken causal chain to every subsequent reading. The evidence is tamper-evident by construction.

The same protocol scales down to one person.

A teenager in rural Kenya writes a poem on a $30 Android phone. No internet, no cell signal, no government registry, no lawyer, no $3,500 patent fee, no specific alphabet required. She taps "Validate" — the phone hashes the poem, signs it with her private key, appends it to her local DAG. The poem is cryptographically hers. Later, when she walks past a Wi-Fi hotspot, the record syncs and witnesses accumulate. If someone in New York publishes the same poem next month and claims authorship, the DAM provides cryptographic evidence: her validation record, signed with her key, timestamped weeks earlier. The cryptography does not care about geography, wealth, or language.

This is the protocol's foundational principle. Layer 1 does not distinguish between a $4 microcontroller and a datacenter, between a defense installation and a teenager's phone. The same architecture that validates a satellite's telemetry on Mars validates a poem written on a phone in Nairobi. The same cryptographic proof that protects a corporation's trade secrets protects a teenager's creative work. A creation is a creation. An execution is an execution.

Every technical decision in this paper — the DAM structure, the post-quantum cryptography, the zero-knowledge proofs, the partition tolerance — serves this use case first. If it does not work for one device with zero network, it does not work.

The minimum viable network is not a cluster. It is not a quorum. It is one device — on a factory floor, in orbit, or in a teenager's hand — proving that something was created or measured by someone or something, at some moment, and that this fact cannot be taken away.

3.6 Industrial Scale Deployment: From One Phone to One Million Sensors

Section 3.5 shows the protocol at its smallest: one teenager, one phone, one poem. This section shows the same architecture at its largest: a factory with a million sensors generating billions of readings per day. The same cryptographic proof covers both.

Scenario: Samsung semiconductor fabrication plant

A single fabrication facility operates 10,000 sensors — vibration monitors on bearings, temperature probes in clean rooms, pressure gauges on gas lines, optical sensors on wafer alignments. Each sensor generates one reading per second.

The numbers:

```
10,000 sensors × 1 reading/second × 86,400 seconds/day = 864,000,000 readings/day

Without batch signing (individual Dilithium3 signatures):
  864M readings × 3,309 bytes per signature = 2.85 TB/day in signatures alone

With Profile C batch signing (1,000 readings per batch):
  864M readings ÷ 1,000 per batch = 864,000 batch signatures/day
  864K batches × 3,309 bytes = 2.85 GB/day in signatures

Compression ratio: 1,000:1
```

Architecture:

```
┌──────────────────────────────────────────────────────┐
│  Factory Floor: 10,000 sensors (Tier 0, Profile C)   │
│  ├── Vibration sensors → HMAC → Gateway 1            │
│  ├── Temperature probes → HMAC → Gateway 2           │
│  ├── Pressure gauges → HMAC → Gateway 3              │
│  └── Optical sensors → HMAC → Gateway 4              │
│                                                      │
│  Gateways (Tier 0, Profile A): batch-sign 1,000 readings │
│  ├── 4 gateways → 864K batch signatures/day          │
│  └── Each batch = one ValidationRecord on the DAM    │
│                                                      │
│  Factory AI (Tier 2): pattern analysis, anomaly detection │
│  ├── Runs Cognitive Continuity Chain (~30 checkpoints/day) │
│  ├── Generates reasoning trails for quality decisions │
│  └── Local DAG: complete factory history             │
│                                                      │
│  Enterprise Mesh (Tier 3): cross-factory coordination │
│  ├── 100 factories → private network (Section 10.6)  │
│  ├── Inter-factory anomaly correlation               │
│  └── Optional: Network Publication (Section 10.6.3)  │
└──────────────────────────────────────────────────────┘
```

Why this works:

1. **Sensors don't run PQC.** A $4 vibration sensor sends HMAC-authenticated readings to a trusted gateway over CAN bus. The gateway does the cryptography. Profile C (Section 4.6) was designed for exactly this.
2. **Batch signing collapses overhead by 1,000x.** Instead of 2.85 TB of signatures, the factory generates 2.85 GB — manageable on commodity hardware.
3. **The Cognitive Continuity Chain runs at the factory AI level, not the sensor level.**

Sensors don't think. The factory AI thinks — it analyzes patterns, makes predictions, detects anomalies. The CCC proves that this cognitive process was unbroken: no gaps, no tampering, no silent model replacement. A Tier 2 node generates ~30 cognitive checkpoints per day, each ~3-4 KB. Negligible.

4. **The private network is free.** The entire factory operates as a private network (Section 10.6). No beats, no witnesses, no Layer 2 fees. Layer 1 is always free.
5. **The same proof.** The ValidationRecord for a bearing vibration reading and the ValidationRecord for a teenager's poem have identical cryptographic structure. The same Dilithium3 signature. The same SHA3-256 content hash. The same DAG references. The protocol does not have an "enterprise mode" and a "personal mode" — there is one mode, at every scale.

Scaling to the enterprise:

Across 100 Samsung factories worldwide:

```
100 factories × 864M readings/day = 86.4 billion readings/day
100 factories × 864K batches/day = 86.4M batch signatures/day
Storage: 86.4M × 3,309 bytes = ~285 GB/day in signatures
```

285 GB/day of cryptographic signatures across 100 factories, validating 86.4 billion sensor readings. On commodity hardware. With post-quantum security. With no blockchain fees.

If Samsung later decides to publish its validation history to the public network — a Network Publication event (Section 10.6.3) — the published records integrate into the global DAM with the same trust scoring that applies to every other record. The bearing vibration readings from a Pyeongtaek fabrication line sit alongside poems from Nairobi in the same data structure, with the same cryptographic guarantees, distinguished only by their content hashes and classification levels. *(Implementation-status note: Network Publication is design-stage — NETWORK_PUBLISH is disabled in the current runtime; see §10.6.3. The "same trust scoring" described here is the original per-record design; it is being reframed to inert-import, which confers zero native standing on imported records.)*

3.7 Genesis and Network Bootstrap

A validation network that proves *who did what, when* must also be able to prove *how it began*. Genesis is the one state transition not justified by prior records — it is justified by a documented ceremony — and everything after it must follow from the protocol's ordinary rules. This section specifies how a blank-slate Elara network reaches normal operation: sealing, attesting, finalizing, and rewarding, with no special-case trust beyond the genesis state itself.

None of the mechanisms below is theoretical. Each exists because a clean-slate launch rehearsal of the reference implementation hit the corresponding failure live; the walls were removed one at a time, and the final rehearsal took a fresh network from empty data directories to self-funding finality in minutes, unattended.

The bootstrap circularity. In any stake-weighted BFT system, finality requires existing stake: a record settles when attestations represent a sufficient share of eligible stake. But stake operations are themselves records, and they take effect on *finalization*, not on creation — applying balance-mutating operations before finality would let any node manufacture state. On a blank network this is a deadlock: the first stake cannot finalize because no stake exists to attest it, and no stake will ever exist because the first stake cannot finalize. A network in this state can mint its genesis allocation and produce empty epoch seals indefinitely — it looks alive — but it can never finalize anything.

Genesis validator set. The resolution is the standard proof-of-stake answer, adapted to the DAM: the genesis configuration carries an explicit validator set — (identity, stake) pairs — ap-

plied directly into the ledger at the genesis baseline, before any record flows. Genesis stake is minted-into-stake: the network's first trust is declared by the ceremony, not earned through the record path, because there is no record path yet. The application is deterministic and idempotent, and every node applies the identical set from its genesis configuration: two nodes with the same configuration produce byte-identical baseline state roots, and a node that applies a different set has, by definition, forked at birth. With at least two genesis validators staked, the first ordinary stake record has a non-creator attestor pool, finality closes, and from that point forward stake changes flow exclusively through ordinary finalized records.

The first seal. Epoch sealing discovers work by scanning recent records and existing seal chains. A blank network has neither: genesis-era records are already older than the first seal window by the time the seal loop runs, and no seal chain exists to anchor discovery. The protocol therefore includes a genesis-bootstrap discovery path — a bounded, capped scan that seeds zone discovery from pre-window records while no epoch has ever been sealed — which becomes inert the moment the first seal lands. Without it, a fresh network idles at epoch zero indefinitely while reporting itself healthy.

Exactly-once finalization effects. Gossip redelivers; that is its job. Attestations for an already-settled record arrive again whenever peers re-synchronize, and a naive implementation re-fires finalization side effects — witness rewards, reputation credit, downstream events — on every redelivery, minting duplicate rewards from replayed traffic. The protocol binds all finalization effects to a single durable edge: the record's transition from unfinalized to finalized in the persistent finalization index. However many times consensus re-derives a record's settlement, the effects fire exactly once, from whichever code path crosses that edge first.

Admission at hour zero. Sybil resistance includes an identity-age gate: peers reject direct attestation pushes from identities younger than a configured threshold, because cheap fresh identities are the raw material of attestation spam. At genesis, *every* identity is younger than the threshold — including the validators the ceremony just declared. Config-pinned genesis validators are therefore exempt from the age gate. This is not a weakening: they are the trust root the gate exists to protect. Every later identity ages in normally.

Genesis-era record visibility. Records created in the network's first minutes face a structural hazard: peers that boot moments later begin their synchronization cursors *after* those records, and a record that arrives before the ledger state it depends on can be rejected and that rejection cached as permanent. The protocol requires two complementary defenses: witnessing must sweep old unfinalized records — bounded, oldest-first — so pre-cursor records remain witnessable; and rejection caching must distinguish permanent invalidity (malformed, cryptographically unsound) from retryable prematurity (dependencies not yet applied), parking the latter for re-fetch rather than poisoning them. Without both, the network's own genesis infrastructure — its pool funding, its earliest stakes — can become permanently invisible to nodes that booted seconds too late.

Economic ignition. Witness rewards draw from the conservation pool (Section 9), and the genesis allocation seeds that pool directly, so the first finalized record can pay its witnesses. Later top-ups, where policy allows them, are ordinary authority-signed records subject to the same finalization rules as everything else.

The launch sequence. Assembled, a genesis ceremony is: (1) generate the genesis authority and validator identities; (2) fix the genesis configuration — authority, validator set, allocations, conservation-pool seed — and distribute it byte-identical to every launch node; (3) boot. The network mints its allocation, applies validator stakes at the baseline, discovers its first zones through the bootstrap path, seals, attests through its age-exempt validators, finalizes, and pays its first witness rewards from the seeded pool. After the first seal, every bootstrap mechanism is inert and the network runs on the ordinary rules alone.

Status note (honest-claims rule): the mechanisms in this section are implemented in the Rust runtime and were validated by repeated blank-slate launch rehearsals on a three-node devel-

opment network (June 2026) — fresh network to self-funding finality in under four minutes, all nodes consistent. A mainnet-scale genesis (many validators, public ceremony) follows the same sequence but has not yet been exercised at that scale.

---

4. Post-Quantum Cryptography

4.1 The Quantum Timeline

Quantum computing capable of breaking current cryptographic standards (RSA-2048, ECDSA, EdDSA) is widely believed to arrive within 10–20 years, though the exact timeline remains highly uncertain and subject to ongoing debate. NIST finalized its first post-quantum standards in August 2024, signaling that the migration is no longer theoretical.

The threat model is "harvest now, decrypt later": adversaries can collect encrypted data and signed records today, then break the cryptography retroactively when quantum computers become available. For a protocol designed to produce proofs that outlive the protocol itself, this is not a future concern — it is a present design requirement.

4.2 Cryptographic Primitives

The Elara Protocol uses three NIST-standardized post-quantum algorithms:

CRYSTALS-Dilithium (ML-DSA, FIPS 204) — Digital signatures - Used for: signing validation records, authenticating node identity - Security basis: Module Lattice-Based Digital Signature (ML-DSA-65, NIST Security Level 3) - Signature size: ~3.3 KB (3,309 bytes — the FIPS 204 final ML-DSA-65 value the implementation ships and enforces; the legacy liboqs Round-3 length of 3,293 bytes is rejected) - Signing speed: ~1 ms on the shipped pure-Rust implementation (measured in-tree, desktop-class release build; optimized AVX2 C implementations reach ~0.3 ms) - Selected for: balance of security, performance, and signature size

CRYSTALS-Kyber (FIPS 203) — Key encapsulation - Used for: establishing encrypted channels between nodes, key exchange - Security basis: Module Learning With Errors (ML-KEM) - Ciphertext size: ~1.1 KB (Kyber768, NIST Security Level 3) - Selected for: efficiency in key exchange, well-studied security proofs

SPHINCS+ (FIPS 205) — Hash-based signatures - Used for: long-term anchor signatures, seed vault attestation, root of trust - Security basis: stateless hash-based signatures (SLH-DSA) - Signature size: ~35 KB (SPHINCS+-SHA2-192f) - Signing speed: ~130 ms on the shipped pure-Rust implementation (measured in-tree; verification ~7 ms) — two orders slower than Dilithium, acceptable for its low-frequency anchor / root-of-trust role - Selected for: conservative security assumptions — if lattice-based cryptography fails, hash-based signatures remain secure under minimal assumptions

4.3 Dual-Signature Strategy

Critical validation records (anchor attestations, identity registrations, governance votes) carry dual signatures:

1. **Primary:** CRYSTALS-Dilithium (fast, compact)
2. **Secondary:** SPHINCS+ (conservative, hash-based)

This provides defense-in-depth against cryptographic breakthroughs. Dilithium (lattice-based) and SPHINCS+ (hash-based) rely on fundamentally different mathematical assumptions — lattice problems and hash function preimage resistance, respectively. Breaking one does not weaken the other. If lattice-based cryptography falls to an unforeseen advance, the hash-based

signature remains valid; if hash functions are weakened, the lattice signature still holds. Both must be broken simultaneously to forge a dual-signed record. The protocol's trust model degrades gracefully rather than failing catastrophically.

4.4 Algorithm Agility

The Elara Protocol does not hardcode cryptographic algorithms. Every signature and key exchange specifies its algorithm identifier:

```
signature {
    algorithm: "dilithium3"
    value: <bytes>
}
```

When new algorithms are standardized or existing ones are deprecated, the protocol can migrate without structural changes. Old records remain valid under their original algorithms; new records use updated algorithms. The DAG preserves the full cryptographic history.

This agility is a core survival mechanism. A protocol that hardcodes today's best cryptography is guaranteed to become insecure. A protocol that specifies algorithms by identifier can evolve with the field.

4.5 Comparison with Existing Systems

System Type	Typical Signature Algorithm	Quantum-Safe	Migration Status
PoW blockchains	ECDSA (secp256k1)	No	Not started
Smart contract platforms	ECDSA (secp256k1)	No	Research phase
High-throughput ledgers	Ed25519	No	Not started
DAG-based systems	Ed25519 / hash-based OTS	Partial	In progress
Elara Protocol	Dilithium (all profiles) + SPHINCS+ (Profile A)	**Yes**	**Native**

The Elara Protocol's signature layer is post-quantum from genesis — there is no legacy migration burden. However, algorithm agility (Section 4.4) ensures the protocol can adopt future PQC standards as the field evolves; "quantum-safe at launch" is not the same as "cryptographically final." (The ZKP layer in Phase 1 ships SHA3-256 hiding commitments — no classical curves in shipped code; the specified Groth16/BN254 construction is design-stage and wire-rejected until it lands. See Section 11.26 for the quantum migration path.)

4.6 PQC Size Penalty and Constrained Device Strategy

Post-quantum cryptography provides stronger security at a measurable cost in size:

Algorithm	Key Size	Signature Size	Classical Equivalent
CRYSTALS-Dilithium3	1,952 bytes	3,309 bytes†	ECDSA: 33 + 72 bytes
SPHINCS+-SHA2-192f	48 bytes	35,664 bytes	Ed25519: 32 + 64 bytes
CRYSTALS-Kyber768	1,184 bytes	1,088 bytes	X25519: 32 bytes

†FIPS 204 ML-DSA-65 standard value (3,309 bytes). Earlier liboqs Round 3 implementations used 3,293 bytes.

Dilithium signatures are **~46x larger** than ECDSA signatures (3,309 vs ~72 bytes). For a datacenter or laptop, this is negligible. For an ESP32 sending thousands of signed readings over LoRa (max payload ~242 bytes), it is prohibitive.

Solution: Tiered Cryptographic Profiles

The protocol defines three cryptographic profiles that devices select based on their capabilities:

Profile A: Full PQC (default) - Dilithium3 signatures, Kyber768 key exchange, SPHINCS+ for anchoring - For: servers, laptops, phones, gateways - Signature overhead: ~3.3 KB per record

Profile B: Compact PQC - Dilithium3 (same parameter set as Profile A: 3,309 byte signatures, NIST Level 3) - No dual signatures (Dilithium only, no SPHINCS+) - For: Raspberry Pi, industrial controllers, modern IoT gateways - Signature overhead: ~3.3 KB per record (identical to Profile A primary signature)

Profile C: Gateway-Delegated Signing - Constrained device (ESP32) sends unsigned readings to a trusted gateway via secure local channel (BLE, CAN, wired) - Gateway batches readings and signs the batch with Profile A or B - Device authenticates to gateway using lightweight symmetric key (pre-shared, established at provisioning) - For: $4 microcontrollers, ultra-low-power sensors - Per-reading overhead on device: ~32 bytes (HMAC) - Per-batch overhead on network: ~3.3 KB (one Dilithium signature per batch of hundreds/thousands of readings)

Profile C is a pragmatic compromise: the constrained device cannot run PQC itself, but its readings are still validated on the DAM through a trusted gateway. The trust boundary shifts from the device to the gateway — acceptable for IoT deployments where the gateway is physically secured alongside the sensors.

All three profiles produce validation records that are interoperable on the DAM. The profile is specified in the record metadata, so verifiers know which security level applies.

Profile B Security Boundary (v0.7.1 clarification). Profile B is Profile A minus SPHINCS+ — it uses the same ML-DSA-65 (Dilithium3, NIST Level 3) primary signature but omits the SLH-DSA secondary signature. Under the quantum adversary model defined in the MESH-BFT companion paper (Theorem 2), Profile B records become forgeable if ML-DSA-65 is broken by a quantum adversary — unlike Profile A records which remain secure via the independent SPHINCS+ signature. Consequently, Profile B identities are treated as lower-trust for consensus purposes. The protocol recommends: (a) transfer limits for Profile B identities (e.g., max 1,000 beats per transaction), (b) settlement requires a minimum fraction of Profile A attestations (e.g., ≥50% of attesting stake from Profile A witnesses), and (c) high-value operations (staking >10K beats, governance votes) require Profile A identity.

Future PQC Size Reduction: NIST Additional Signatures Project

The PQC size penalty described above reflects the first generation of NIST-standardized post-quantum signatures. This is not the final generation. In November 2024, NIST announced the **Post-Quantum Cryptography: Additional Digital Signature Schemes** project, accepting ~50 submissions for evaluation. Several candidates offer dramatically smaller signatures than Dilithium:

Candidate	Signature Size	vs. Dilithium3 (3,309 B)	Basis
SQIsign	~204 bytes	**16x smaller**	Supersingular isogenies
HAWK	~555 bytes	**6x smaller**	Lattice (NTRU)
UOV (variants)	~96–128 bytes	**25-34x smaller**	Multivariate

These are candidates, not standards — NIST evaluation will take years, with standardization likely in 2027-2028 at the earliest. Signing performance varies (SQIsign is significantly slower than Dilithium), and security assumptions for some candidates are less studied.

The Elara Protocol's **algorithm agility** (Section 4.4) means that adoption of compact PQC signatures is a configuration change, not a protocol redesign. When NIST standardizes a compact alternative:

1. New algorithm identifier added to the protocol via governance vote
2. Transition period: both Dilithium and the new algorithm accepted
3. New records use the compact algorithm; old records remain valid under Dilithium
4. Profile B and C devices benefit most — a 200-byte signature eliminates the size penalty that drove the Profile C delegation model

The current 46x size penalty over classical signatures is a first-generation cost, not a permanent constraint. The protocol is designed to absorb future improvements without structural change.

---

5. Zero-Knowledge Validation

*Implementation-status note (honest-claims rule): throughout Section 5, "zero-knowledge proof" / "ZKP" describes the **specified design**. Phase 1 — the current Rust runtime — implements the PRIVATE and RESTRICTED privacy properties with **SHA3-256 hiding commitments** (verified by hash recomputation), not succinct zero-knowledge proofs; the Groth16/BN254 and zk-STARK constructions are design-stage and are rejected fail-closed on the wire until a prover/verifier lands. Section 5.3 gives the full status and Section 14.3 the migration path. Read the present-tense ZKP language below as the target architecture, not a description of shipped behavior.*

5.1 The Privacy Paradox

Validation and privacy are traditionally in tension. To prove you created something, you must reveal what you created. This is acceptable for open-source code or public art, but unacceptable for:

- Trade secrets under development
- Medical data from IoT health devices
- Military or intelligence sensor readings
- Unreleased creative work
- Corporate R&D
- Personal journals or private communications

The Elara Protocol resolves this through zero-knowledge proofs (ZKPs): cryptographic constructions that prove a statement is true without revealing the underlying data.

5.2 Classification Levels

Every validation record carries a classification level that determines what the network can see:

PUBLIC — Full content hash visible. Anyone can verify the exact content. Default for open-source code, published work, public sensor data.

PRIVATE — Content hash wrapped in a zero-knowledge proof. The network validates that: - A valid content hash exists - It was signed by a valid keypair - The timestamp is consistent with the DAG - No conflicting claim exists

...without learning what the content is. The creator can selectively reveal the content later (e.g., in a legal dispute or patent filing) by providing the pre-image.

RESTRICTED — Key-group access. The content hash is encrypted to a set of public keys. Only designated parties can verify the content. The network validates the structural integrity of the record without accessing the encrypted payload.

SOVEREIGN — Maximum privacy. Multi-key authorization required for any access. Time-locked release optional. Validator nodes process the mathematical proof without any visibility into the content, the creator's identity (which is itself wrapped in a ZKP), or the classification metadata.

5.3 ZKP Construction

*Implementation-status note (honest-claims rule): Phase 1 — the current Rust runtime — ships **SHA3-256 hash-commitment** schemes for the PRIVATE and RESTRICTED privacy properties (BalanceRange, MetadataProperty, ContentCommitment). These are hiding commitments verified by hash recomputation, **not** succinct zero-knowledge proofs. The Groth16/BN254 zk-SNARK construction described in this section, and the zk-STARK construction for SOVEREIGN, are **specified design-stage targets that are not yet implemented**: the runtime reserves the Groth16 wire version (0x02) and currently **rejects it fail-closed** rather than accept an unverifiable proof. This section documents the specified design; Section 14.3 gives the post-quantum migration timeline.*

The Elara Protocol specifies Groth16 zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) on the BN254 curve for PRIVATE and RESTRICTED classifications. Three circuit types are specified:

5.3.1 Circuit Types

Type	Public Inputs	Private Witness	Proves
BalanceRange	threshold	balance, excess	"I have ≥ threshold beats" without revealing exact balance
MetadataProperty	key_hash, commitment	value_hash, salt	"Metadata key has a specific value" without revealing it
ContentCommitment	commitment	content_hash_fr, blinding_factor	"I know content behind this commitment"

BalanceRange circuit (R1CS): Proves `balance ≥ threshold` by decomposing `excess = balance - threshold` into 64 bits, constraining all higher bits to zero. This implicitly constrains excess to `[0, 2^64)`.

MetadataProperty circuit (R1CS): Algebraic commitment `commitment = key_hash + value_hash × salt`. Demonstrates the R1CS structure for property commitments.

ContentCommitment circuit (R1CS with Poseidon hash): The primary ZK-friendly circuit. Uses Poseidon hash inside R1CS constraints:

```
Public inputs:  commitment
Private witness: content_hash_fr, blinding_factor
Constraint:     commitment == Poseidon(content_hash_fr, blinding_factor)
```

Poseidon parameters for BN254: rate=2, capacity=1, alpha=17, 8 full rounds + 57 partial rounds (standard security parameters from the Poseidon specification).

5.3.2 Proof Format The specified Groth16 proof format on BN254 is 2 G1 points + 1 G2 point = ~192 bytes compressed, verifiable in ~1ms on commodity hardware. Wire format: `[version:0x02][proof_type:u8][proof_bytes][public_inputs_count:u8][public_inputs]`. The version byte `0x02` is **reserved** for Groth16 proofs and distinguishes them from the SHA3 commitments (`0x01`/`0x03`) that Phase 1 actually ships; until a prover/verifier lands, the runtime rejects `0x02` fail-closed.

5.3.3 PQC + ZK Orthogonal Composition ZK proofs and post-quantum signatures serve orthogonal purposes and compose naturally:

- **ZK proof = privacy** — hides content while proving properties about it
- **PQC signature = authenticity** — proves the record was created by a specific identity

Dilithium3 signature verification inside R1CS is infeasible: lattice operations produce millions of constraints, making proof generation take minutes and CRS generation hours. The pragmatic split keeps both properties without combining them inside a single circuit. A PRIVATE record carries both: a ZK proof (anyone can verify the property without learning the content) and a PQC signature (anyone can verify who created it).

5.3.4 Trusted Setup The CRS (Common Reference String) is generated deterministically from a seed for testnet:

```
seed = SHA3-256("elara-protocol-groth16-crs-v2")
```

For mainnet, a multi-party computation (MPC) ceremony will produce the CRS. Security requires ≥1 honest participant who destroys their randomness. See Section 11.11 for the ceremony process and migration path to transparent proof systems (STARKs).

5.3.5 Gossip Verification ZK proof verification in the gossip layer uses dual-dispatch: the first byte of `zk_proof` determines the format. SHA3 commitments (`0x01`/`0x03`) are verified via hash recomputation — this is the Phase 1 path. The Groth16 path (version `0x02`) is specified to verify against the CRS loaded at node startup, but until the prover/verifier is implemented the runtime rejects `0x02` fail-closed rather than accept an unverified proof. WASM browser nodes relay proofs without verification — native witness nodes verify before consensus acceptance.

(Note: the BN254 curve provides ~100-bit security against classical attacks and is NOT post-quantum. See Section 14.3 for the migration timeline to post-quantum ZKP constructions.)

For SOVEREIGN classification, the protocol specifies zk-STARKs (Scalable Transparent Arguments of Knowledge) as a future extension — larger proofs (~100 KB) but no trusted setup required.

5.4 Selective Disclosure

A creator who validates work as PRIVATE can later:

1. **Reveal to specific parties** — provide the content hash and blinding factor to a court, patent office, or business partner. They can verify it matches the on-chain commitment.

2. **Upgrade to PUBLIC** — publish the content hash, making the validation fully transparent. The DAG timestamp proves the work existed at the original validation time.
3. **Maintain privacy indefinitely** — the zero-knowledge proof is sufficient for priority claims without ever revealing the content.

This streamlines a workflow that was previously impractical at scale: validate now, decide on disclosure later. A researcher can timestamp a discovery, continue working in private, and prove priority years later if needed.

---

6. Identity and Attribution

6.1 Self-Sovereign Identity

Every entity on the Elara Protocol generates its own cryptographic keypair. No central authority issues, approves, or revokes identities. This is a fundamental departure from traditional identity systems:

- **X.509 certificates** require a Certificate Authority hierarchy
- **DNS** requires ICANN and registrars
- **Government ID** requires citizenship and bureaucracy
- **OAuth** requires a third-party platform

An Elara identity requires only entropy and computation — available on any device, in any jurisdiction, on any planet.

Identity Structure

```
ElaraIdentity {
    public_key:     ML-DSA-65 (CRYSTALS-Dilithium) public key
    identity_hash:  SHA3-256(public_key)  // short identifier
    created:        timestamp
    entity_type:    HUMAN | AI | DEVICE | ORGANIZATION | COMPOSITE
    metadata:       optional, creator-defined, signed
    succession:     optional, designated heir public keys
    revocation:     self-revocation mechanism (signed revocation record)
}
```

The identity hash serves as a short, human-communicable identifier (similar to a fingerprint in PGP). The full public key is used for cryptographic operations.

6.2 Entity Types

The protocol recognizes five entity types, each with the same cryptographic standing:

HUMAN — Individual creators. One or more keypairs per person (separation of personal and professional identity is supported).

AI — Artificial intelligence systems. Each AI model instance generates its own keypair. This solves the AI attribution problem: when an AI generates content, its validation record includes the AI's identity, the model version, and (optionally) the prompt that triggered the generation.

DEVICE — IoT sensors, robots, vehicles, satellites. Capable devices (Raspberry Pi and above) generate a keypair at first boot and sign readings directly (Profile A/B). Constrained devices ($4 ESP32) authenticate to a local gateway via pre-shared symmetric key; the gateway signs batches on their behalf (Profile C, Section 4.6). Device identity persistence across physical

resets — including hardware-bound keys, organizational binding, and behavioral fingerprinting — is addressed in Section 11.33.

ORGANIZATION — Companies, research labs, governments. Organizational identities can designate authorized signers (multi-signature schemes).

COMPOSITE — Human-AI collaborations. A composite identity explicitly records the relationship: who prompted, who generated, who edited, who approved. This creates an unambiguous attribution chain that courts, patent offices, and licensing systems can interpret.

6.3 AI Attribution in Detail

The rise of AI-generated content has created an attribution crisis. Current systems cannot distinguish between: - Fully human-created work - AI-assisted work (human-directed, AI-generated) - Fully AI-generated work - AI-to-AI generated work (one model's output fed to another)

The Elara Protocol makes this explicit:

```
CollaborationRecord {
    work_hash:       content hash of the final output
    participants:    [
     { identity: human_key,  role: "prompter", contribution: "direction, editing" },
       { identity: ai_key,    role: "generator",  contribution: "initial draft",
          model: "claude-opus-4-6", version: "2026-02" }
    ]
    chain:           optional, references to intermediate outputs
    signed_by:       all participants
}
```

This record is immutable on the DAG. When a dispute arises about who created what, the cryptographic evidence is already in place.

6.4 Digital Succession

Physical death should not orphan digital work. The Elara Protocol supports:

- **Designated heirs** — public keys listed in the identity's succession field. Upon activation (by the heir providing a signed succession claim), the heir gains read access to the creator's PRIVATE and RESTRICTED work.
- **Time-locked release** — content becomes PUBLIC after a specified duration (e.g., 70 years, matching current copyright terms, or any custom period).
- **Dead man's switch** — if a node does not produce a heartbeat within a configured period, succession activates automatically.

The cryptography enforces the creator's wishes automatically, reducing dependence on legal proceedings for routine succession. Courts retain authority for contested cases, but the protocol provides the evidentiary foundation.

---

7. Interplanetary Operations

Note: This section describes theoretical protocol behavior under interplanetary communication constraints. The physics are documented; the protocol's response to them is specified but untested. No implementation or simulation has been conducted.

7.1 The Latency Problem

Communication delays in the solar system are not engineering failures — they are physics:

Route	One-Way Delay	Round-Trip
Earth surface	< 100 ms	< 200 ms
Earth-Moon	1.3 s	2.6 s
Earth-Mars (closest)	3 min	6 min
Earth-Mars (farthest)	22 min	44 min
Earth-Jupiter	33–54 min	66–108 min

No consensus mechanism that requires real-time communication can operate across these delays. A proof-of-work block time of ~10 minutes is barely acceptable for Earth-Moon; it is unusable for Earth-Mars.

7.2 Time Without Clocks

Speed-of-light delays make clock synchronization impractical across planetary distances. While relativistic time dilation between Earth and Mars is negligible at protocol timescales (microseconds), the fundamental issue is communication latency: a round-trip to Mars takes 6–44 minutes, making any synchronization-dependent protocol unworkable. The Elara Protocol does not rely on synchronized clocks. Instead, it uses:

Local timestamps — each node records its local time in its validation records. These are informational, not authoritative.

Vector clocks — logical counters that track causal ordering. If node A's validation references node B's validation, then A happened after B, regardless of wall-clock times. Vector clocks establish partial ordering without requiring clock synchronization.

Causal references — every validation record lists its parent records in the DAG. This creates an unambiguous "happened before" relationship. The DAG IS the clock.

Ordering rule: if A references B, A happened after B. If A and B have no causal relationship, they are concurrent — and the protocol does not force an artificial ordering.

Conflict resolution for concurrent records: When two causally unrelated validation records make incompatible claims (e.g., two entities claim authorship of semantically identical work within a short time window), the protocol preserves both records and applies the following resolution strategy:

1. **Preservation** — both records remain in the DAM permanently. Neither is deleted or hidden.
2. **Annotation** — a ConflictSet record is created linking the conflicting validations, with metadata including detection timestamp, similarity score, and conflict type.
3. **Evidence accumulation** — subsequent witnesses can attest to either or both records. The trust scores evolve independently.
4. **No automatic winner** — the protocol does not algorithmically determine which claim is "correct." Priority disputes involve human judgment, legal context, and evidence that a cryptographic protocol cannot evaluate.
5. **Query transparency** — any query that returns a conflicted record also returns the ConflictSet, ensuring consumers are aware of the dispute.

This design reflects a core principle: a validation protocol proves *possession at a point in time*, not *original creation*. Resolving authorship disputes is a human problem that the protocol supports with evidence but does not attempt to automate.

7.3 Partition Tolerance

Network partitions are not failures in the Elara Protocol — they are expected operating conditions. A partition occurs when:

- A solar flare disrupts Earth-Mars communication for days
- A submarine enters radio silence
- A remote sensor network loses satellite uplink
- Political conflict severs internet connectivity between regions
- A spacecraft is in transit between planets

During a partition:

1. Each zone continues operating independently
2. Local validations proceed normally (Layer 1 never requires connectivity)
3. Zone-internal consensus proceeds normally (Layer 2 within the zone)
4. The DAG branches — each zone grows its own branch

When a partition heals:

1. Zones exchange their branch tips
2. DAGs merge — all records from all zones are incorporated
3. Conflicting claims (if any) are preserved with annotations, not resolved by deletion
4. Witness counts update as the merged records propagate

This is fundamentally different from blockchain's approach to partitions, where a fork must be resolved by discarding one chain. The Elara DAG preserves both branches because both branches contain valid work by real entities.

7.4 Bandwidth Economics

Interplanetary communication is expensive. The Deep Space Network allocates bandwidth in kilobits per second. The Elara Protocol optimizes for minimal bandwidth:

Merkle roots — a single 32-byte hash summarizes millions of validation records. Zones exchange Merkle roots to detect divergence, then synchronize only the differences.

Delta sync — only new records since last synchronization are transmitted. A zone with 1 million records that has added 100 since last sync transmits only 100 records plus the updated Merkle root.

Priority sync — records are prioritized for transmission based on: - Classification level (SOVEREIGN records sync first) - Witness count (more-attested records are higher priority) - Age (recent records are more likely to be queried) - Creator priority (configurable per zone)

Bloom filters — probabilistic data structures (~10 bytes per element) that answer "does this record exist in your zone?" with a small false-positive rate. Bloom filter exchange enables efficient detection of missing records before full synchronization begins.

Compression — validation records use a compact binary encoding (not JSON or XML) with protocol-buffer-style varint encoding. A typical PUBLIC validation record is approximately 4–5 KB, dominated by the Dilithium3 signature (~3.3 KB).

7.5 Zone Architecture

7.5.1 Zone Model: Semantic Subscription with Stake-Gated Consensus Zones use hierarchical semantic paths that reflect real-world organizational and geographic structure:

```
"medical/eu/west/germany/bavaria"
"finance/global"
"iot/manufacturing/toyota/plant-7"
"personal/alice"
```

Record routing is determined by the record's `zone_refs` field — records go to zones based on their purpose and context. A medical record from a Bavarian hospital naturally belongs in `medical/eu/west/germany/bavaria`.

Zone subscription is voluntary — nodes choose which zones to store and process. A hospital server subscribes to `medical/eu/west`, a Toyota factory subscribes to `iot/manufacturing/toyota/*`, a phone subscribes only to its owner's personal zone. Nodes never store records for zones they don't subscribe to, enabling each node to hold a fraction of the global dataset.

Consensus participation is stake-gated — while any node can subscribe to store records, participating as a witness (attesting to epoch seals) requires: - Minimum 100 beats staked - PoW-verified identity with `min_pow_difficulty` (currently 20 bits) - Identity age ≥ 48 hours - Diversity check: no single entity or /24 IP subnet may control >33% of a zone's total staked weight

This separates Sybil resistance from zone assignment. The earlier hash-based model (`SHA3-256(public_key) mod NUM_ZONES`) provided Sybil-resistant zone assignment but created semantically meaningless groupings — a hospital might share a zone with unrelated IoT sensors. At quintillion-record scale, zone-scoped gossip REQUIRES semantic grouping to bound bandwidth. Sybil defense now operates at the witness admission layer through the existing mechanisms (PoW, stake, age, diversity scoring) rather than at the zone assignment layer.

Zone splitting: Zones split like biological cells when they grow too large. A zone exceeding a threshold of witnesses (>N) or records per epoch (>M) can split into sub-zones. Parent zone anchor nodes authorize the split through governance (Section 10.2). The hierarchical path naturally accommodates splitting: `medical/eu` can split into `medical/eu/west` and `medical/eu/east` without restructuring.

Wire format: Zone identifiers use variable-length hierarchical paths (wire format v3), replacing the previous `u8 zone_id` which limited the network to 256 zones.

7.5.2 Interplanetary Extensions (Aspirational) As humanity expands beyond Earth, the zone model naturally extends to geographic/planetary boundaries where communication latency makes cross-zone synchronization physically constrained:

```
Zone: Earth-Primary
├── Subzone: North America
├── Subzone: Europe
├── Subzone: Asia-Pacific
└── Subzone: Africa

Zone: Luna
├── Subzone: Artemis-Base
└── Subzone: FarSide-Observatory

Zone: Mars
├── Subzone: Ares-Colony-1
└── Subzone: Orbital-Relay

Zone: Deep-Space
└── Subzone: Voyager-Relay
```

In this model, zone assignment transitions from semantic to latency-based: nodes physically located on Mars would join the Mars zone because cross-planet consensus with Earth nodes is impractical at 3-22 minute one-way delays. The semantic subscription model described in Section 7.5.1 would operate *within* each planetary zone, providing intra-planet zone distribution. This architecture is specified but untested — it is included to demonstrate that the protocol's partition tolerance (Section 7.3) and asynchronous design accommodate interplanetary operation without structural changes.

7.5.3 Zone Lifecycle: Activity-Driven Split and Merge (v0.7.7)

> **PARTIALLY IMPLEMENTED — authority-gated, not autonomous.** The `TRANSITION_SPLIT`/`TRANSITION_MERGE` seal types, their structural validation, canonical signing encoding, and multi-anchor signature verification ship in the runtime (`zone_transition_seal.rs`); what is not yet live is autonomous account-rehoming across the split (see §14.8). Signed records carry no realm/zone binding in their signed bytes, so activating this mechanism safely requires a wire-version transition that adds that binding first — a decided one-way door, gated before any federation (see the MESH-BFT merge-semantics design notes in the runtime repository). What *is* shipped is the zone auto-scaling **decision engine** (activity tracking with hysteresis, split/merge recommendations, and dynamic zone-count routing — the `elara-zone-autoscaler` crate); the per-zone transition-seal protocol below is the design it will eventually drive. See also the limitation in §14.8.

The problem: Semantic zones are static in §7.5.1 — an operator picks `medical/eu/west` at creation time and the zone carries its full traffic forever. At 1M-zone scale, some zones will become hot (thousands of records per second) while others go cold (weeks between records). A single zone cannot scale past a single anchor's capacity; a cold zone wastes witness attention. The protocol needs a dispute-free mechanism to split hot zones and merge cold ones without interrupting finality.

Solution: Anchor-attested transition seals.

Each zone tracks its own activity: records-per-second over a rolling 1-hour window, average epoch inclusion count, and ledger account cardinality. Two configurable thresholds define the lifecycle:

- **Split threshold** (shipped default: average zone rate above **40 rec/sec** — 2× the 20 rec/sec per-zone target rate — held for 4 consecutive scaler ticks) — zone forks into two child zones.
- **Merge threshold** (shipped default: average zone rate below **2 rec/sec** — 0.1× the per-zone target — held for 4 consecutive scaler ticks; the design adds an account-cardinality floor not yet enforced) — sibling cold zones coalesce with their parent.

Split protocol:

1. The anchor observes its zone is above the split threshold. It computes a deterministic split key by SHA3-256 hashing the account_id of each account in the zone; accounts with `key[0] < 0x80` go to child A, the rest to child B. Account-hash redistribution is verifiable by anyone holding the zone's SMT (§ Stage 2, MESH-BFT).
2. The anchor proposes a **transition seal** — an epoch seal with the special `TRANSITION_SPLIT` flag, containing: parent zone path, child A path, child B path, account→zone assignment Merkle root, parent ledger state at split epoch, child A opening balance_total, child B opening balance_total.
3. The transition seal is attested by at least 2/3 of the parent zone's stake-weighted witnesses (standard epoch settlement rules). Once zone-settled, both child zones begin sealing epochs independently from the next epoch number.

4. Clients rebuild their account→zone routing by walking the transition seal chain. Pending cross-zone transfers (§11.22.2) targeting the parent zone are redirected to the correct child by account hash.

Merge protocol:

Symmetric to split. Two sibling zones under the merge threshold for 24 hours jointly propose a `TRANSITION_MERGE` seal signed by both anchors. The merge seal combines both ledger states; conservation is preserved because `merged.balance_total = sibling_A.balance_total + sibling_B.balance_total`.

Dispute-free property. Transition seals are epoch seals — they inherit the safety properties of §11.12. There is no separate voting mechanism, no new attack surface. A Byzantine anchor proposing a bogus split produces a seal that witnesses will reject (the account→zone Merkle root will not match their independent computation).

Implementation status: The activity-tracking and decision layer is shipped in the Elara Runtime (the `elara-zone-autoscaler` crate, re-exported through `auto_scale.rs`: hysteresis-gated split/merge recommendations and anchor-signed zone-count transition records with hash-based re-routing). The transition-seal protocol above — per-zone forking with account→zone assignment Merkle roots and balance-partition seals — is partially implemented: the runtime constructs, signs, and verifies `TRANSITION_SPLIT`/`TRANSITION_MERGE` seals (`zone_transition_seal.rs`, proposed via `POST /transitions/propose` and applied on the health tick), but the seals are authority-gated and autonomous account-rehoming across the split is not yet live (§14.8). Existing tests cover the decision engine and seal validation, not end-to-end seal-based migration.

---

8. IoT and Hardware Integration

8.1 The Physical-Digital Bridge

The Elara Protocol extends validation from digital content to physical-world data through IoT device integration. Every sensor, actuator, and controller can contribute to cryptographically validated records on the DAG — either by signing directly (Profile A/B) or through gateway-delegated signing (Profile C).

8.2 Device Capabilities

Minimum viable device: ESP32 ($4) — Profile C (Gateway-Delegated) - Authenticates to local gateway via pre-shared symmetric key (established at provisioning) - Sends readings over secure local channel (BLE, CAN, wired); gateway signs batches with PQC (Profile A/B) - Stores local readings in flash memory (circular buffer for constrained devices) - Communicates via Wi-Fi, BLE, LoRa, or CAN bus

Standard device: Raspberry Pi / industrial controller ($35–$200) - Full node capabilities including relay and witness roles - Local AI inference for anomaly detection (Layer 3) - Multiple communication interfaces

Gateway device: edge server ($500+) - Aggregates readings from leaf devices - Provides network connectivity for air-gapped devices - Runs full DAG synchronization

8.3 Supported Protocols

The Elara Protocol integrates with standard IoT communication protocols:

Protocol	Use Case	Integration
MQTT	Lightweight pub/sub messaging	Signed payloads as MQTT messages
CoAP	Constrained RESTful protocol	Validation records as CoAP resources
gRPC	High-performance RPC	Native Elara service definitions
HTTP/HTTPS	General web integration	REST API for validation records
BLE	Short-range device communication	Signed readings via BLE characteristics
CAN	Automotive/industrial bus	Signed frames on CAN bus
LoRa/LoRaWAN	Long-range, low-power	Compact validation records for LPWAN

8.4 Use Cases

Supply chain provenance — A coffee bean's journey from farm to cup: soil sensors sign moisture readings, GPS trackers sign location, temperature sensors sign cold-chain compliance. Every reading is on the DAG. A consumer scans a QR code and sees the cryptographically verified history of their product.

Autonomous vehicle accountability — Every decision by a self-driving car is signed by the vehicle's keypair: sensor readings, object detection results, path planning decisions, actuator commands. In an accident investigation, the DAG provides a millisecond-by-millisecond cryptographic audit trail.

Medical device integrity — A robotic surgical system signs every movement command. A patient's implanted sensor signs every heart rate reading. The data is immutable, attributed, and tamper-evident — meeting regulatory requirements while enabling the patient to own their own health data.

Smart grid validation — Solar panels sign generation readings, smart meters sign consumption readings, grid controllers sign distribution decisions. Energy trading between neighbors is validated on the DAG without a utility company intermediary.

8.5 Physical Object Authentication

Physical objects can be linked to DAG identities through:

- **NFC/RFID tags** — embedded chips that sign a challenge with their keypair, proving authenticity (luxury goods, pharmaceuticals, art)
- **PUF (Physical Unclonable Functions)** — semiconductor fingerprints that cannot be cloned, linked to DAG identities
- **Visual hashing** — high-resolution photographs of physical objects hashed and signed (art authentication, evidence collection)

8.6 Firmware Integrity and Device Attestation

Physical access to IoT deployments is a realistic attack vector. An adversary who captures a Profile C device can clone its firmware, implant a backdoor, and inject false readings through the gateway. The protocol mitigates this through:

Secure boot attestation — devices that support it produce a firmware hash at boot time. The security level varies by hardware: ARM TrustZone provides hardware-isolated key storage; ESP32-S3 secure boot uses eFuse-based verification of a signed bootloader in flash (not a full hardware-fused signing key). The gateway verifies the firmware hash before accepting readings. Devices with unexpected firmware hashes are quarantined. The attestation strength is reflected in the device's trust weight — hardware-isolated attestation carries more weight than flash-based verification.

Heartbeat anomaly detection — gateways monitor timing patterns, reading distributions, and communication behavior of leaf devices. A cloned device with modified firmware will exhibit measurable behavioral differences (response latency, reading noise profile, boot timing). The AI layer (Layer 3) can flag statistical anomalies for operator investigation.

Key rotation — pre-shared symmetric keys between leaf devices and gateways are rotated periodically (configurable, default: 30 days). A captured device with an extracted key has a limited exploitation window.

Physical tamper detection — for high-security deployments (medical, military, critical infrastructure), devices can include tamper-evident enclosures that zeroize key material upon case opening. This is not required by the protocol but is recommended for Profile C deployments with physical access risk.

For the broader threat of device wipes, identity resets, and reputation escape through hardware recycling — including hardware-bound identity persistence, organizational binding, behavioral fingerprinting, and decommissioning protocols — see Section 11.33.

---

9. Public Network Incentive Layer

> **Scope: Public permissionless network only.** This section describes the incentive mechanism for coordinating a public network where participants have no pre-existing trust relationship. Enterprise, government, and defense deployments operate as private networks (Section 10.6) with zero beat involvement. Private networks use the same cryptographic protocol (Layer 1) and the same data structure (DAM) without any economic layer. Beats are not required for validation, signing, verification, or any cryptographic operation.
>
> **This section is the economic specification.** The public network's beat mechanics are specified here and in the related protocol sections — there is no separate economic document. Because the beat is an internal accounting unit and never a tradeable asset (Section 9.2), the protocol defines no token-sale, market-distribution, or securities-classification model. Supply mechanics, the participation faucet, governance weighting, and anti-centralization measures appear here and in Sections 10.3–10.4, 11.1, and 11.17.

9.1 Role of the Beat

The Elara Protocol **beat** is a utility unit that enables four protocol functions:

1. **Witness staking** — nodes stake beats to participate in attestation, creating an economic barrier against Sybil attacks (Section 11.1) and aligning incentives for honest witnessing.
2. **Priority network services** — requesting priority propagation, requested witnessing, and Layer 3 AI capabilities consumes beats. Contributing resources (bandwidth, compute) earns beats. Basic propagation and Layer 1 validation are always free (Section 3.5).
3. **Storage delegation** — nodes that cannot store records long-term pay storage-specialized nodes to hold them. The delegating node always retains its signed record header.
4. **Governance participation** — beat stakers participate in protocol governance through conviction voting (Section 10.3), subject to the anti-centralization constraints detailed in Section 10.4.

9.2 Design Principles

The beat economic model is guided by four principles:

- **Conservation over inflation** — the protocol targets a fixed-supply model where beats circulate between producers (witnesses) and consumers (record submitters) rather than being continuously minted. Supply mechanics are described in Section 11.17.
- **No gas fees** — transaction costs are borne by the network's reciprocal witnessing model, not by per-transaction fees. At scale (millions of nodes), gas fees would be economically prohibitive.
- **Layer 1 is always free** — local validation never has a cost (Section 3.5, Section 11.10). The beat economy applies only to Layer 2 network services.
- **Utility, not speculation** — the beat is an internal protocol accounting unit, not an investment vehicle: it is never offered, sold, listed, or traded. No ICO, no pre-sale, no exchange listing, none planned.

9.3 Protocol Integration Points

The beat interacts with the protocol at these specific points:

- **PoWaS attestation** (Section 11.1) — witnesses stake beats to participate; difficulty scales with stake
- **Priority propagation** (Section 11.10) — paid tier for faster global reach and requested witnessing
- **Dispute arbitration** (Section 11.13) — parties stake beats to invoke arbitration panels
- **Conviction voting** (Section 10.3) — beat-weighted governance with time-locked conviction
- **Zone health metrics** (Section 11.22) — total staked beats as a zone health indicator
- **Storage delegation** — nodes delegate record storage to storage-specialized nodes in exchange for beats

The mechanisms summarized here are specified in detail elsewhere in this document: the supply model in Section 11.17, governance and anti-centralization measures in Sections 10.3–10.4, the participation faucet in Section 9.5, and Sybil-cost analysis in Section 11.1.

Validation beats are not a financial instrument — by design. Beats exist only to meter participation — staking for sybil-resistance, witnessing, and storage delegation. They are never sold, issued for money, listed, or traded: no market, no price, no monetary claim. The unit is built for granular accounting, not value — each beat divides into **one billion base units** (10^9 — nine decimal places), for a total of about **ten quintillion** base units (10^{19}) across the whole network: enough to meter fractional work across billions of devices. Each base unit is economically meaningless **not because there are so many, but because beats are never traded** — they are accounting entries for work done, not an asset to hold. That is deliberate: a unit of account, never a store of value.

9.4 Per-Epoch Reward Formula (v0.7.3)

Witness rewards are computed per epoch based on actual contribution:

`reward = base × record_count_factor × trust_multiplier × diversity_bonus × entity_diminishing_`

Where: - `record_count_factor = min(record_count / 100, 2.0)` — more records sealed in the epoch means more reward. Empty epochs (record_count = 0) produce zero reward. - `trust_multiplier` — node's accumulated trust score (Section 11.12) - `diversity_bonus` — reward premium for witnesses that increase zone diversity (different subnet, organization) - `entity_diminishing_returns` — per-identity scaling that prevents reward concentration, following the same anti-centralization principle as the governance cap (Section 10.4)

This formula ensures rewards flow to nodes that do real work (witnessing actual records) rather than to idle stakers.

9.5 Bootstrap Distribution (Participation Faucet)

The public network seeds its initial participant set through a **participation faucet, not an airdrop**. A fixed fraction of genesis supply (the Network Bootstrap pool — 30%) is reserved for the first 10,000 node operators and claimed first-come-first-served once a node has demonstrated participation (joined, synced, and begun witnessing). Beats are **earned by running a node, never granted to an address for merely existing** — there is no ICO, no pre-sale, and no airdrop. A node that never participates never receives a bootstrap allocation, and each identity may claim once (`claim_bootstrap`).

Claimed bootstrap beats land in the account's `available` balance and are immediately usable for staking, priority services, and storage delegation. Per-epoch witness rewards (Section 9.4) accrue to the same `available` balance.

Future work — uptime-weighted unlock (designed, not yet active). The account model reserves a second balance field, `vested_locked`, for a planned scheme that would release part of a bootstrap allocation against sustained-uptime milestones rather than all at claim time — rewarding nodes that stay online through the network's early growth:

Cumulative Uptime	Unlocked %
24 hours	1%
72 hours	5%
168 hours (7d)	10%
720 hours (30d)	20%
2,160 hours (90d)	30%
4,320 hours (180d)	50%
8,760 hours (1yr)	100%

When active, unlocked beats would move monotonically from `vested_locked` to `available` at each milestone (once unlocked, never re-locked), and a node offline for 15 consecutive days would have its remaining `vested_locked` drain at 1%/day back to the bootstrap pool for new participants. **This mechanism is implemented in source but disabled by default and inactive in every current release — `vested_locked` is zero for all production accounts.** The activation gate is held closed deliberately: the current uptime accumulator mutates a consensus-sealed account field from wall-clock time, which is non-deterministic across nodes and would diverge the account-state root. Activation awaits an epoch-gated, record-driven redesign that makes uptime accumulation consensus-safe. It is described here as a roadmap direction only; the live distribution is the participation faucet above.

---

10. Governance

10.1 Multi-Zone Autonomy

The Elara Protocol's governance is federated, not centralized:

- Each zone (geographic or planetary) governs itself
- Zones can adopt different policies for local matters (storage requirements, minimum witness counts, AI layer participation)
- Cross-zone matters (protocol version, cryptographic standards, beat economics) require multi-zone consensus

10.2 Decision Categories

Zone-local (decided by zone stakeholders): - Minimum witness count for local trust thresholds - Storage retention policies - AI layer participation requirements - Local fee structures

Cross-zone (decided by all zones): - Protocol version upgrades - Cryptographic algorithm additions or deprecations - Beat supply changes - New entity type definitions - Zone creation or dissolution

10.3 Voting Mechanism

Cross-zone decisions use a **conviction voting** model [36] (inspired by conviction voting mechanisms pioneered by Commons Stack and 1Hive, 2019):

- Beat stakers express preferences by staking beats toward proposals
- Voting weight accrues over time according to: conviction(t) = stake × (1 - e^(-t/τ)) where t is days staked and τ = 7 days (time constant). Weight reaches ~63% at 7 days, ~86% at 14 days, ~95% at 21 days, and ~98.6% (effectively full conviction) at 30 days. The exponential ramp makes flash-vote attacks economically pointless — meaningful conviction requires sustained commitment.
- Proposals require both **supermajority** (>67% of conviction-weighted stake) and **quorum** (>25% of all staked beats participating)
- Implementation is delayed 30 days after passing (allowing zones to prepare)

  Note: The conviction-voting model described here is complemented by additional governance mechanisms — trust-weighted random committee selection, identity-based voting caps (Section 10.4), and anti-pooling measures.

10.4 Governance Attack Mitigations

The conviction voting mechanism (Section 10.3) is designed to resist several known governance attacks:

Sybil resistance: Voting power is proportional to staked beats, not identity count. Creating 1,000 identities with 1 beat each provides the same voting power as 1 identity with 1,000 beats — Sybil attacks gain nothing. The economic cost of acquiring sufficient beats to dominate governance scales with network value.

Flash-vote attacks: Conviction voting's time-weighted staking prevents an attacker from borrowing beats, voting, and returning them in a single transaction. The exponential conviction curve means beats staked for less than 7 days carry less than 63% weight, and full conviction (~98.6%) requires 30 days of sustained staking. This makes flash attacks economically pointless — the capital lockup cost exceeds any governance manipulation benefit.

Vote buying: While the protocol cannot prevent off-chain vote buying, the 30-day implementation delay (Section 10.3) allows the community to detect and respond to suspicious voting patterns before changes take effect. Zones can invoke emergency veto (requiring >75% of anchor nodes) to block proposals that passed through suspected manipulation.

Plutocracy mitigation: Raw beat-weighted voting favors wealthy participants. The protocol applies a **square-root dampening** to conviction weight. The combined governance weight formula is:

```
governance_weight = min(√stake × (1 - e^(-t/τ)), (1/√N) × TOTAL_STAKED)
where τ = 7 days, N = total active stakers, and the cap is per-identity (see below)
```

An entity staking 10,000 beats has √10 ≈ 3.16× the influence of an entity staking 1,000 beats, not 10×. The conviction curve then weights by lock duration (reaching ~98.6% at 30 days), and the scaling cap ensures no single identity exceeds 1/√N of total governance weight regardless

of stake size (where N = total active stakers — at 100 stakers the cap is 10%, at 10,000 it's 1%, at 1M it's 0.1%). Section 10.4 details the governance attack mitigations and pool-centralization limits.

Emergency veto abuse: The emergency veto (>75% of anchor nodes) is a powerful mechanism that could itself be gamed. Constraints: (1) a veto can only block, never propose — it cannot be used to force changes, only prevent them; (2) vetoes are rate-limited to 2 per zone per quarter; (3) any veto triggers a mandatory public disclosure of the veto rationale within 72 hours; (4) if a vetoed proposal passes a second vote with >80% conviction after the disclosure period, the veto is overridden. This ensures the veto is a circuit breaker, not a permanent kill switch.

10.5 Graceful Divergence

If zones cannot reach consensus on a cross-zone matter, the protocol supports **graceful divergence**:

1. The disagreeing zone announces a fork intention
2. A 90-day mediation period allows for compromise
3. If unresolved, the zone forks the protocol — maintaining DAG compatibility for historical records but diverging on the disputed feature
4. Cross-zone sync continues for shared historical data
5. New validations under the divergent rule are tagged with the fork identifier

This is not a failure state — it is a design feature. A protocol intended to span planets and centuries must survive political and philosophical disagreement without collapsing.

10.6 Private Networks and Network Publication

The governance model described above assumes participation in the public Elara network. But the protocol's layered architecture (Section 3.2) enables a distinct deployment model: **private Elara networks** that operate independently, with no public network participation.

10.6.1 Private Deployment Model A private Elara network uses the same protocol stack — post-quantum signatures, DAM structure, wire format, witness consensus — within a closed organizational boundary. This is not a fork or a modification: it is the protocol operating at Layer 1 + private Layer 2, without connecting to the public Layer 2.

Examples: - An aerospace manufacturer validating firmware provenance across factory sites - A space agency recording mission-critical software authorship and satellite telemetry within its engineering teams - A pharmaceutical company maintaining tamper-evident drug trial data chains - An automotive OEM tracking supply chain component validation across suppliers - Defense contractors validating firmware and mission-critical software across classified environments - Military operations validating tactical decisions, sensor data, and autonomous system logs - Robotics companies validating every autonomous decision for regulatory compliance and liability - Semiconductor fabrication plants validating billions of sensor readings per day across factory floors (Section 3.6)

These organizations benefit from the cryptographic properties (post-quantum security, causal ordering, tamper evidence) without requiring public consensus or beat economics. Governance is organizational, not protocol-level. Anti-centralization mechanisms are unnecessary — internal trust hierarchies are appropriate for corporate environments.

Key architectural properties of private deployments:

1. **Layer 1 is unchanged.** The same keygen, signing, hashing, and DAG operations. Records created on private networks are structurally identical to public records.

2. **Layer 2 is scoped.** Discovery, propagation, and witnessing occur only within the private network boundary. The organization controls the peer set.
3. **Layer 3 operates independently.** AI analysis runs against the private DAG. No data leaves the boundary.
4. **No beat requirement.** Resource allocation is organizational, not market-based.

10.6.2 The Publication Spectrum Private networks exist on a spectrum from fully closed to fully public:

Mode	Description	Trust Model	Beat Involvement
Fully private	Closed network, no external connections	Internal hierarchy	None
Federated	Bilateral sharing between partner organizations	Mutual trust agreements	Optional (cross-org settlement)
Selective publication	Some record types published to public network	Hybrid: internal + public	For published records only
Full publication	All records participate in public consensus	Public witness consensus	Full participation

An organization may operate at different points on this spectrum for different record types simultaneously. Internal engineering records remain private. Supply chain attestations are shared with partners (federated). Finished product validations are published to the public network.

10.6.3 The NETWORK_PUBLISH Protocol

> **Implementation-status note (honest-claims rule): the NETWORK_PUBLISH transition described in this section is DISABLED in the current runtime** (`NETWORK_PUBLISH_ENABLED = false`, compile-time guarded). The per-record model below — imported records *entering* public consensus and gaining *retroactive* native standing (step 5 and §10.6.4) — was found unsound: a validation record's signed bytes carry no realm/network binding (Assumption A8 of the MESH-BFT merge semantics), so an imported record is consensus-indistinguishable from a native one, and the single-network safety theorem does not cover adopting a foreign network's records as native settlement parents. The mechanism is being reframed to **inert-import** — public consensus attests only that a publication *bundle* (source root + Merkle root of the imported set + completeness proof + external time anchors) existed at an anchored time, conferring **zero native standing** (no settlement-parent role, no stake, no witness weight, no cross-zone debit basis). This section documents the original design, retained for reference pending that reframe and a proven multi-root merge theorem.

When a private network transitions records to the public network — partially or fully — the protocol defines a **NETWORK_PUBLISH** record type:

```
Record Type: NETWORK_PUBLISH (0x0E)

Fields:
  source_network_id    bytes     Public key of private network root authority
  published_records   RecordSet Record ID range, classification filter, or DAG subtree
  publication_scope     enum      FULL | SELECTIVE | FEDERATED
  target_zone           ZoneID    Destination zone in the public network
  historical_depth      uint64    How far back (in records or time) to publish
  redaction_policy    Policy   Which metadata fields are stripped before publication
  transition_mode       enum      SNAPSHOT | STREAMING | GRADUAL
  completeness_proof   bytes    Optional Merkle proof that published set is complete
                                relative to the source DAG (prevents selective omission)
  anchor_trail       AnchorProof[]  Chain of external state-root commitments made
                                    DURING private operation (public-mesh seals, independent
                                    timestamping services, or other widely-witnessed media).
                                    Determines the age credibility of the published history.
```

Transition modes:

- **SNAPSHOT** — Publish entire DAG subtree at once. Immediate verifiability, high bandwidth cost.
- **STREAMING** — Publish records chronologically over a defined period. Allows the public network to absorb and verify incrementally.
- **GRADUAL** — Begin with recent records, extend historical depth over time. Lowest initial exposure.

Verification process:

When the public network receives published records:

1. **Signature verification.** Every record's post-quantum signature is verified independently. Signatures are self-contained — they do not depend on network state.
2. **Causal chain verification.** Parent references are followed to ensure the DAG structure is internally consistent. Missing parents (unpublished records referenced by published records) are flagged as known gaps, not errors.
3. **Temporal consistency (internal only).** Timestamps are checked for monotonicity within causal chains — a child cannot predate its parent. This proves internal ordering only, **never calendar time**: every clock and key inside a private network belongs to its operator, so a fabricated DAG can satisfy this check perfectly.
4. **Completeness check.** If a completeness proof is provided, it is verified against the published record set. This proves the organization is not selectively omitting unfavorable records from a subtree.
5. **Retroactive witnessing.** Public nodes can witness historical records, adding new trust attestations that reference the original (unchanged) records.
6. **Anchor-trail verification.** Each `anchor_trail` entry is verified against its external medium (a public-mesh epoch seal, an independent timestamp proof). Each verified anchor proves the committed state — and every record beneath it — **existed by** the anchor's time.

Age credibility and the anchor-density law. External anchoring is the *only* mechanism that makes a published history's age verifiable. Internal evidence cannot prevent backdating (a fabricator controls all keys and clocks in its own realm); beacon references at key generation prove only freshness (*no older than*), while anchors prove existence (*no younger than*) — age claims require the latter. The maximum undetectable backdating window equals the largest gap between consecutive anchors, so **publication credibility is proportional to anchor density**. A private network that never anchored may still publish: its records receive structural verification (steps 1–4) and prospective trust from retroactive witnessing (step 5), but the

network MUST treat its declared age as zero. Operators intending a future Validation IPO should anchor from day one.

Anchor-media requirements (no single foundation). An anchor's evidentiary value MUST rest on hash structure — a Merkle path into a widely-replicated hash chain — never on the host medium's signatures. Quantum adversaries break signature schemes (Shor), not 256-bit hash preimages (Grover leaves ~2^128 work), so a hash-bound anchor survives even the collapse of its host's signature economy; witnesses whose value rests on a signature (e.g., RFC 3161 credits) diversify the set but MUST themselves be re-wrapped by hash-based media over time. Anchors MUST span multiple independent media; verifiers SHOULD archive the host-chain headers their proofs traverse, removing dependence on the host's future availability or canonical-chain consensus. Finally, the running state root MUST be re-anchored continuously: each new anchor re-witnesses the entire history beneath it under every newer, stronger medium — including, eventually, the public mesh itself — which is how the system migrates off any bootstrap medium without losing its past.

10.6.4 Governance Implications

> **Implementation-status note: DISABLED in the current runtime.** The governance events below follow from the NETWORK_PUBLISH transition of §10.6.3, which is compile-time disabled (`NETWORK_PUBLISH_ENABLED = false`) pending the inert-import reframe. The retroactive trust-bootstrapping and IPO-style governance weight described here do not occur on the live protocol; retained for reference. See §10.6.3.

Private-to-public transitions create governance events:

Zone registration. A large private network publishing as a new zone follows the zone creation process (Section 10.2, cross-zone decision). The zone's internal governance may differ from the public network's governance model.

Trust bootstrapping. Published historical records carry internal trust (accumulated from private witnesses) but zero public trust. Public trust accumulates through retroactive witnessing. A 10-year-old published record may reach high public trust within weeks if many public nodes verify and witness it.

Representation. Once published, the organization's nodes become public network participants with governance weight proportional to their staked beats and accumulated conviction (Section 10.3). A large organization entering the public network could represent significant governance weight — the square-root dampening and 5% cap per identity (Section 10.4) limit this concentration.

The analogy to traditional markets is deliberate: a private network choosing to publish is structurally similar to a company filing an IPO — historical records are disclosed, public trust is established based on track record, and the entity gains access to the broader ecosystem's resources (public witnessing, storage delegation, cross-network attestation) in exchange for transparency.

Section 11.34 analyzes the economic dynamics and failure modes of this transition, including ingestion rate-limiting as an anti-gaming mechanism.

---

11. Adversarial Resilience

A protocol that cannot answer its critics is a protocol that does not survive peer review. This section addresses 35 attack vectors and design challenges — from Sybil resistance to formal verification strategy.

11.0 Threat Model

Before analyzing specific attacks, we define the adversary model:

Adversary capabilities (ordered by strength):

Level	Adversary	Capabilities
1	Individual	Controls one or a few nodes, limited resources
2	Organization	Controls hundreds of nodes, significant capital, legal standing
3	Nation-state	Controls network infrastructure, can compel ISPs, legal coercion
4	Quantum adversary	Access to cryptographically relevant quantum computer (future)

Assumptions:

- **Honest majority:** At least 2/3 of staked weight in any zone is held by honest nodes (standard BFT assumption)
- **Cryptographic hardness:** NIST PQC primitives (Dilithium, Kyber, SPHINCS+) remain computationally infeasible to break for the claimed security levels
- **Network model:** Asynchronous with eventual delivery — messages may be delayed arbitrarily but are eventually delivered to honest nodes
- **No trusted hardware:** The protocol does not assume TEEs, secure enclaves, or tamper-proof devices. Security derives from cryptography and economics, not hardware trust.
- **Rational actors:** Witness nodes are economically rational — they will not spend resources on actions that reduce their expected returns

What the protocol does NOT defend against:

- Compromise of a creator's private key combined with physical theft of unreleased content (this is a physical security problem, not a protocol problem)
- A quantum adversary that breaks both lattice-based AND hash-based cryptography simultaneously (no known path to this)
- Social engineering that induces a user to sign content they did not create (the protocol validates signatures, not intent)

Security (11.1–11.9)

11.1 Sybil Resistance

The attack: An adversary generates one million keypairs and creates one million fake "witness" nodes. Every validation record they publish instantly accumulates a million witnesses, making fraudulent claims appear globally trusted.

This is the oldest problem in decentralized systems. The Elara Protocol addresses it through layered defense:

Layer 1: Proof of Work-at-Stake (PoWaS)

Witness attestation is not free. To attest to a validation record, a witness node must:

1. Stake a minimum amount of beats (economic cost)
2. Solve a lightweight proof-of-work puzzle calibrated to the attestation (computational cost — not industrial-scale mining, but enough to make mass attestation expensive)
3. Maintain a reputation score based on attestation history (time cost)

PoWaS puzzle construction:

```
puzzle_input  = SHA3-256(record_id || witness_pubkey || nonce)
difficulty    = BASE_DIFFICULTY × 1000 / sqrt(stake_base_units)
target        = 2^256 / difficulty
valid_if      = puzzle_input < target
```

The difficulty is inversely proportional to the square root of the witness's staked amount, bounded by a minimum and maximum difficulty to prevent both trivial solutions by large stakers and impossible puzzles for small stakers:

```
effective_difficulty = clamp(difficulty, MIN_DIFFICULTY, MAX_DIFFICULTY)
where MIN_DIFFICULTY = BASE_DIFFICULTY / 10 (no one gets a free pass)
      MAX_DIFFICULTY = BASE_DIFFICULTY × 10 (minimum stake is viable)
```

With stake measured in base units (10^9 per beat), the unclamped inverse-sqrt curve differentiates only sub-floor stakes: every witness at or above the 100-beat admission floor clamps to MIN_DIFFICULTY, so all admitted witnesses perform the same minimum work — the curve's differentiation is a property of the sub-floor band, not of the admitted set. The bounds ensure that very large stakers still perform meaningful computation and very small stakers are not excluded entirely. This creates a combined economic-computational barrier: attacking cheaply requires massive computation; attacking with minimal computation requires massive stake.

Per-zone, per-epoch difficulty retargeting toward a target attestation rate is a design-stage extension — the shipped runtime uses the static clamp above, and difficulty does not currently adjust at runtime.

An adversary creating a million Sybil nodes would need to acquire beats for each (economic barrier), solve puzzles for each attestation (computational barrier), and build reputation over time for each node (temporal barrier). The cost of attack scales linearly; the defense is multiplicative.

Layer 2: Social Graph Analysis

The AI-assisted analysis layer (Layer 3 of the protocol architecture) monitors witness patterns:

- Nodes that only attest to each other's records (closed clusters) receive reduced trust weight
- Nodes with attestation patterns inconsistent with their stated geography or entity type are flagged
- Sudden spikes in new witnesses for a specific creator trigger anomaly alerts

This is not a centralized filter — it is a distributed heuristic that each node can independently compute from the DAM's public data.

Layer 3: Trust Decay for New Identities

New keypairs start with zero trust. Their attestations carry minimal weight. Trust accumulates through:

- Duration of existence (older keys are harder to fake at scale)
- Diversity of attestation partners (attesting to many unrelated creators)
- Consistency of behavior (regular, plausible patterns)
- Cross-zone attestation (a key attested by nodes in multiple geographic zones is harder to Sybil)

A Sybil army of fresh keys carries near-zero attestation weight. By the time they've aged enough to matter, the temporal and economic costs make the attack unprofitable. A related variant — Sybil amplification through physical device recycling (wiping and re-enrolling the same hardware) — is addressed in Section 11.33.

Layer 4: Leaf Node Independence

Critically, Layer 1 validation (local, offline) is immune to Sybil attacks entirely. The poem on the phone in Nairobi is cryptographically valid regardless of how many fake witnesses exist on the network. Sybil attacks can pollute trust scores, but they cannot invalidate a legitimate local validation. The creator's signature is the ground truth; witnesses are corroboration, not authority.

11.2 Key Compromise and Revocation

The scenario: A teenager's phone is stolen. The thief has her private key. Without intervention, the thief can sign new work as her and validate fraudulent claims under her identity.

Revocation mechanism:

Every Elara identity supports a **revocation record** — a special validation record that:

1. Is signed with the compromised key (proving ownership)
2. Contains a revocation timestamp (all signatures after this time are invalid)
3. Optionally designates a successor key (migration, not just termination)
4. Is flagged as REVOCATION type and propagates with highest priority across the network

Once published, the revocation record is immutable on the DAM. All future signatures from the compromised key are rejected by any node that has received the revocation.

But what if the thief revokes first?

This is the harder problem. The protocol handles it through **pre-committed recovery keys:**

At identity creation, the user can (and is strongly encouraged to) generate a **recovery keypair** stored separately — written on paper, saved on a USB drive, held by a trusted person. The recovery key's public half is embedded in the original identity record.

Only the recovery key can: - Override a fraudulent revocation - Designate the legitimate successor key - Prove identity in a dispute where both parties hold valid keys

If no recovery key was pre-committed, the dispute becomes a social/legal matter — but the DAM preserves the full timeline of both parties' claims, providing evidence for resolution.

Dual compromise (both primary and recovery keys lost): If an adversary obtains both keys, the identity is irrecoverably compromised. The user must create a new identity and rely on out-of-band evidence (legal records, prior witnesses, incremental creation chains) to re-establish attribution of their work. This is the catastrophic failure mode — analogous to losing both a password and all recovery options. The protocol cannot solve this cryptographically; it can only preserve the evidentiary record for dispute resolution. Users who require higher assurance should use multi-party recovery schemes (e.g., Shamir secret sharing across trusted contacts) to protect their recovery key.

Key rotation:

The protocol **specifies** scheduled key rotation without identity loss: a rotation record — signed by both the old and new keys — would maintain continuity, keeping all past work attributed to the identity while future work uses the new key, limiting the damage window of a compromise. **This mechanism is specified but not yet operational** (an identity is currently addressed by the hash of its active key, so a rotated key resolves to a different account — see §14 and KNOWN-LIMITATIONS). Today, key **revocation** (the compromised-key tombstone) is the live compromise-recovery path; rotation lands with a versioned identity→active-key index.

Note: deliberate key destruction through device wipes — where the goal is to sever accountability rather than steal identity — is a distinct attack vector addressed in Section 11.33.

11.3 Light Clients

The problem: A $30 phone cannot store the full DAM. A Mars colony with limited bandwidth cannot download Earth's complete history. How do lightweight nodes verify claims without the full dataset?

Solution: Merkle Proof Verification

The DAM maintains a Merkle tree over all validation records. A light client can verify any specific record with:

1. The validation record itself (~4-5 KB)
2. A Merkle proof path (~log2(N) × 32 bytes — for a billion records, this is ~960 bytes)
3. The Merkle root (32 bytes, published by anchor nodes)

Total verification payload for a single record: **under 6 KB**, regardless of how large the DAM grows.

Trust headers:

Anchor nodes publish periodic **trust headers** — compact summaries containing:

```
TrustHeader {
    zone:           zone identifier
    timestamp:      vector clock position
    merkle_root:    root hash of all records in this zone
    record_count:   total validation records
    witness_summary: aggregated trust statistics
    anchor_signatures: [signed by multiple anchor nodes]
}
```

A light client that trusts at least one anchor node can verify any claim against the trust header without downloading the full DAM. This is analogous to Simplified Payment Verification (SPV) in proof-of-work systems, extended for the DAM's multi-zone structure.

Progressive download:

Light clients can optionally download DAM data incrementally:

- **Level 0:** Trust headers only (~1 KB/day) — sufficient for basic verification
- **Level 1:** Headers + own records + records they've queried (~100 KB–10 MB)
- **Level 2:** Headers + local zone history (~10 MB–1 GB)
- **Level 3:** Full DAM replica (anchor/archive nodes only)

A budget smartphone operates at Level 0 or 1. It validates locally, syncs its own records, and verifies others via Merkle proofs. It never needs the full DAM.

11.4 Network Bootstrap

The problem: The first node has no witnesses. The second node has only one possible witness. How does a trust network start from zero?

Phase 1: Genesis Anchor (nodes 1–10)

The founding team operates the first anchor nodes. These nodes are explicitly identified in the protocol's genesis block as **genesis anchors** — trusted not by accumulated attestation, but by their role in creating the network. This is centralization, and it is acknowledged openly.

Every decentralized network starts centralized. Early proof-of-work networks had one miner. Smart contract platforms launched with foundations. The difference is the exit plan.

Phase 2: Early Growth (nodes 10–1,000)

Genesis anchors actively attest to new nodes' identity registrations. Early participants earn elevated trust through:

- Direct attestation by genesis anchors (bootstrapping trust)
- Participation in testnet validation (proving reliability)
- Contribution to the codebase, documentation, or tooling (proof of commitment)

Beat incentives during this phase are elevated — early validators earn disproportionate rewards to compensate for the network's low utility. The bootstrap distribution (participation faucet) is described in Section 9.5.

Phase 3: Decentralization Threshold (nodes 1,000–10,000)

At 1,000 active witness nodes across at least 10 geographic regions, the protocol reaches its **decentralization threshold.** At this point:

- Genesis anchors' special trust status expires (they become regular anchor nodes)
- Governance transitions from founding team to beat-weighted voting
- The protocol is self-sustaining — no single entity can disrupt consensus

Phase 4: Critical Mass (10,000+ nodes)

The network effects take over. Developers build on the protocol because users are there. Users join because developers have built tools. Institutions adopt because the network is too large to ignore.

The bootstrap problem is real, but it is a solved problem in practice. The challenge is not technical — it is social. The protocol must be useful enough that the first 1,000 people choose to run nodes. Section 3.5 (Minimum Viable Validation) is the answer: the protocol is useful to a single person with a single device before anyone else joins.

Fast Snapshot Sync for Onboarding (v0.7.7)

Once a zone has sealed its first few hundred epochs, genesis replay becomes impractical for new full nodes. A 10M-record zone with 100K epoch seals takes hours to reconstruct from scratch. The protocol addresses this with **epoch-indexed state snapshots** served by archive nodes:

1. Archive nodes (node_profile = Archive, §11.3) emit a signed state snapshot every `archive_snapshot_every_n_epochs` (default 10). Each snapshot is a JSON artifact at `epoch-{N:012}.json` — zero-padded for lexicographic ordering equals numeric ordering — containing: epoch number, SHA3-256 checksum of the contained state, full ledger snapshot, per-zone epoch cursors, zone registry root, Merkle tree roots, and the archive node's signature over the checksum (Dilithium3 always; a second SPHINCS+ leg is added when the archive runs signing Profile A — the SPHINCS+ fields are absent otherwise).
2. Retention is bounded: archives keep the last `archive_snapshot_retention` snapshots (default 20) and prune older files. At the default cadence of 10 epochs per snapshot (≈50 s–10 min, since the epoch interval adapts between the 5 s floor and 60 s ceiling), 20 retained snapshots cover roughly 17 minutes (under sustained load) to 3.3 hours (at the 60 s ceiling) of state history.
3. A new node onboards by: (a) fetching the snapshot index from any archive via `GET /snapshot/epochs` (returns sorted epoch list); (b) downloading the latest snapshot via `GET /snapshot/epoch/{N}`; (c) verifying both signatures and the checksum; (d) applying the snapshot to its local state; (e) fetching only records produced *after* epoch N via the standard delta sync path (§11.22.1 Merkle proofs + gossip). Onboarding time collapses from hours to minutes.
4. Snapshot emission resumes from the highest on-disk epoch on archive restart — the archive does not re-emit snapshots it has already written. Snapshot *emission* is archival-only — full-zone and light nodes never produce snapshots; *consumption* is open to any onboarding node (step 3 above), after which non-archive nodes pull records directly.

Safety under compromised archive: A malicious archive that signs a bogus snapshot is detected by the receiving node on signature or checksum verification. Because snapshots are addressable by epoch number, a receiving node can cross-verify by fetching the same epoch from a second archive and comparing checksums — if they disagree, at least one archive is Byzantine and the receiver can escalate to dispute resolution (§11.13). Snapshots never replace the attested epoch seal chain; they are an acceleration mechanism for consensus-identical state.

Testnet status: Epoch-indexed snapshots are implemented in the Elara Runtime (public `v0.2.0` release, `archive_snapshot_loop`); an archival anchor node has been observed emitting epoch-indexed snapshots (e.g. `epoch-000000004004.json`) under the default 10-epoch cadence.

11.5 Immutability vs. Right to Deletion (GDPR)

The conflict: The EU's General Data Protection Regulation (GDPR) grants individuals the "right to erasure" — the right to demand that their personal data be deleted. The Elara DAM is immutable — records cannot be deleted. These appear to be irreconcilable.

The resolution: The Elara Protocol validates hashes, not content.

A validation record on the DAM contains:

- A cryptographic hash of the content (not the content itself)
- A public key (pseudonymous, not a name or address)
- A timestamp and DAG references

The actual content — the poem, the document, the sensor reading — is stored off-DAM, under the creator's control. The DAM stores proof that the content existed, not the content itself.

GDPR compliance path:

1. **Delete the content** — the creator removes the original work from their device and any storage. The hash on the DAM becomes an orphan — it proves that *something* existed, but that something is gone. The hash cannot be reversed to recover the content (SHA3-256 is a one-way function).
2. **Revoke the identity** — the creator issues a revocation record. Their public key is marked as revoked. The pseudonymous link between the hash and any real-world identity is severed.
3. **The DAM retains only:** an orphaned hash signed by a revoked pseudonymous key. This satisfies GDPR's erasure requirement because no personal data remains — only mathematical artifacts that cannot be linked to a natural person.

For PRIVATE and SOVEREIGN classifications, the situation is even cleaner: the content hash was never visible on the DAM in the first place. Only a zero-knowledge proof exists. Revoking the key makes the proof unattributable.

For IoT and device data, GDPR applies only to personal data. Sensor readings from industrial equipment or environmental monitors are not personal data and are not subject to erasure rights.

Precedent: This approach aligns with guidance from EU data protection authorities on blockchain and GDPR, which acknowledge that storing hashes of personal data (rather than the data itself) may satisfy the regulation's requirements, particularly when combined with key deletion.

The Elara Protocol does not fight regulation. It is designed so that compliance is architecturally natural, not a retrofit.

11.6 Timestamp Gaming (Offline Backdating)

The attack: A malicious actor sets their device clock to January 2024, validates stolen work, then syncs to the network in February 2026. The validation record shows a 2024 timestamp, granting false priority over the actual creator.

This is a fundamental weakness of any system that allows offline validation with local timestamps. The Elara Protocol addresses it through three mechanisms:

Mechanism 1: Causal Anchoring

When an offline node syncs to the network, its validation records must reference existing DAM records as parents. The sync protocol automatically inserts a **causal anchor** — a reference to the most recent record the node received during synchronization.

This creates an ironclad constraint: a record with a causal anchor from February 2026 cannot have been created in January 2024, regardless of what the local timestamp claims. The DAG structure itself disproves the backdated timestamp.

Mechanism 2: Temporal Witness Scoring

Trust scores weight the gap between claimed creation time and first network appearance:

```
trust_penalty = f(time_claimed, time_first_witnessed)

If first_witnessed - time_claimed < 24 hours:  no penalty
If first_witnessed - time_claimed < 7 days:    minor penalty (0.9x trust)
If first_witnessed - time_claimed < 30 days:   moderate penalty (0.5x trust)
If first_witnessed - time_claimed > 30 days:   severe penalty (0.1x trust)
If first_witnessed - time_claimed > 1 year:    near-zero trust (0.01x)
```

A record that claims to be two years old but was first seen today carries almost no trust weight. It exists on the DAM (nothing is deleted), but its trust score reflects the suspicion.

Mechanism 3: Concurrent Priority Protocol

When two records claim the same content hash at different times, the protocol does not automatically award priority to the earlier timestamp. Instead, it evaluates:

1. **Causal anchoring** — which record has DAM-verifiable temporal context?
2. **Witness accumulation speed** — a legitimate record from a connected device accumulates witnesses in real-time; a backdated record arrives in a burst
3. **Device attestation history** — a device that has been consistently online and validating is more credible than one that appears with years of backdated records
4. **Cross-reference** — do other records from the same creator show consistent timelines, or is there a suspicious gap?

The result: local timestamps remain useful (and honest nodes produce accurate ones), but they are never the sole basis for priority claims. These three mechanisms work in concert — causal anchoring provides structural proof, temporal witness scoring applies economic penalties, and the concurrent priority protocol evaluates the full context. No single mechanism is sufficient alone, but together they make timestamp gaming detectable, penalized, and ultimately unprofitable. The DAG's causal structure serves as the primary authoritative clock; timestamps are supplementary metadata.

11.7 The Originality Problem

The challenge: The Elara Protocol proves that a specific hash, signed by a specific key, existed at a specific point in the DAM's causal order. It does NOT prove that the key holder created the content. A thief could hash a stolen novel and register it before the author does.

This is not a bug — it is an inherent limitation of any cryptographic validation system. No protocol can prove creation; it can only prove possession and timing. The Elara Protocol is honest about this boundary and provides tools to make theft as difficult and detectable as possible:

Tool 1: Incremental Validation

Creators are encouraged to validate work incrementally — not just the final product, but drafts, outlines, sketches, intermediate versions. A poet who validates six drafts over three weeks has a validation trail that a thief cannot replicate. The thief can hash the final poem, but they cannot produce the creative history.

```
Draft 1 (Feb 1)  → hash_a, signed by author
Draft 2 (Feb 5)  → hash_b, signed by author, references hash_a
Draft 3 (Feb 12) → hash_c, signed by author, references hash_b
Final  (Feb 20)  → hash_d, signed by author, references hash_c

vs.

Stolen copy (Mar 1) → hash_d, signed by thief (no history)
```

The DAM preserves the full creative chain. A single hash proves possession. A chain of hashes proves process — and process is much harder to fake.

Tool 2: Composite Attribution Records

For AI-assisted work (the majority of future digital creation), the CollaborationRecord structure (Section 6.3) creates inherent provenance. The AI's keypair co-signs the work. A thief who steals the output cannot produce a matching CollaborationRecord unless they also compromised the AI's private key.

Tool 3: Content Fingerprinting (Layer 3)

The AI intelligence layer can compute similarity hashes (locality-sensitive hashing, perceptual hashing for images/audio) alongside cryptographic hashes. When a new validation is submitted, the AI layer can flag near-duplicates:

- Exact content match with different creator → immediate conflict flag
- High similarity (>90%) with existing validated work → plagiarism warning
- Same content validated by two keys → priority dispute initiated

This does not prevent theft, but it detects it — and detection on an immutable ledger is a powerful deterrent.

Tool 4: Honest Framing

The whitepaper and all protocol documentation explicitly state: **validation proves possession and timing, not creation.** This prevents false expectations and ensures that legal systems interpret DAM records correctly. A validation record is evidence, not verdict. It is strong digital evidence of "I had this at this time," but it is not omniscience.

Courts already understand this distinction — a notarized document proves the document existed at a date, not that the signer wrote it. The Elara Protocol provides cryptographic notarization at global scale.

11.8 Storage Growth and DAM Sustainability

The problem: "Nothing is ever deleted" combined with IoT-scale validation creates unbounded storage growth. A single factory with 10,000 sensors producing readings every second generates ~864 million records per day. At ~4.5 KB per validation record (dominated by the PQC

signature), that is **~3.9 TB per day from one deployment.** At planetary scale, the DAM would grow by petabytes daily.

No single node can store this. The protocol must handle it.

Solution: Hierarchical Storage with Validation Summarization

Tier 1: Leaf and relay nodes — store only what they need

Leaf nodes (sensors, phones) maintain a circular buffer of their own records. Once synced to the network, old records can be evicted from local storage. The node retains its keypair and a pointer to its last synced position — not the full history.

Tier 2: Zone-level summarization

The protocol introduces **epoch summaries** — periodic Merkle tree snapshots that compress historical records into compact proofs:

```
EpochSummary {
    zone:           zone identifier
    epoch:          sequential epoch number
    time_range:     [start_vector_clock, end_vector_clock]
    record_count:   number of records in this epoch
    merkle_root:    root hash of all records in epoch
    creator_roots:  per-creator Merkle subtrees
    summary_hash:   hash of this summary (self-referential)
    signatures:     [anchor node signatures]
}
```

After an epoch is summarized and signed by multiple anchor nodes, individual validation records within that epoch can be pruned from standard nodes. The Merkle root preserves the ability to verify any individual record if the full record is retrieved from an archive node.

Tier 3: Archive nodes — store everything

Archive nodes (Seed Vault Tiers 2-4) maintain the full, uncompressed DAM. These are purpose-built for storage — high-capacity, redundant, geographically distributed. They serve the same role as the Internet Archive: not every node needs the full history, but the full history must exist somewhere.

Tier 4: IoT-specific compression

For high-frequency sensor data, the protocol supports **batch validation** — aggregating multiple readings into a single validation record:

```
BatchValidation {
    device:         device public key
    readings:       [reading_1, reading_2, ..., reading_n]
    batch_hash:     Merkle root of all readings
   individual_hashes: [hash_1, hash_2, ..., hash_n] // required for individual verifiability
    time_range:     [first_reading_time, last_reading_time]
    signature:      device signature over batch_hash
}
```

A sensor producing 1 reading/second can batch 3,600 readings into one hourly validation record. Storage reduction: **3,600x** with no loss of verifiability (individual readings can still be proved via the batch Merkle tree).

Storage projections with batching:

Scale	Raw (unbatched)	With hourly batching	With epoch summarization
1 factory (10K sensors)	~3.9 TB/day	~1.1 GB/day	~100 MB/day (after epoch)
1M personal devices	~450 GB/day	~450 GB/day (no batching)	~45 GB/day
Global IoT (1B devices)	~390 PB/day	~108 TB/day	~11 TB/day

At ~11 TB/day after optimization, the full global DAM grows at ~4 PB/year — large but within the capacity of distributed archive infrastructure. For comparison, YouTube ingests ~720,000 hours of video daily (~1.5 PB). The DAM's storage challenge is significant but solvable with existing technology, and no single node bears the full load.

11.9 Vector Clock Scalability

The problem: Traditional vector clocks maintain one counter per node in the system. With 1 million nodes, each validation record would carry a vector of 1 million integers. At 4 bytes each, that is **4 MB per record** — orders of magnitude larger than the content hash it validates.

Solution: Zone-Scoped Interval Tree Clocks

The Elara Protocol replaces traditional vector clocks with a two-tier temporal ordering system:

Intra-zone: Interval Tree Clocks (ITCs)

Interval Tree Clocks (Almeida et al., 2008) provide the same causal ordering guarantees as vector clocks but with O(log n) space complexity instead of O(n). ITCs work by dynamically splitting and joining identity intervals as nodes enter and leave the system:

- New node joins → receives a split of an existing node's interval
- Node leaves → its interval is available for merger
- Causal ordering is maintained through interval comparisons

For a zone with 100,000 active nodes, the ITC overhead per record is ~40 bytes instead of ~400 KB. This is acceptable even for IoT devices.

Inter-zone: Zone Sequence Numbers

Cross-zone ordering does not require per-node granularity. Each zone maintains a **zone sequence number** — a monotonically increasing counter incremented with each epoch summary. Cross-zone causal ordering uses these sequence numbers:

```
ZoneCausalReference {
    zone_id:        zone identifier
    zone_sequence:  sequence number at time of last sync
    epoch:          epoch number
}
```

A record from Mars that references Earth zone_sequence 45,892 is causally after all Earth records up to that sequence. The overhead is ~20 bytes per cross-zone reference, regardless of how many nodes exist in the referenced zone.

Combined overhead per validation record:

Component	Size
ITC (intra-zone ordering)	~40 bytes

Component	Size
Zone references (inter-zone)	~20 bytes per referenced zone
Total (3 zones referenced)	~100 bytes

Compare to naive vector clocks at 1M nodes: **100 bytes vs. 4 MB** — a 40,000x improvement.

Economics and Incentives (11.10–11.20)

11.10 Free Tier and Economic Accessibility

> **Scope note:** The beat economics described in this section apply exclusively to the public permissionless network. Private network deployments (Section 10.6) — enterprise, government, defense — have no beat involvement. Layer 1 validation is always free, always offline, always sovereign. No network, no beats, no fees.

The question: Section 3.5 shows that any device can validate locally for free. Section 9 describes beat staking for public network witnessing. For participants on the public network without beats, who pays for their records to be witnessed?

Solution: Tiered Economic Model

Layer 1: Always free. No exceptions.

Local validation — generating a keypair, hashing content, signing with a private key, storing on local DAG — costs zero beats. This is a cryptographic operation that runs on the device's own hardware. It is free because it consumes no network resources. Private network deployments operate entirely at this level with organizational witnessing — no beat layer exists.

This is the foundational principle of the protocol. Layer 1 never has a cost, on any deployment model.

Layer 2: Free propagation, paid priority.

When a node syncs to the network, its validation records propagate through the gossip protocol. Basic propagation is free — relay nodes forward record announcements as part of their normal operation (they benefit from a well-connected DAM, so relaying is incentive-compatible).

What costs beats is **priority**: requesting faster propagation, higher witness counts, or guaranteed inclusion in the next epoch summary. Free-tier records propagate and accumulate witnesses organically. Paid-tier records get expedited service.

```
Free tier:
- Local validation: always free
- Network propagation: free (best-effort gossip)
- Witness accumulation: organic (witnesses choose what to attest)
- Epoch inclusion: guaranteed (but not prioritized)
- Storage: included in epoch summaries

Paid tier (beats):
- Priority propagation: faster global reach
- Requested witnessing: specific anchor nodes attest on request
- Priority sync: records synced first during bandwidth-limited windows
- Layer 3 AI: pattern analysis, similarity search, anomaly detection
```

Community witnessing pool:

A percentage of the Community/Governance beat allocation (20% of genesis supply) funds a **public witnessing service** — anchor nodes that attest to free-tier records. This creates a baseline level of network attestation for all participants, regardless of economic status.

Free-tier records propagate for free, get witnessed by community-funded anchors, and accumulate organic witnesses over time. Any user has the same Layer 1 cryptographic proof as a Fortune 500 company. Their Layer 2 trust score grows slower than paid-tier — but it grows.

Earn-by-participation:

Nodes that contribute resources (relay bandwidth, storage, compute) earn beats. The teenager's phone, by relaying other users' records, earns enough beats to request priority witnessing if she ever needs it. The protocol pays its participants.

11.11 zk-SNARK Trusted Setup

The problem: zk-SNARKs (used for PRIVATE and RESTRICTED classifications) require a trusted setup ceremony — a one-time generation of cryptographic parameters (Common Reference String). If the setup is compromised, an attacker can forge proofs. Who performs this ceremony, and how is it decentralized?

Solution: Multi-Party Computation Ceremony + Migration Path

Phase 1: Initial ceremony (pre-mainnet)

The trusted setup uses a **multi-party computation (MPC) ceremony** where multiple independent participants each contribute randomness. The security guarantee: as long as at least ONE participant destroys their random input, the setup is secure.

The Elara Protocol's ceremony will:

1. Include a minimum of 100 independent participants across 20+ countries
2. Accept contributions from anyone (public participation, open-source verification tools)
3. Publish all contributions and verification proofs on the DAM itself
4. Use the Zcash Powers of Tau ceremony model [26] (battle-tested, well-understood)

*Implementation-status note: this MPC ceremony is **contingent on the Groth16 prover/verifier landing** (Section 5.3), which is design-stage and not yet implemented in the runtime. As of this version the ceremony has **not been conducted** and no date is scheduled — Phase 1 ships SHA3-256 hiding commitments, which require no trusted setup. The participant counts and process above specify the design for when zk-SNARK proving is activated; they are not a description of a ceremony in progress.*

Phase 2: Progressive migration to zk-STARKs

zk-STARKs (specified for SOVEREIGN classification; not yet implemented) require NO trusted setup. They are transparent — all parameters are derived from public randomness. The tradeoff is larger proof sizes (~100 KB vs ~288 bytes for SNARKs).

As hardware improves and STARK proof compression advances, the protocol will migrate PRIVATE and RESTRICTED classifications from SNARKs to STARKs:

- **Year 0-3:** SNARKs for PRIVATE/RESTRICTED (compact proofs, trusted setup)
- **Year 3-5:** Dual support (SNARKs and STARKs accepted)
- **Year 5+:** STARKs only (no trusted setup dependency)

The algorithm agility mechanism (Section 4.4) makes this migration seamless — old SNARK-based proofs remain valid; new proofs use STARKs.

Phase 3: Long-term (post-quantum ZKPs)

Research into post-quantum zero-knowledge proofs is active. The protocol's algorithm agility ensures it can adopt new ZKP systems (lattice-based ZKPs, hash-based ZKPs) as they mature, maintaining both quantum safety and zero-knowledge properties.

11.12 Formal Consensus Specification

The gap: "Witness accumulation" is described conceptually throughout this paper but not formally specified. A reviewer expects Byzantine fault tolerance analysis, safety and liveness guarantees, and formal proofs or at minimum, precise algorithm specification.

Specification: Adaptive Witness Consensus (AWC)

The Elara Protocol uses a consensus mechanism designed for its unique properties: threshold settlement with continuous trust accumulation, partition tolerance.

Definitions:

- **Record r** — a validation record on the DAM
- **Witness set W(r)** — the set of nodes that have attested to record r
- **Record trust score T(r)** — a continuous value in [0, 1] representing network confidence in a specific record
- **Node trust score T(n)** — a continuous value in [0, 1] representing a node's accumulated reputation, defined as T(n) = 1 - 1/(1 + attestation_count) where attestation_count is the number of honest attestations by that node. Node trust is earned through sustained honest behavior and cannot be purchased.
- **Witness weight w(n)** — the trust weight of node n's attestation, derived from its node trust score, stake, and age

  Two levels of trust: The protocol distinguishes between **node trust** (a node's reputation, earned over time) and **record trust** (network confidence in a specific record, computed from its witness set). Node trust feeds into record trust — a record witnessed by high-trust nodes accumulates record trust faster. The node-level formula is defined above; this section specifies the record-level formula.

Trust score computation:

```
T(r) = 1 - ∏(1 - w(n) × d(n, W))  for all n in W(r)
```

Where d(n, W) is a **correlation discount** factor in (0, 1] that reduces a witness's marginal contribution based on its independence from other witnesses in the set:

```
d(n, W) = 1 / (1 + Σ corr(n, m) for all m ≠ n in W(r))

corr(n, m) = α × same_org(n,m) + β × same_subnet(n,m) + γ × same_geo(n,m)
             where α=0.5, β=0.3, γ=0.2 (ramped — see below)
```

Witnesses from the same organization or IP subnet contribute progressively less marginal trust. This prevents trust inflation through correlated attestation (e.g., a company running 1,000 witness nodes in one datacenter). The third term weights *geographic* overlap (`same_geo`), not consensus-zone membership: a zone-membership discount would be meaningless because all epoch-seal attestors necessarily share a consensus zone. The shipped runtime sets γ=0.2 with an honest-degradation ramp — γ reads 0 with fewer than 2 distinct geographic buckets (or no more than 2 witnesses), rises linearly with witness-set size, and reaches the full 0.2 at 12 or more witnesses, so a thin witness set never manufactures a false diversity signal. A fully-correlated witness pair reaches the maximum discount α+β+γ = 1.0. (This supersedes the v0.7.3 "audit E2" text that set the third term to 0.0 as a zone-membership discount; the geographic meaning is detailed under "Correlation discount in zone-scoped attestation" below.)

This formulation means: - Zero witnesses → T = 0 (local only) - One high-weight independent witness (w=0.5, d=1.0) → T = 0.5 - Multiple independent witnesses → T approaches 1 asymptotically - Correlated witnesses (same org/subnet) → diminishing returns - Trust never reaches exactly 1 (absolute certainty is impossible)

Layered Consensus Model (v0.7.3)

At production scale (quintillions of records across millions of zones), per-record attestation tracking becomes infeasible — unbounded state accumulation. The protocol uses a three-layer consensus model that preserves the per-record trust formula while achieving bounded-state epoch-level finalization.

Layer 1 — Immediate Validation (per-record, sub-second):

When a record arrives at a zone's witnesses, they perform immediate validation: - Dilithium3 / SPHINCS+ signature verification - Wire format and schema validation - Balance sufficiency (for beat operations) - Entropy score assessment (anti-spam) - Duplicate detection

The per-record trust formula T(r) applies here. Each witness that validates a record contributes to its trust score. Records that pass Layer 1 are assigned status **Pending** and enter the current epoch's candidate set.

Layer 2 — Epoch Finalization (per-batch, anchor-proposed):

Each zone designates anchor nodes (high-trust witnesses with sealing authority, selected by VRF). At adaptive intervals (load-driven, 5–60 s per zone), the anchor proposes an epoch seal:

```
EpochSeal {
    epoch_number,           // per-zone counter (no cross-zone conflicts)
    zone_path,              // hierarchical zone identifier
   merkle_root,           // SHA3-256 Merkle root over all records in this epoch
  record_hashes,       // inline enumeration of included record hashes (bounded; see below)
    record_count,           // number of records in this epoch
    zone_balance_total,     // sum of all account balances in this zone
    previous_seal_hash,     // chain of seals back to genesis
    zone_registry_root,     // Merkle root of all known zones (not full list)
    zone_registry_delta,    // only zone changes since last seal
    vrf_output + proof,     // verifiable seal (Dilithium3-VRF, alg 0x11)
    anchor_dilithium3_sig   // post-quantum signed by proposing anchor
}
```

The anchor proposes which records are in the epoch. Witnesses verify: (a) they have the same records, (b) all records pass Layer 1 validation, (c) the Merkle root matches their independent computation, (d) the zone_balance_total matches their ledger. If verified, witnesses attest to the seal by signing it.

The epoch boundary is deterministic by construction — the anchor's proposal defines what's in the epoch. Records arriving after the proposal go into the next epoch. This eliminates clock-drift consensus failures.

Bounded record enumeration (v0.7.27): The seal's authoritative commitment to its record set is `merkle_root`; the inline `record_hashes` list is a convenience enumeration of the same set, and it is bounded. A seal carries the inline list only when the epoch contains at most `SEAL_INLINE_ENUM_MAX = 96` records — safely inside the per-value metadata bound (Section 11.25.5) — so a proposer can never construct a seal that its own ingestion gate refuses. Busier epochs omit the list, and consuming nodes **derive** the enumeration from their local copy of the zone's epoch window, using the derived set only if its recomputed Merkle root reproduces the sealed `merkle_root`. The same consistency rule guards the inline path: an inline list that does not reproduce `merkle_root` is dropped to empty at parse time — the seal itself remains valid (its signature covers the bytes as transmitted) and the node falls back to the derive path.

A truncated or inconsistent enumeration therefore behaves exactly like an omitted one, and no consumer acts on an enumeration that has not been checked against the sealed root. A node that cannot yet derive the set (still synchronizing its window) degrades gracefully: completeness accounting falls back to the signed `record_count`, and the standard missing-record recovery pulls the window — the same self-healing used for any gap. At mainnet target rates (about 100 records per epoch per active zone) the derive path is the common case by design; the inline path keeps quiet zones — the majority by zone count — on cheap point lookups.

Multi-anchor VRF sealing (v0.7.3): Any anchor node with a VRF key can propose epoch seals — seal authority is not limited to the genesis anchor. Proposer selection uses Dilithium3-VRF **rank gating**: each anchor computes its verifiable output `SHA3-256("elara-vrf-v1" || pk || alpha)` — a domain-tagged hash over the epoch input `alpha` — and its position in the resulting rank order determines *when* it may propose (lower ranks unlock earlier; Section 11.13 specifies the exponential-backoff schedule). There is no pass/fail selection threshold. The anchor's Dilithium3 signature over the output is the proof witnesses verify before attesting, and verifiers independently check that a seal's declared rank matches the proposer's VRF rank and that enough time elapsed for that rank. This is a verifiable, unique, unforgeable selection function — **not a full RFC-9381 VRF**: the output is publicly computable from the public key and input, so unpredictability rests on the input's entropy, not on output secrecy. Same-epoch seal collisions resolve deterministically — the seal with the lexicographically smallest record hash becomes canonical on every node regardless of arrival order. This ensures decentralized seal production without coordination overhead.

A record's status advances to **Sealed** upon inclusion in a proposed epoch seal, and to **Finalized** when the epoch seal accumulates attestations from witnesses controlling at least 2/3 of the zone's staked weight.

Layer 3 — Post-Epoch Challenges (per-record, fisherman):

After an epoch is sealed, fisherman nodes can challenge individual records within it. Challenges operate on epoch-scoped batches:

- **Epoch challenge format:** `{ epoch_number, zone_path, challenged_record_hash, evidence, merkle_proof }`. The challenger must prove the challenged record IS in the epoch by providing a valid Merkle path from the record hash to the epoch seal's `merkle_root`. Invalid proofs are rejected without jury evaluation.
- Jury selected via VRF (existing fisherman mechanism, Section 11.1)
- **Differentiated penalties** if challenge is upheld:
  - **Anchor** (epoch proposer): **100% slash** — the anchor built the epoch and included the bad record. Full accountability.
  - **Attesting witnesses**: **15% slash** — they attested to the batch containing the bad record, but did not construct it. The 15% rate (within the 10-20% design range) reflects shared but limited responsibility.
  - **Non-attesting witnesses**: **0% penalty** — safe to abstain. This ensures witnesses are never punished for honest caution.
- A record surviving the 24-hour challenge window without an upheld challenge advances to **Anchored** status — the highest confirmation level.

Confirmation levels:

Level	Definition
Pending	Layer 1 validated, not yet in an epoch seal
Sealed	Included in an anchor-proposed epoch seal
Finalized	Epoch seal has ≥ 2/3 stake-weighted diverse attestations

Anchored	Finalized + challenge window elapsed with no upheld challenges

Safety guarantee:

An epoch seal attested by witnesses controlling at least 2/3 of staked weight in a zone is considered **zone-settled**. All records within a zone-settled epoch are finalized. No conflicting epoch seal (different Merkle root for the same epoch number) can achieve settlement — the standard BFT bound applies to epoch seals exactly as it applied to per-record attestation. The MESH-BFT diversity-weighted safety theorem (Theorem 1) holds because the diversity function d(n, W) applies to the set of epoch seal attestors.

Liveness guarantee:

As long as >50% of staked weight in a zone is held by honest, online nodes, new epochs will be sealed and attested within one epoch interval. The anchor-proposed model ensures deterministic epoch boundaries even under network asynchrony. Note the gap between the two thresholds: with honest-online weight between 50% and 2/3, the zone keeps *producing* seals but cannot *finalize* them (finalization requires ≥ 2/3 attesting weight) — safety is preserved while finality liveness degrades until enough weight returns.

Partition behavior:

During a network partition, each partition continues sealing epochs independently. Zone-scoped epoch numbers are per-zone, so different partitions' epochs do not conflict. When partitions merge:

1. DAM tips connect across fragments (W.parents = [tip_A, tip_B])
2. Epoch seal chains from both partitions are preserved and verifiable
3. Cross-partition records become verifiable via Merkle proofs from either partition's epoch seals
4. No fork choice — both histories are valid by the mesh property of the DAM

Formal properties (preserved from MESH-BFT):

- **Agreement:** If honest node A considers epoch E zone-settled, all honest nodes in the same zone will eventually consider E zone-settled (assuming partition heals)
- **Validity:** Only records signed by valid keypairs can be included in epoch seals
- **Termination:** Every record propagated to honest nodes will be included in an epoch seal within bounded time (one epoch interval)
- **Partition safety:** Epochs settled before a partition remain settled in all resulting partitions

Formal proofs are provided in the companion paper: "MESH-BFT: Diversity-Weighted Post-Quantum Byzantine Fault Tolerance for Directed Acyclic Meshes" (Vasic, 2026). Three theorems are proven: (1) diversity-weighted safety — correlated adversaries with k ≥ 2 sybil identities sharing a profile cannot achieve double settlement regardless of stake, because the diversity discount reduces their effective stake exponentially; (2) post-quantum security — dual-algorithm signing (ML-DSA-65 + SLH-DSA) ensures consensus safety even if one algorithm is completely broken, with composite security λ_d + λ_s; (3) per-record causal finality — under the layered model, records achieve confirmation in O(1) epoch rounds and O(log n) wall-clock time, independent of total network size. A Monte Carlo safety simulation (`tools/mesh-bft-sim` in the reference implementation: a 64-combination adversary grid — network sizes 10–500, Byzantine fraction up to 0.33, sybil cluster size up to 50 — with zero double-settlement violations) supports the diversity-discount result; it runs at the companion paper's original correlation parameterization (γ = 0.1), and re-validation at the shipped runtime constants (γ = 0.2, correlation cap 1.0) is pending.

Correlation discount in zone-scoped attestation: The γ term weights *geographic* overlap, not consensus-zone membership (all epoch-seal attestors necessarily share a consensus zone, which would make a zone-membership discount meaningless). The shipped runtime sets γ=0.2 over geographic buckets with an honest-degradation ramp: γ reads 0 when fewer than 2 distinct geographic buckets (or no more than 2 witnesses) are present, ramps linearly with witness-set size, and reaches the full 0.2 at 12 or more witnesses — a thin witness set never manufactures a false diversity signal. The same_org (α=0.5) and same_subnet (β=0.3) discounts remain active and prevent trust inflation from correlated witnesses within a zone; a fully-correlated witness pair reaches the maximum discount α+β+γ=1.0. Cross-zone diversity is additionally achieved through the zone registry mechanism and cross-zone Merkle proof verification.

Scalability: Consensus state is bounded by `pending_epochs × subscribed_zones`, not by total records. At 10T records/day across 1M zones with adaptive 5–60 s epochs: 1,440–17,280 epoch seals per zone per day. A witness node in 3 zones tracks ~4,300–52,000 epoch seals/day — trivially manageable.

Slot mutual exclusion (v0.7.6, MESH-BFT Phase 3 Stage 1):

Between a record's arrival at a zone and its inclusion in an epoch seal, an ingesting node must decide whether the record is well-formed under the consensus rules of its creator. A single creator equivocating by emitting two different records that both claim to be its "next operation" is the classical double-spend threat in the DAM setting. The protocol defends against it with a **slot mutex** layered between wire validation (Layer 1) and epoch proposal (Layer 2).

Each v5+ record carries a signed monotonic `nonce ∈ u64` in its wire header, covered by the record's Dilithium3 / SPHINCS+ signature. The pair `(account, nonce)` — where `account = SHA3-256(creator_public_key)` — defines the record's **slot**:

```
slot_key(r) = SHA3-256(creator_public_key(r)) || ":" || to_be_bytes(nonce(r))
```

The slot is a property of the record's creator, not of its routing. It is intentionally zone-independent: DAM routing assigns records to zones by a hash of `record_id`, which differs per record, so the zone cannot participate in slot identity without breaking the mutex. An earlier draft included zone in the slot key; this was rejected during conflict-injection verification (§11.12.1) because two equivocating records always produced distinct `record_id` → distinct zones → distinct slot keys, defeating the mutex entirely.

Ingest-time enforcement. On receipt of a v5+ record r:

1. Compute `slot_key(r)` from the signed nonce.
2. Atomically consult the slot index `S : slot_key → record_id`:
   - If `S(slot_key)` is unset, write `S(slot_key) ← r.id` and accept.
   - If `S(slot_key) = r.id`, accept idempotently (dedup re-ingest).
   - If `S(slot_key) = r'.id` with `r'.id ≠ r.id`, this is an equivocation candidate.
3. On equivocation, the node constructs a **conflict proof** `π = (r', r)`, verifies both signatures over `signable_bytes(·)`, checks `creator_public_key(r') = creator_public_key(r)` and `nonce(r') = nonce(r)`, and rejects r with a `slot_conflict` error. If π verifies, the node also writes a conflict marker `C(slot_key) ← (r'.id, r.id)` to the conflicts index.

Settlement gate. The epoch proposer (Layer 2) consults C before including any Pending record in an epoch seal. A record whose slot is in C is **ineligible for finalization regardless of attestation weight**, even if ≥ 2/3 of zone stake has already attested to it. The anchor drops flagged records from the proposed seal; witnesses independently verify the drop by checking C during Layer 1 validation. This blocks the race in which a Byzantine creator submits r to a majority of validators fast enough to clear attestation before r' arrives — the settlement gate is a hard floor below the attestation majority.

Safety. Under the standard Dilithium3/SPHINCS+ signature unforgeability assumption, only

the holder of `creator_public_key` can emit two records sharing a slot. Once such records exist, the slot mutex guarantees at most one is ever registered in S and the conflict is recorded in C. Therefore no honest epoch can finalize a record whose slot is contested, which preserves the safety theorem of §11.12 (diversity-weighted safety) against the specific attack vector of stake-majority-race equivocation.

Liveness. The mutex is O(1) per record (two RocksDB point lookups into dedicated column families `CF_SLOT_INDEX` and `CF_SLOT_CONFLICTS`). No global iteration, no cross-zone coordination. Honest creators who never reuse a nonce are entirely unaffected; the mutex is transparent to well-formed traffic.

Client requirements. Client signing software must persist `slot_nonce` per identity, increment and durably commit the counter *before* signing the next record, and treat a crash between increment and signing as a one-nonce gap (never a reuse). The browser-node reference client implements this contract as of v0.7.6.

Backward compatibility. Pre-v5 records carry no nonce and are grandfathered — they neither acquire slot entries nor block on them. A schema migration (DB v3 → v4) rebuilds `CF_SLOT_INDEX` under the canonical `(account, nonce)` format on first boot under this spec; pre-existing equivocations uncovered on rebuild are flagged in C.

11.12.1 Conflict-Injection Verification The slot mutex is validated end-to-end by an adversarial test suite (`tests/slot_conflict_injection.rs` in the Rust reference implementation) that drives two signed v5 records sharing `(account, nonce)` through the exact primitives used by the production ingest path, and asserts:

1. The first record claims the slot cleanly.
2. The second is rejected with a proof-verified `slot_conflict`.
3. The conflict marker persists across reads.
4. A settlement query against the conflicted slot returns `true` (gated).
5. An independent slot (same account, different nonce) is untouched.
6. Cross-account slots are disjoint (namespace separation).
7. A tampered signature on the second record is rejected by `ConflictProof.verify()` and does **not** flag the honest slot (no griefing).

This suite was the source of the v0.7.6 slot-key format correction: the earlier 3-tuple `(zone, account, nonce)` key passed unit tests but failed the adversarial tests, surfacing the zone-independence requirement.

11.12.2 Super-Seal Consolidation (v0.7.7) **The problem:** At 1M zones with adaptive 5–60 s epochs, each zone produces 1,440–17,280 epoch seals per day. Over a year of operation, a light client that wants to verify the history of a single zone must walk hundreds of thousands of seals. The seal chain is compact (one Dilithium3 signature per seal, ~4 KB including witness attestations) but the walk is linear in the number of epochs. Light-client verification becomes a bandwidth and compute burden.

Solution: N-epoch aggregation into a super-seal.

Every `super_seal_interval` epochs (default 64, chosen so roughly one super-seal is emitted per zone per hour at the 60 s epoch ceiling — more frequently under load — but the interval is a governed parameter), the current anchor proposes a **super-seal**:

```
SuperSeal {
    zone_path,
    start_epoch,             // first sealed epoch in this super-seal
    end_epoch,               // last sealed epoch (inclusive)
    super_merkle_root,       // Merkle root over the N epoch seal hashes
    aggregate_record_count,  // sum of record_count across the N seals
    aggregate_balance_total, // zone_balance_total at end_epoch
    previous_super_seal_hash,
    vrf_output + proof,      // Dilithium3-VRF, alg 0x11
    anchor_dilithium3_sig    // dual-signed
    anchor_sphincs_sig
}
```

A light client verifying a record `r` that was sealed in epoch `e ∈ [start_epoch, end_epoch]` follows this pipeline:

1. Fetch super-seal containing epoch e. Verify anchor signatures and chain it to its predecessor super-seal.
2. Fetch epoch seal `e` with a Merkle proof against `super_merkle_root` (sibling hashes: `log2(N) × 32 bytes`).
3. Fetch record `r` with a Merkle proof against epoch seal e's `merkle_root` (standard §11.22.1 path).

At `N = 65`, the light client verifies one super-seal plus `log2(65) ≈ 7` sibling hashes instead of walking 65 full epoch seals — a 65× reduction in signatures to verify and a 9× reduction in wire bytes fetched for historical verification. Super-seal walking remains `O(log n)` in the total number of epochs per zone.

Settlement rule. A super-seal is zone-settled when attested by at least 2/3 of the zone's stake-weighted witnesses for the `end_epoch`. Because super-seals cover already-finalized epochs, attestation is fast: witnesses who already attested each constituent epoch seal cryptographically re-attest the aggregation in one signature. Non-attesting witnesses incur no penalty (same rule as §11.12 epoch seals).

Hostile-recovery path. If no honest anchor produces a super-seal for `> 2 × super_seal_interval` epochs (censorship scenario), witnesses fall back to the raw epoch seal chain. Super-seals are an acceleration layer, not a safety layer — removing them costs bandwidth, never correctness.

Testnet status: Super-seal consolidation is implemented in the Elara Runtime (public `v0.2.0` release, `SUPER_SEAL_INTERVAL = 64` in `epoch.rs`). Light clients consume super-seals via the public `/super-seal/{zone}/{epoch}` endpoint.

11.12.3 Aggregator Chain and Bounded View Depth (v0.7.8) **The problem.** §11.12's multi-anchor VRF sealing makes any staked anchor a candidate proposer for a given (zone, epoch). This is safe (duplicate seals are rejected at ingest) but weakly live: if the first-to-propose happens to be offline, malicious, or behind a partition, the epoch stalls while every other anchor polls the same VRF lottery with no ordering. At mainnet scale (10K+ nodes, 1M zones, adaptive 5–60 s epochs), the expected stall per silent proposer compounds — the epoch does not seal until an honest anchor happens to observe the gap, re-evaluate, and race to emit. There is no escalation schedule, no rank, and no bound on how many rounds that race can take.

Solution: a log-bounded, RTT-adaptive aggregator chain. Per (zone, epoch) the protocol deterministically ranks up to `MAX_VIEW_DEPTH = 7` eligible aggregators, and unlocks them one at a time under an exponential-backoff schedule. Rank-0 holds the exclusive right to propose at epoch start; each higher rank becomes eligible after a bounded wait. If the full ranked ladder times out, the zone enters a **stuck** state and may be unstuck by a cross-zone **global quorum**

seal (Part C). A **liveness-failure proof** (Part D) imposes a small slashing penalty on a rank-0 aggregator that fails to propose during its exclusive window — turning rank-0 silence from a free bug into a priced hazard.

All rank math is integer-only (u128 hash scoring, `u64::isqrt` stake weighting). No floating point anywhere in the critical path — determinism must hold byte-identically across x86, ARM, and WebAssembly validators.

Part A — Ranked proposal (stake-weighted VRF sorting).

Define the **chained beacon**

```
beacon(zone, epoch) = SHA3-256(prev_seal_hash(zone) || u64_be(epoch) || zone_path)
```

where `prev_seal_hash(zone)` is the hash of the most recently registered epoch seal for that zone (32 zero bytes at genesis). Chaining the beacon to the previous seal hash binds rank selection to a value no single node controls at proposal time — a proposer cannot bias its own rank by choosing the seed.

For the active set of staked, VRF-registered anchor identities `{(id_i, stake_i)}` — anchors that have published a VRF public key (Section 11.12); staked identities without a registered VRF key are not in the ranked set — each identity's **priority score** is

```
score(id_i) = u128_from_be_bytes(SHA3-256(beacon || zone_path || id_i)[0..16])
                / max(1, isqrt(stake_i))
```

with ties broken by lexicographic order on `id_i`. The identities sorted in ascending order of `score` form the **aggregator chain** for this (zone, epoch). Rank 0 is the aggregator with the smallest score; rank k = the k-th position in the sort (0-indexed). The chain is truncated to the first `MAX_VIEW_DEPTH = 7` ranks; identities past rank 6 play no role in this epoch.

Why integer-sqrt weighting. A low-stake sybil farm must not dilute rank-0 away from honest large stakeholders. Dividing by `isqrt(stake)` (the same √-dampening the beat economy uses throughout) means a 100× larger honest stake has roughly 10× the rank-0 probability of a single sybil, not 100×. This balances "stake matters" against "no single huge stake dominates" and prevents rank-0 capture by any single large staker.

Part B — Exponential-backoff view schedule.

Let `epoch_start_ts` be the wall-clock moment at which the zone's rank-0 proposal window opens (populated when the previous epoch's seal is registered). Let `elapsed = now - epoch_start_ts`. Let `base_timeout_ms` be the per-zone adaptive timeout defined below. Rank k becomes eligible to propose once

```
elapsed ≥ (2^k − 1) · base_timeout_ms
```

rank	unlock time
0	0 (immediately)
1	base_timeout
2	3 · base_timeout
3	7 · base_timeout
4	15 · base_timeout
5	31 · base_timeout
6	63 · base_timeout

The schedule is exhaustive: after `(2^MAX_VIEW_DEPTH − 1) · base_timeout_ms = 127 · base_timeout_ms` the last rank's window has elapsed and the zone is considered **stuck** (see Part C).

A node evaluates, for itself, the predicate

`rank_of(self) ≤ allowed_rank(elapsed, base_timeout_ms)`

where `allowed_rank` returns the maximum `k` for which the unlock threshold has elapsed, capped at `MAX_VIEW_DEPTH − 1`. If the predicate holds *and* no seal is yet registered locally for `(zone, epoch)`, the node proposes. The node's rank is embedded in the emitted seal as `aggregator_rank: u8` so that any receiving witness can verify the schedule was respected.

Why depth 7, not f + 1. Classical BFT view changes allow up to `f + 1` rotations. At mainnet scale `f ≈ 1000` per zone, and `f + 1`-deep chains are useless: timeouts approach zero and every transient jitter triggers a spurious view change, eating throughput. Bounding depth at 7 gives

`Pr[7 consecutive malicious aggregators] = (1/3)^7 ≈ 0.046 %`

under the honest-majority assumption (< 1/3 Byzantine). The rare case falls through to cross-zone escalation, which is designed to be expensive and infrequent.

Adaptive base_timeout. The protocol derives `base_timeout_ms` from a per-zone RTT estimator maintained over gossip attestation round-trips:

`base_timeout_ms = clamp(2 · p95_zone_rtt_ms, 1000, 600000)`

Clamped to `[1 s, 10 min]` at verification (proofs carrying a `base_timeout_ms` outside this range are rejected). The shipped proposer computes its schedule with a **5 s floor** — `base_timeout = max(2 · p95_rtt, 5 s)` (`zone_rtt.rs`) — so a zone whose witnesses are tightly clustered (e.g., mostly EU, `p95_rtt = 120 ms`) runs with `base_timeout = 5 s` → full rank ladder exhausted in 127 · 5 s ≈ 635 s (~10.6 min); a zone with global spread (`p95 = 900 ms`) also floors at 5 s. The 1 s figure is the wire-accepted minimum, not the shipped proposer floor. The clamp prevents degenerate behavior on both ends: a misreporting local-only zone cannot drive the schedule to zero, and a zone without attestation data yet defaults to the maximum so honest nodes retain room to propose.

Part B.2 — Rank verification at attestation time.

When a witness receives a proposed seal, it computes `beacon(zone, epoch)` from its local state and derives the chain. The seal's declared `aggregator_rank` must satisfy three independent checks:

1. `0 ≤ aggregator_rank < MAX_VIEW_DEPTH`.
2. The chain at rank position `aggregator_rank` equals the seal's proposer identity (rank-forgery check).
3. `elapsed_ms ≥ (2^aggregator_rank − 1) · base_timeout_ms` (time-gate check — no rank-jumping).

If any check fails, the seal is rejected before attestation. Rank verification is skipped in two narrow cases: (a) fresh restart with no local `epoch_start_ts`; (b) catch-up / partition-merge where the witness does not share the proposer's stake view. In both cases the witness cannot reconstruct the schedule deterministically; safety is unaffected because the seal's Dilithium3 signature and epoch-body checks continue to enforce the core invariants.

Bootstrap carve-out. Where the active staker set has fewer than three distinct identities, the rank ladder degenerates — the genesis authority alone proposes at rank 0, and no non-genesis rank exists. This allows a fresh network to begin producing seals before the rank mechanism is meaningful.

Part C — Cross-zone escalation (global quorum seal).

If `elapsed_ms > 127 · base_timeout_ms`, every rank window in the chain has elapsed and the zone is **stuck**. At this point the zone cannot produce a seal through the normal per-zone

rank path — even rank 6's exclusive window is past. Cross-zone escalation permits any staker in a *different* zone to emit a **global quorum seal** for the stuck `(zone, epoch)`:

```
GlobalQuorumSeal {
    stuck_zone,                  // zone that failed to seal locally
    stuck_epoch,                 // epoch number that remained unsealed
    emitter_zone,                // zone of the emitting staker
    previous_seal_hash,          // stuck_zone's previous seal hash
    observed_base_timeout_ms,    // must be within [1000, 600000]
    observed_elapsed_ms,         // > (2^MAX_VIEW_DEPTH - 1) · observed_base
    emitted_at,                  // wall-clock proposal time
    vrf_output + proof,          // Dilithium3-VRF (alg 0x11) by emitter
    dilithium3_sig               // by the emitter
}
```

Seven ordered verification checks apply at ingest:

1. `emitter_zone ≠ stuck_zone` — the emitter must live outside the stuck zone (a failed zone cannot rescue itself with the same nodes that just timed out).
2. `stuck_epoch` equals the next unsealed epoch for `stuck_zone` (not a backfill, not a future claim).
3. `previous_seal_hash` matches `stuck_zone`'s locally registered previous seal.
4. The rank schedule is demonstrably exhausted: `observed_elapsed_ms > (2^MAX_VIEW_DEPTH − 1) · observed_base_timeout_ms`, and `observed_base_timeout_ms ∈ [1000, 600000]`.
5. The emitter's VRF public key is registered in `CF_VRF_REGISTRY`; `verify_vrf(emitter, α = global_seal_alpha(stuck_zone, stuck_epoch), proof)` succeeds, where `global_seal_alpha` is a distinct domain-tagged input so per-zone and global VRF inputs cannot collide.
6. The emitter is a registered staker with positive stake in `emitter_zone` (anti-freeloader).
7. Dilithium3 signature over the canonical bytes verifies against the emitter's registered key.

Settlement of global seals. A global seal is **cross-zone settled** when witnesses outside `stuck_zone` attest to it with combined stake satisfying `3 · num ≥ 2 · den`, where `den = max(zone_stakes[z])` over non-stuck zones and `num` sums stake-weighted attestations from distinct non-stuck zones. The denominator is the maximum per-zone stake (*not* the sum) because the protocol replicates the per-zone staker list across zone registers under the canonical `register_stakes_from_ledger` convention; summing would double-count and make the 2/3 threshold unreachable. Attestations originating from `stuck_zone` are dropped from both `num` and the diversity count so a malicious majority within the stuck zone cannot finalize its own rescue.

On cross-zone settlement the stuck zone's `latest_epoch` advances and `epoch_start_ts` resets — the zone is unstuck, and normal per-zone sealing resumes for the next epoch.

Why cross-zone. The honest-majority assumption is global, not per-zone. A single zone can be captured by a transient majority of Byzantine stake (e.g., a regional outage silencing honest witnesses) yet the network as a whole remains > 2/3 honest. Cross-zone escalation converts that global surplus into local liveness: so long as at least two other zones retain honest majorities, stuck zones can be rescued without suspending the protocol.

Part D — Liveness-failure slashing.

Silent rank-0 aggregators cost the network one base-timeout of latency per epoch — free, if unpriced. The protocol prices this hazard with a 1 % stake slash on any aggregator that fails to propose within its exclusive rank-0 window (the interval from `epoch_start_ts` to `epoch_start_ts + base_timeout_ms`). A **liveness-failure proof** bundles ≥ 1 Dilithium3 attestations over a domain-tagged canonical message:

```
liveness_failure_msg(offender, zone, epoch, base_timeout_ms, epoch_start_ts_ms)
    = DOMAIN_TAG
      || offender
      || zone_path
      || u64_be(epoch)
      || u64_be(base_timeout_ms)
      || u64_be(epoch_start_ts_ms)
```

Verification rejects empty proofs, proofs with `base_timeout_ms` outside the `[1000, 600000]` clamp, self-attestation (the offender signing its own failure), duplicate signer pubkeys, and any bad signature. A layered `verify_with_stakers` check ensures the attestation set controls $3 \cdot \text{num} \geq 2 \cdot \text{den}$ of the zone's staked weight under integer-safe u128 math so that large stakes do not overflow.

On verification success, the protocol slashes 1 % of the offender's largest stake (`LIVENESS_SLASH_PERCENT = 0.01`) — materially lighter than the 25 % equivocation slash because liveness failure is offline-rational, not byzantine-intentional, and a disproportionate penalty would chill honest participation. The slash is dedup-keyed on `(offender, zone, epoch)` so a retransmitted proof does not compound the penalty.

Liveness slashing is genesis-gated during bootstrap (same rule as equivocation slashing) to protect early operators from a zero-stake topology in which one partition could mass-prove every other node. Post-genesis, the mechanism applies universally to any rank-0 aggregator.

Part E — Safety.

The rank mechanism does not change the safety invariant of §11.12: at most one seal per `(zone, epoch)` can be finalized, because the settlement threshold still requires ≥ 2/3 (66.67%) zone-stake-weighted diverse attestations on a single seal and the slot-mutex/conflict-proof apparatus still prevents double-attestation by honest nodes.

Rank-verification failure is a local rejection; it never attests two different seals. A Byzantine proposer forging rank-0 (e.g., declaring `aggregator_rank = 0` while the chain places it at rank 5) is detected by every witness running the same deterministic derivation from the chained beacon — the seal is rejected before any attestation.

Global quorum seals inherit the same safety property: check 1 (`emitter ≠ stuck_zone`) plus the cross-zone denominator convention ensure the non-stuck majority alone must concur, and any zone-internal capture cannot outvote it.

Part F — Liveness (log-bounded finality).

Under partial synchrony with message delays bounded by `p95_zone_rtt_ms` after GST, and under the honest-majority assumption (< 1/3 stake-weighted Byzantine, no correlation beyond the zone boundary), the protocol guarantees that every epoch for every live zone is eventually sealed, with an upper bound on finality time of

```
finality_time ≤ Σ_{k=0}^{MAX_VIEW_DEPTH-1} base_timeout · 2^k
              = (2^MAX_VIEW_DEPTH − 1) · base_timeout
              = 127 · base_timeout
```

for the per-zone path, plus at most one additional cross-zone round for the rare case (probability $\approx 5 \cdot 10^{-4}$ under uniform Byzantine sampling) in which the full per-zone ladder is Byzantine. A formal proof of the log-bounded liveness theorem is provided in §5.6, Theorem 4, of the companion paper "MESH-BFT: Diversity-Weighted Post-Quantum Byzantine Fault Tolerance for Directed Acyclic Meshes" (Vasic, 2026).

Concrete numbers. A zone with tight EU-only witnesses (`p95 = 120 ms`) runs the shipped proposer floor, `base = max(5000, 240) = 5000 ms`, so in the absolute worst case of a Byzantine rank ladder, the stuck zone escalates after $127 \cdot 5$ s ≈ 635 s (~10.6 min) and unsticks in the

next cross-zone round; the $127 \cdot$ base bound with a 1 s base is the wire-accepted illustrative minimum, not shipped behavior. In typical honest operation rank-0 proposes within ~1 s of epoch open and observed finality is indistinguishable from the pre-v0.7.7 seal path — the rank machinery only imposes latency when a proposer is actually silent.

Testnet status. The aggregator chain is implemented in the Elara Runtime (MESH-BFT Phase 3 Stage 3). Rank gating, chained beacon, rank verification, cross-zone escalation, and liveness-failure proofs ship in the public `v0.2.0` release (`aggregator.rs`). A testnet validated two adversarial scenarios that together exercise the full mechanism: (a) `byzantine-aggregator-offline` — rank-0 silent on every node, the chain rotates through rank-1+ and every node advances ≥ 9 epochs with conservation preserved; (b) `byzantine-all-ranks-malicious` — all 7 ranks silent in zone 0, the cross-zone escalation path unsticks zone 0 from baseline to min-epoch 9 while zone 1 advances to 12. Both runs maintain `ledger_supply` $= 10^{19}$ base units across all nodes.

> **Implementation status:** AWC/MESH-BFT is implemented in the Elara Runtime (Rust, ~350k LOC, 5,700+ tests). A multi-node testnet validated: zone-based consensus with correlation discounting (zone_count=2 forced), dual-signature verification (Dilithium3 + SPHINCS+), gossip propagation (p50=250ms, p90=500ms), ZK proof scaffolding with a fail-closed verifier, beat transfers and staking, checkpoint-based startup (31µs replay), cross-zone transfers, and failure recovery. All nodes maintained identical supply (10B beats) across kills, restarts, partitions, and re-genesis, with 12K–21K settled records per node across 5+ days of continuous operation. The layered consensus model — epoch sealing, multi-anchor VRF proposals, differentiated fisherman penalties, peer liveness probes, zone-scoped gossip, cross-zone transfers, batch consensus processing, per-peer attestation watermarks, and priority eviction — is implemented.

11.13 Dispute Resolution

The gap: The paper states that conflicting claims are "preserved, not resolved by deletion." But preservation is not resolution. When two entities claim authorship of the same work, someone or something must eventually decide.

Solution: Three-Tier Dispute Resolution Framework

Tier 1: Automated Resolution (no human involvement)

The protocol automatically resolves clear-cut cases:

- **Temporal priority with causal proof:** If record A has a causal anchor (DAM-verifiable timing) earlier than record B, and both claim the same content hash, A is marked as the **prior claim**. B is not deleted but is annotated as a **subsequent claim**. This happens automatically.
- **Incremental chain vs. single hash:** If one claimant has a chain of drafts (Section 11.7, Tool 1) and the other has only the final hash, the chain-holder receives an automated **provenance score** boost. Not a verdict — a weighted signal.
- **Identical key families:** If both claims come from keys in the same organizational identity, it's an internal matter. The protocol flags it but takes no action.

Tier 2: Community Arbitration (decentralized human judgment)

For cases that automated analysis cannot resolve, the protocol supports **arbitration panels:**

- Any party to a dispute can invoke arbitration by staking beats (refundable if the claim is upheld)

- A panel of 5-11 arbitrators is randomly selected from a pool of staked, reputable nodes with HUMAN entity type
- Arbitrators review the evidence on the DAM: timestamps, causal chains, witness patterns, provenance scores, incremental history
- Majority vote produces an **arbitration record** on the DAM — a non-binding recommendation that carries significant trust weight
- Losing party can appeal once (new panel, larger size)

Arbitration records are not protocol enforcement — they are advisory. But they carry weight: a record with a favorable arbitration result receives a trust boost; an unfavorable one receives a penalty.

Tier 3: Legal Integration (real-world enforcement)

For disputes that require legal force (copyright infringement, patent priority, contractual obligations), the protocol provides:

- **Court-admissible evidence export:** A standardized format that packages a validation record with its full causal chain, witness attestations, Merkle proofs, and timestamp verification into a document that legal systems can interpret. Designed in consultation with digital forensics standards (ISO 27037).
- **Expert witness protocol:** Anchor nodes can generate signed attestations explaining the technical meaning of DAM records in language suitable for legal proceedings.
- **Jurisdiction mapping:** Validation records can optionally include jurisdiction metadata, enabling creators to indicate which legal system they consider authoritative for disputes.

The protocol does not replace law. It provides the strongest possible evidence for legal systems to use. A DAM record with causal anchoring, thousands of witnesses, and an incremental creation chain is the digital equivalent of a notarized document, a chain-of-custody record, and a witness testimony — combined.

11.14 Network Topology and Peer Discovery

The gap: The paper describes what happens when nodes communicate but never specifies how nodes find each other. Without a peer discovery mechanism, the network cannot form.

Solution: Hybrid Discovery with Kademlia DHT

The Elara Protocol uses a three-layer peer discovery system:

Layer A: Bootstrap Nodes

Every Elara client ships with a configurable bootstrap-node list — the seed peers a new node dials to be introduced to the network. In the current public release this list is **empty by default** (`TESTNET_SEED_PEERS = []`): there are no foundation-run public endpoints, and operators set their own seeds (`seed_peers` / `ELARA_SEEDS`; see `docs/JOIN-DEVNET.md`). Seed nodes serve one purpose — introducing new nodes — and are not privileged in any other way.

```
Bootstrap list (example):
  bootstrap-eu.elara.network:4001
  bootstrap-us.elara.network:4001
  bootstrap-asia.elara.network:4001
  bootstrap-africa.elara.network:4001
```

The bootstrap list is updatable through protocol governance. If all bootstrap nodes go offline simultaneously, nodes that already know peers continue operating — bootstrap is only needed for first contact.

Layer B: Kademlia DHT (Distributed Hash Table)

Once connected to at least one peer, nodes join a Kademlia-based DHT (a widely deployed algorithm in peer-to-peer networks). Kademlia provides:

- O(log n) lookup for any node in the network
- Self-healing: the routing table automatically repairs when nodes leave
- Resistance to targeted attacks: no single node is critical for routing
- NAT traversal via hole-punching (UDP-based, with relay fallback)

Each node maintains a routing table of ~20 × log2(N) entries. For a million-node network, this is ~400 entries — negligible memory.

Layer C: Local Discovery

For devices on the same local network (IoT deployments, mesh networks), the protocol uses mDNS/DNS-SD (multicast DNS / Service Discovery) for zero-configuration local peer discovery. A sensor and its gateway find each other without any internet connectivity.

For Bluetooth-capable devices, BLE advertisements enable peer discovery within ~100 meters. This enables the mesh-networking scenarios described in the Emergency Protocols (Section 12.3).

Gossip Protocol for Record Propagation:

Once peers are discovered, validation records propagate via an epidemic gossip protocol:

1. Node creates or receives a new record
2. Node selects √n random peers from its routing table (where n = number of known peers)
3. Node sends compact announcements (record ID, content hash, creator, timestamp, zone) to selected peers (~250 bytes each)
4. Recipients request full records only for announcements they haven't seen (pull-on-demand). ~10x bandwidth reduction vs full-record forwarding

With √n fan-out, theoretical propagation completes in ~2-3 rounds. In practice, duplicate messages, network latency, and partial peer overlap increase this to ~6-10 gossip rounds for 1 million nodes — typically under 15 seconds on Earth-zone networks.

Zone-Scoped Gossip Filtering (v0.7.3):

Records are only relayed to peers subscribed to the record's zone. This prevents bandwidth waste — a node subscribed to `medical/eu/west` does not receive or relay `iot/manufacturing/toyota` records. Two exceptions bypass zone filtering: epoch seals and beat operations are relayed globally, as they affect cross-zone state (zone registry, supply conservation). Nodes with empty zone subscriptions accept all records (backward compatibility with pre-zone nodes).

Content-Routed DHT above 100 peers (v0.7.7):

Flood-gossip with √n fan-out works well for small networks but becomes wasteful above 100 peers: every record reaches every subscribed node regardless of whether that node stores or serves it. At 10K+ nodes per zone, the protocol uses a **content-routed overlay** layered on the same Kademlia DHT that handles peer discovery.

Each record's `record_id` (SHA3-256 of the wire-canonical body) is routed to a deterministic responsibility set by XOR-distance in the DHT keyspace:

```
responsible_nodes(record_id) = k-closest Kademlia peers to record_id
                               where k = replication_factor (default 8)
```

A record is gossiped once to its k responsible nodes instead of flood-forwarded. Consumers retrieve records by issuing Kademlia GET lookups against `record_id`: the lookup converges in `O(log n)` hops to a responsible replica. Responsibility rotates naturally as peers join and leave — Kademlia's self-healing property applies unchanged.

Fallback rule. Below 100 peers in a zone, flood-gossip remains active (the overhead of content routing exceeds its savings at small scale). At 100+ peers, the overlay activates automatically. The threshold is not a governance parameter — it is a per-node heuristic so partitioned or bootstrapping nodes gracefully degrade.

Safety. Content routing changes only the propagation topology, not the consensus model. Epoch seal attestation, slot mutex (§11.12 v0.7.6), and cross-zone proofs (§11.22.1) all operate identically. A record that arrives via content-routed DHT is indistinguishable from one that arrived via flood-gossip — ingest validation is invariant.

Testnet status: Content-routed placement above 100 peers is implemented in the Elara Runtime (public `v0.2.0` release; content-routing threshold default 100 in `config.rs`). The testnet runs under the threshold, so flood-gossip is the active path; overlay testing is pending testnet expansion.

Peer Liveness Probes (v0.7.3):

Nodes periodically probe peers via `POST /probe` — a three-in-one protocol that combines liveness checking, record exchange, and trust scoring in a single round-trip. The probe interval scales with network size:

```
interval = 300s × √(peers / 5), clamped to [60s, 3600s]
```

At 5 peers: 300s. At 20 peers: 600s. At 500 peers: ~3000s. This keeps the network alive with zero user traffic while avoiding probe storms in large peer sets. Nodes that fail 3 consecutive probes are marked offline and deprioritized in gossip fan-out.

11.15 Zero-Knowledge Proof Feasibility on Constrained Devices

The problem: Generating a zk-SNARK proof requires significant computation:

Device	zk-SNARK Generation	RAM Required
Modern laptop	1–5 seconds	1–4 GB
Flagship smartphone (2026)	3–10 seconds	500 MB–2 GB
Budget smartphone ($30)	15–60 seconds	200 MB–1 GB
Raspberry Pi 4	10–30 seconds	500 MB–2 GB
ESP32	**Infeasible**	520 KB total RAM

The Kenya teenager's $30 phone can generate a proof, but it takes up to a minute and drains the battery. An ESP32 cannot generate a proof at all. This limits PRIVATE classification to devices with meaningful compute.

Solution: Layered Privacy by Device Capability

Capable devices (phones, laptops, servers): Generate ZK proofs locally. PRIVATE, RESTRICTED, and SOVEREIGN classifications fully supported.

Constrained devices (IoT, ESP32): Three options for privacy-preserving validation:

1. **Gateway-delegated proof generation** — the constrained device sends content to a trusted gateway over a secure local channel. The gateway generates the ZK proof. Trust boundary: the gateway sees the content (acceptable within a single deployment, e.g., factory sensors reporting to a factory server).
2. **Deferred proof generation** — the device validates as PUBLIC initially (just the hash), then a more capable device generates a ZK proof later, converting the record to PRIVATE. The record's DAM position is preserved — it does not lose its timestamp.

3. **Symmetric encryption fallback** — for IoT deployments where even the gateway should not see individual readings, the device encrypts the reading with a pre-shared key before hashing. The hash is of encrypted content. Decryption requires the key. This is not zero-knowledge in the cryptographic sense, but it provides practical privacy for the constrained device use case.

Budget smartphones — optimization path:

ZK proof generation on smartphones is an active area of research and optimization. Specific measures for the Elara Protocol:

- Use Groth16 (most compact and fastest zk-SNARK) for the standard proof circuit
- Pre-compute the proving key once and cache it (~50 MB)
- Offload to GPU/NPU on devices that support it (most 2025+ smartphones have ML accelerators that can be repurposed)
- As STARK compression advances, migrate to STARKs which have simpler prover algorithms

The protocol's algorithm agility ensures that as ZKP technology improves, constrained devices gain privacy capabilities without protocol changes.

11.16 Censorship Resistance

The threat: A government orders all ISPs within its jurisdiction to block Elara Protocol traffic. Or mandates that all domestic nodes refuse records from certain creators (political dissidents, journalists, specific organizations). State-level censorship is the most powerful adversary the protocol faces.

Defense Layer 1: Traffic Obfuscation

The protocol supports **pluggable transports** — the same concept used by Tor to operate in jurisdictions with restrictive internet policies:

- **Domain fronting:** Elara traffic masqueraded as HTTPS requests to permitted domains (cloud providers, CDNs)
- **Steganographic encoding:** Validation records embedded in innocent-looking traffic (images, video calls, DNS queries)
- **Bridge relays:** Unlisted relay nodes operated by volunteers outside the censoring jurisdiction, accessible via out-of-band key exchange

These are not theoretical — they are proven techniques deployed at scale by the Tor Project and Signal messenger.

Defense Layer 2: Partition Resilience (Already Built-In)

The DAM's partition tolerance (Section 7.3) means censorship IS a partition. If a government blocks cross-border traffic:

1. The domestic zone continues operating independently
2. Domestic validations remain cryptographically valid
3. When the censorship lifts (regime change, policy reversal, VPN access), the zones merge
4. Nothing is lost — the domestic branch of the DAM is fully preserved

A government can slow the network. It cannot kill records that already exist on the DAM, and it cannot prevent domestic validation from continuing.

Defense Layer 3: Mesh Networking Fallback

In extreme censorship scenarios (internet shutdown), devices can form local mesh networks:

- **Bluetooth mesh:** Phone-to-phone, ~100 meter range, chain across a city

- **LoRa mesh:** 10+ km range, low bandwidth but sufficient for compact validation payloads (full PQC records require gateway relay)
- **Sneakernet:** Physical transfer of DAM data via USB drives, SD cards — the protocol supports offline sync by design

Records validated during an internet blackout propagate when any node in the mesh eventually reaches the global network. The DAM is patient. It can wait.

Defense Layer 4: Geographic Distribution of Anchor Nodes

The protocol requires anchor nodes on at least 3 continents for the decentralization threshold (Section 11.4). No single government can compel all anchor nodes to comply. Even if a government seizes all domestic anchor nodes, the global DAM continues — and the domestic zone's records are already replicated internationally.

11.17 Beat Supply Model

> **Scope: Public permissionless network only.** This section applies to the public permissionless network. Private deployments have no beat supply.
>
> **Scope.** This section describes the public network's beat supply model — a fixed-supply conservation system — and its security implications. See Section 9 for the protocol-level economic summary.

The public network's beat supply model is designed as a **conservation system** — beats circulate between producers (witnesses) and consumers (record submitters) rather than being continuously minted. This design choice has specific security implications:

- **No inflationary dilution attack** — because supply is fixed, an attacker cannot devalue existing stakes by inflating the supply
- **MEV prevention** — witness attestation order does not affect outcome (trust scores are order-independent — the same witnesses produce the same trust score regardless of when they attest). This eliminates Maximal Extractable Value by design. There is nothing to extract from reordering.
- **No gas fee exploitation** — because the protocol does not charge per-transaction gas fees, there is no fee market to manipulate

Genesis allocation. The fixed supply (10 billion beats) is partitioned at genesis into six internal accounting pools. Only the bootstrap pool has an active distribution path (the participation faucet, Section 9.5); the others are reserved genesis balances. Consistent with Section 9.2, **no pool is sold, listed, auctioned, or made tradeable** — this is internal accounting, not a token distribution.

Pool	Share	Distribution path
Bootstrap	30%	**Active** — earned by participating nodes via the faucet (§9.5)
Development	20%	3-of-5 multisig, protocol development; no market
Community	20%	Conviction-voting-controlled treasury (§10.3–10.4); no market
Team	15%	Reserved genesis balance — **no active distribution path**
Contributors	10%	Reserved genesis balance — **no active distribution path**
Reserve	5% (+ rounding remainder)	4-of-5 multisig emergency reserve; no market

The split is computed with exact integer arithmetic and a conservation check (the six pools sum to the total supply — no minting beyond genesis). Because beats are never tradeable, these shares confer no monetary claim; they bound how much internal staking/accounting weight each pool may hold, not a financial position.

11.18 Protocol Upgrade Mechanism

The gap: A protocol designed to outlive its creators must evolve. New cryptographic algorithms, new ZKP systems, new device types, unforeseen requirements. How does the protocol upgrade without breaking the network?

Solution: Semantic Versioning with Soft Fork Default

Version field in every record:

Every validation record includes a protocol version number. Nodes advertise the versions they support. This enables:

Soft forks (backward-compatible changes): - New optional fields in validation records - New classification levels - New entity types - Optimized encoding formats

Soft forks require no coordination. New nodes produce new-format records. Old nodes ignore fields they don't recognize. Both coexist on the DAM. Governance vote recommends adoption; individual nodes upgrade at their own pace.

Hard forks (breaking changes): - Changes to the consensus mechanism - New required fields in validation records - Deprecation of cryptographic algorithms

Hard forks require governance approval (>67% supermajority per Section 10.3) and follow a strict process:

```
1. Proposal published (with reference implementation)
2. 90-day discussion period
3. Governance vote
4. If approved: 180-day transition window
5. During transition: both old and new formats accepted
6. After transition: old format deprecated (but old records remain valid)
```

Emergency upgrades (critical security patches):

If a cryptographic algorithm is broken or a critical vulnerability is discovered, the protocol supports **emergency governance** — a fast-track process:

1. Security advisory published by any anchor node
2. Emergency vote: 48-hour window, >75% supermajority required
3. If approved: 30-day transition (vs. normal 180 days)
4. Anchor nodes enforce the new version; other nodes have 30 days to upgrade

The emergency mechanism is intentionally rare and high-threshold. It exists for true existential threats (e.g., quantum computing breaks Dilithium earlier than expected), not for feature additions.

Algorithm deprecation lifecycle:

```
ACTIVE     → algorithm used for new signatures
LEGACY     → algorithm accepted for verification, not recommended for new signatures
DEPRECATED → algorithm accepted for verification of old records only, rejected for new
ARCHIVED   → algorithm documented in protocol history, old records still verifiable
             through algorithm agility (Section 4.4)
```

No algorithm is ever deleted from the protocol's specification. A record signed with Dilithium3 in 2026 must be verifiable in 3026 — even if Dilithium3 was deprecated in 2050. The verification code for every algorithm ever used is preserved in the protocol's reference implementation, explicitly tagged as archival.

11.19 Spam and Denial-of-Service Prevention

The attack: Layer 1 validation is free. An attacker generates millions of garbage validation records — random hashes, signed by valid keys — flooding the network. Each record is individually valid (correctly signed, properly formatted). The DAM fills with noise, relay and witness nodes waste bandwidth and storage, and legitimate records are drowned out.

This is the cost of "Layer 1 is always free." Free creation means free spam.

Defense 1: Stake-Scaled Propagation Rate Limiting

Layer 1 (local validation) is unrestricted — an attacker can fill their own local DAG with garbage. But Layer 2 (network propagation) applies rate limits per identity:

$$\text{effective\_limit} = \text{base\_rate} + \left\lfloor \frac{\text{staked\_base}}{\text{stake\_ratio} \times 24} \right\rfloor$$

Where `base_rate` is the unstaked floor (default: 120 records/hour, governance-adjustable) and `stake_ratio` is base units of stake (10^9 per beat) per daily record (default: 100,000,000 = 0.1 beats, governance-adjustable). The formula converts the daily stake allowance to an hourly propagation ceiling.

Stake	Hourly Limit	Records/sec	With 60× Batch
0 beats (unstaked)	120	0.03	2/sec
100 beats	~161	~0.04	~3/sec
1,000 beats	~536	~0.15	~9/sec
10,000 beats	~4,286	~1.19	~71/sec

Key properties:

- **Unstaked identities get the base floor.** Fresh or anonymous identities can still participate — 120 records/hour is sufficient for individual use. This preserves the "Layer 1 is always free" guarantee.
- **Stake scales linearly.** No tiered jumps. Staking 100 beats and staking 101 beats produce proportionally different limits. Industrial users get exactly the throughput they pay for.
- **Batching multiplies effective throughput.** A single record can contain multiple data points (Protocol §4.2 batch records). Combined with stake-scaling, a factory staking 1,000 beats with 60-reading batches achieves ~9 verified data points per second (~770,000 per day).
- **Gossip relay is exempt.** Records arriving via gossip pull or push relay from known peers are not rate-limited — only direct submissions from the record creator. This prevents sync failures between nodes with different propagation histories.
- **Both parameters are governance-adjustable.** Beat stakers can vote to change `base_rate` or `stake_ratio` without protocol upgrades, adapting to network growth or shifts in beat earning economics.

An attacker generating 1 million records per hour from an unstaked identity would see 120 propagate and the rest rejected. To sustain that rate, they would need to stake ~2.4 million beats — real economic commitment that can be slashed if the records are malicious.

Defense 2: Proof-of-Work for Burst Propagation

If a node needs to propagate more than its rate limit (legitimate use case: IoT gateway syncing a batch of sensor readings), it can solve a lightweight proof-of-work puzzle for each excess record. The puzzle difficulty is calibrated so that:

- Normal usage (under rate limit): zero computational cost
- Moderate burst (2–10x limit): seconds of compute
- Spam-scale burst (1000x+ limit): hours/days of compute — economically infeasible

This is the same approach used by Hashcash [24] (email anti-spam, 2002) and later adopted by blockchain networks. It adds no cost to honest users and makes spam expensive.

Defense 3: Content-Independent Duplicate Detection

Bloom filters at the zone level detect records with identical content hashes. If the same hash is submitted by different identities simultaneously (a classic spam pattern — resubmitting the same garbage with different keys), only the first propagation proceeds. Subsequent duplicates are annotated as conflicts but not relayed further.

Defense 4: Economic Filtering at Layer 2

Witness nodes choose what to attest. They are rational economic actors — attesting costs computational resources (PoWaS). No witness will spend resources attesting to records from identities with zero trust, anomalous patterns, or rate-limit violations. Spam records exist on the DAM but accumulate zero witnesses and zero trust. They are dead weight — present but invisible to anyone querying the DAM with a minimum trust threshold.

11.20 Nothing-at-Stake and Witness Incentive Alignment

The problem: In proof-of-stake systems, validators can sign conflicting blocks at no cost because there is "nothing at stake" — they don't burn energy like proof-of-work miners. Does the Elara Protocol's witness accumulation have the same problem? Can a witness attest to conflicting records (two creators claiming the same work) without penalty?

Analysis:

In blockchain PoS, nothing-at-stake matters because fork choice affects which chain becomes canonical. Attesting to both forks is profitable because you earn rewards on whichever one wins.

In the Elara Protocol, this dynamic does not apply because **both branches of a conflict are preserved.** There is no fork choice. Both records exist. Neither is "canonical." The trust score mechanism (Section 11.12) handles precedence through temporal and causal analysis, not by choosing a winner.

However, a related problem exists: **indiscriminate witnessing.** A witness could attest to everything — every record from every identity — to maximize rewards without performing any quality check. This degrades the trust signal.

Solution: Witness Reputation Scoring

Witnesses earn not just beats but **reputation** based on the quality of their attestations:

```
reputation_delta = f(record_outcome)

Record never disputed:          +1 reputation
Record disputed, witness sided with winner: +2 reputation
Record disputed, witness sided with loser:  -5 reputation
Record flagged as spam/anomaly:             -10 reputation
```

A witness that attests to everything — including spam and disputed records — rapidly loses reputation. Reputation loss reduces the weight of future attestations (Section 11.12), reducing earned rewards. Indiscriminate witnessing is therefore economically irrational.

A witness that carefully evaluates records before attesting — checking for rate limit compliance, duplicate content, identity trust, and causal consistency — maintains high reputation and earns more. Selective, honest witnessing is the Nash equilibrium. The witness reward mechanics are specified in Section 9.4.

The related problem of **reputation escape** — where an entity destroys a damaged identity to start fresh — is addressed in Section 11.33, which introduces identity continuity scoring and organizational binding as countermeasures.

11.21 Reserved

This subsection number is intentionally unused: an earlier attack-vector analysis was removed during revision, and the numbering was left stable so that existing citations to the later subsections (§11.22 onward) did not shift.

11.22 Cross-Zone Trust Reconciliation

The problem: Earth zone has 100,000 witnesses. Mars zone has 50. When zones merge after a partition, a record with 10,000 Earth witnesses and a record with 45 Mars witnesses have vastly different absolute trust scores — but within their zones, both may represent near-universal consensus. How do trust scores combine meaningfully?

Solution: Relative Trust Normalization

Trust scores are computed **relative to zone size**, not as absolute witness counts:

```
T_zone(r) = 1 - ∏(1 - w(n) × d(n, W_zone))  for all n in W(r) ∩ zone

T_global(r) = weighted_average(T_zone_i(r) for each zone i that has witnessed r)
              where weight_i = ln(zone_size_i + 1) / Σ ln(zone_size_j + 1)
```

Where d(n, W_zone) is the same correlation discount defined in Section 11.12, computed within the zone's witness set. This ensures that correlated witnesses within a zone do not inflate per-zone trust scores.

Interpretation:

- A record witnessed by 45 of 50 Mars nodes has T_mars = ~0.95 (very high within Mars)
- A record witnessed by 10,000 of 100,000 Earth nodes has T_earth = ~0.85
- Upon merge, T_global reflects both: ~0.87 (weighted by zone size logarithmically)

The logarithmic weighting prevents Earth's massive node count from completely dominating. Mars's consensus matters proportionally — a smaller zone with near-unanimous agreement is a strong signal.

Edge case — conflicting records across zones:

If Earth and Mars both validated records claiming the same content hash by different creators during a partition:

1. Both records are preserved (no deletion)
2. Each record carries its zone-relative trust score
3. The conflict is flagged with a **partition conflict** annotation
4. Resolution follows the dispute framework (Section 11.13): temporal priority via causal anchoring where possible, arbitration where not

Zone trust calibration:

Each zone publishes a **zone health metric** in its trust headers:

```
ZoneHealth {
    active_witnesses:    count
    total_staked:        beat amount
    median_reputation:   reputation score
    uptime_30d:          percentage
}
```

This allows global trust calculations to discount zones with low participation or unhealthy metrics, preventing a tiny zone with 3 colluding nodes from injecting high-trust records.

11.22.1 Cross-Zone Merkle Proof Protocol (v0.7.2) **The problem:** With zone-scoped storage, a node in Zone A does not hold Zone B's records. How does it verify a record from Zone B without trusting Zone B's nodes?

Solution: Self-Verifiable Records with Merkle Proofs

A record is self-verifiable when it carries: 1. The record itself (signed with Dilithium3/SPHINCS+) 2. A Merkle proof: the sibling hashes from the record's leaf to the zone's Merkle root 3. The epoch seal containing the Merkle root (signed by the zone's anchor, attested by witnesses) 4. The epoch seal chain: `previous_seal_hash` links back to genesis

Verification without holding any Zone B data:

```
1. Verify record signature (Dilithium3) → authentic
2. Compute Merkle path from record hash using proof hashes → get root
3. Compare computed root with epoch seal's merkle_root → matches
4. Verify epoch seal's anchor signature (Dilithium3) → authentic
5. Verify epoch seal attestations → ≥ 2/3 stake-weighted witnesses signed
6. Verify witness identities → staked, PoW, age requirements met
7. Verify epoch seal chain → previous_seal_hash chain to genesis_authority
```

Steps 1-3 are pure math (no network). Steps 4-7 require the epoch seal and witness identity data, which can be cached or requested via DHT.

Cross-zone proof request protocol:

```
1. Node A wants to verify record R from Zone B
2. DHT lookup: sha256("elara-zone:" || zone_B_path) → find Zone B peers
3. Request from Zone B peer: GET /zone/B/proof/{record_hash}
4. Response: { record, merkle_proof, epoch_seal, witness_attestations }
5. Node A verifies everything locally (steps 1-7 above)
6. Cache the epoch seal for future Zone B verifications
```

Proof size: For a zone with N records, Merkle proof is log2(N) × 32 bytes. For 1 billion records: 30 × 32 = 960 bytes. Negligible bandwidth.

Cross-zone transfers use this mechanism. A transfer-lock record in Zone A is presented to Zone B with its Merkle proof. Zone B verifies without contacting Zone A.

11.22.2 Cross-Zone Transfer Protocol (v0.7.3) Cross-zone beat transfers use a two-phase lock/claim mechanism that preserves the conservation invariant across zone boundaries:

Phase 1 — Lock (sender zone):

The sender creates a `TRANSFER_LOCK` record in their zone specifying: recipient identity, amount, destination zone path, and a 24-hour timeout. The sender's balance is debited and the amount moves to `pending_outbound` state. The lock record is included in the next epoch seal and becomes Merkle-provable.

Phase 2 — Claim (recipient zone):

The recipient (or any node in the destination zone) presents the lock record with its Merkle proof (Section 11.22.1) to the destination zone. The destination zone verifies the proof, confirms the lock has not been claimed or refunded, and credits the recipient's balance. A `TRANSFER_CLAIM` record is created in the destination zone.

Timeout and refund:

If 24 hours elapse without a valid claim, the lock expires. The sender (or any node in the sender's zone) can create a `TRANSFER_REFUND` record that returns the locked amount to the sender's `available` balance. The timeout is enforced by epoch number: if `current_epoch - lock_epoch > timeout_epochs`, the refund is valid.

Conservation invariant:

At all times: `sum(zone_balance_totals) + pending_cross_zone + conservation_pool = GENESIS_SUPPLY`. Each zone's `zone_balance_total` in its epoch seal tracks its portion. The `pending_cross_zone` amount is the sum of all outstanding (locked but unclaimed/unrefunded) cross-zone transfers. This is auditable from epoch seals alone.

11.23 Search, Indexing, and DAM Usability

The gap: The paper specifies how records get onto the DAM but not how anyone finds them. With billions of records, the DAM is useless without search. How does a creator find their own records? How does a verifier find a specific claim? How does a researcher discover related work?

Solution: Multi-Layer Index Architecture

Layer A: Content Hash Lookup (exact match)

The simplest query: "Does this exact content exist on the DAM?"

Every node maintains a local hash index (key: content_hash → value: record_id). The Kademlia DHT (Section 11.14) enables global lookup: any node can find any record by its content hash in O(log n) hops.

This is how verification works: hash your content, look it up, see if someone already claimed it.

Layer B: Creator Index (identity lookup)

"Show me everything validated by this identity."

Each node maintains an index of its own records. Anchor nodes maintain comprehensive per-creator indexes for the identities they've witnessed. Light clients query anchor nodes for creator-specific records.

Layer C: Semantic Search (Layer 3 — AI)

"Find all validated work related to quantum computing from 2026."

The AI intelligence layer (Layer 3) builds semantic indexes over validation record metadata. This includes:

- Full-text search over PUBLIC metadata fields
- Category/tag-based filtering
- Temporal range queries
- Similarity search (finding related work via content fingerprints)

Semantic search is a Layer 3 capability — optional, beat-gated, and computationally expensive. It is not required for basic DAM operation but enables rich discovery for nodes that participate in the AI layer.

Layer D: Zone Catalogs

Each zone publishes a periodic **zone catalog** — a compact index of:

- Recently validated record IDs and metadata
- Active creators in the zone
- Popular content categories
- Cross-references to related records in other zones

Zone catalogs enable browsing and discovery without full-text search. They are small enough for light clients to download and cache.

User Interface Implication:

The protocol specification defines the data layer, not the UI. But reference implementations will include:

- **CLI tool:** elara validate, elara search, elara verify
- **Mobile app:** camera-based validation (photograph → hash → validate), gallery of own validated work
- **Web interface:** search, browse, verify — hosted by anchor nodes or community

End users see a simple app: create, validate, browse. The underlying DAM complexity is invisible.

11.24 Energy Consumption Analysis

The obligation: The paper discusses the energy costs of existing consensus mechanisms. Credibility requires disclosing the Elara Protocol's own energy footprint.

Estimation by component:

Layer 1 (local validation): Negligible. Hashing + signing = milliseconds of CPU time. Energy cost: ~183 µJ (~50.8 nWh) per validation on a smartphone (see Hardware Whitepaper v0.2.0, Section 10.2 for detailed breakdown). Orders of magnitude less than sending a text message.

Layer 2 (network propagation + witnessing):

- **Gossip propagation:** Network traffic comparable to a peer-to-peer file-sharing swarm. Each node sends/receives a compact record announcement (~1 KB header + hash, not the full ~4.5 KB record) to √n peers per hop; full records are fetched on demand by interested nodes. Records propagate within their zone (Section 7), not globally — zone partitioning bounds the effective fanout. For a zone with ~10,000 active nodes, epidemic gossip converges in ~4 rounds of √n fan-out. Most records are created by leaf nodes and propagated to their zone's witness set, not broadcast to the entire network. The energy estimate below uses network-wide averages accounting for zone partitioning and leaf-node locality.
- **Witness attestation (PoWaS):** The proof-of-work component is calibrated to be lightweight — ~0.1 seconds of CPU per attestation (vs. ~10 minutes of ASIC-scale

computation in proof-of-work blockchains). Energy per attestation: ~0.005 Wh. For 1,000 witnesses per record: ~5 Wh total.

- **Anchor node operation:** Comparable to running a modest server. ~100W continuous. Per anchor node per year: ~876 kWh. For 100 anchor nodes globally: ~87,600 kWh/year.

Layer 3 (AI-assisted analysis): Dependent on the model and hardware. Ollama running a 1.5B parameter model: ~50W during inference. Intermittent, not continuous. Per node per year (assuming 4 hours/day active): ~73 kWh.

Total network estimate at scale (1 million nodes, 100 anchor nodes):

Component	Annual Energy
Layer 1 (local validation)	< 1 MWh (negligible)
Layer 2 (propagation)	~2,000 MWh
Layer 2 (witnessing PoWaS)	~5,000 MWh
Anchor nodes	~88 MWh
Layer 3 AI (optional, ~10% of nodes)	~7,300 MWh
Total	**~14,400 MWh/year**

Comparison:

Network Type	Annual Energy
Proof-of-work blockchains	~150,000,000 MWh (150 TWh)
Proof-of-stake platforms	~1,500–2,600 MWh
Elara Protocol (estimated)	**~14,400 MWh**

The Elara Protocol at full scale would consume roughly **10,000x less energy than proof-of-work blockchains**. It consumes approximately 6x more than proof-of-stake platforms in absolute terms, but the comparison is not apples-to-apples: proof-of-stake networks process financial transactions (~30 TPS), while the Elara Protocol validates all forms of digital work at IoT scale (millions of records per day) with quantum-safe cryptography. Layer 1 validation energy is negligible (~183 µJ per operation) — the dominant costs are network propagation, witness consensus, and optional AI analysis.

The PoWaS component is the largest contributor. If the Sybil resistance proves achievable without PoW (through reputation and staking alone), the energy footprint drops to ~9,400 MWh. Without optional Layer 3 AI, it drops further to ~2,100 MWh — comparable to proof-of-stake networks.

11.25 Harmful Content and Ethical Boundaries

The dilemma: The protocol validates all digital work. "All" includes content that society broadly agrees should not be preserved: child exploitation material, terrorist recruitment, bioweapon designs, doxxing databases. If the DAM is immutable and censorship-resistant, does it become a permanent haven for the worst of human output?

This is not a hypothetical. Every decentralized system faces this challenge — immutable ledgers and content-addressed storage networks have encountered cases of embedded or distributed harmful material. The Elara Protocol must have an answer.

Principle: The DAM stores hashes, not content.

A validation record contains a cryptographic hash — not the content itself. The hash of an illegal image is not an illegal image. It is a 64-character string of hexadecimal digits. It cannot

be “viewed” or “consumed.” The harmful content exists wherever the creator stored it (their device, a file host), not on the DAM.

If the content is removed from all storage, the hash becomes an orphan pointing to nothing. The hash remains, the harm does not.

However: Validation records also carry metadata — operation identifiers, transfer memos, governance proposals, dispute descriptions. These are user-writable text fields. Without protocol-level constraints, metadata becomes a vector for embedding harmful text, links to illegal content, or communication channels for criminal coordination. Every node that stores and gossips a record becomes an unwitting host.

The Silk Road precedent established that platform operators bear criminal liability when their systems enable illegal content distribution — even when operators did not create the content. Node operators who propagate records containing drug transaction details, child exploitation references, or terrorist recruitment text face the same exposure. “The protocol doesn’t judge the payload” is not a legal defense when the payload is criminal communication carried in cleartext metadata fields on every node in the network.

The Elara Protocol therefore implements **structural hostility to content distribution** — not content filtering (which is an arms race), but architectural constraints that make the protocol unsuitable for hosting, distributing, or linking to content of any kind.

11.25.1 Defense Architecture Overview The protocol implements eight independent defense layers, applied at the record ingestion gate (`insert_record_with_origin()`). All layers enforce on ALL entry paths — HTTP POST, WebSocket submission, gossip propagation, and internal record creation (faucet, rewards, epoch seals). No code path bypasses the gate.

Layer	Type	Action	Applied At
1. Metadata key discipline	Structural	Reject blocked/malformed keys; admit unknown keys as inert	Ingestion
2. Text field byte limits	Structural	Reject oversized text	Ingestion
3. URL rejection	Structural	Reject URLs in text	Ingestion
4. Tombstone suppression	Reactive	Suppress gossiped records	Post-storage
5. Browser-side enforcement	Defense-in-depth	Client-side validation	Pre-submit
6. Propagation bounds	Structural	64 entries, 8 KB values	Ingestion
7. Identity ban list	Proactive	Block all records from identity	Ingestion (first check)
8. Content blocklist	Proactive	Block records matching terms	Ingestion

Layers 1–3 and 6 are **structural constraints** — they limit the protocol’s capacity to carry content, like TCP’s maximum segment size limits the payload of a single packet. They are engineering constraints, not content judgments.

Layers 4, 7, and 8 are **operator tools** — they give node operators the means to comply with their jurisdiction’s laws and protect themselves from criminal liability. They are not centralized censorship.

Layer 5 is **defense-in-depth** — client-side validation that prevents user confusion but is not a security boundary. Server-side enforcement is authoritative.

11.25.2 Metadata Key Discipline (Layer 1) The runtime maintains an allowlist covering the complete set of keys used by all protocol operations: beat transfers, staking, governance, disputes, fisherman challenges, key rotation, algorithm sunset, epoch management, versioning, tombstoning, and collaboration. The allowlist is a **producer-side schema registry**: continuous-integration meta-tests enforce that every in-tree record builder registers its keys in the same change that introduces them.

At ingestion, key discipline is enforced structurally. Key names are frozen to a permanent character set — lowercase `[a-z0-9_]`, at most 128 bytes — and records violating it are rejected. Keys on the blocked list (below) are rejected outright. Unknown keys that satisfy the structural rules are **admitted and treated as inert**: the record is stored and gossiped, its unknown values subject to every size, count, sanitization, and URL bound in this section, but no execution path reads them — protocol logic dispatches only on keys it explicitly recognizes, never by iterating a record's metadata. An operator counter tracks admitted unknown keys, so a node running behind the current schema is visible in monitoring long before it matters.

This forward-compatible admission rule is what keeps mixed-version fleets synchronized. A strict reject-unknown gate would couple every deployed binary to the newest schema: the first additive metadata key would make older nodes reject every newer record — epoch seals included — freezing them at their last synchronized epoch while the network advances. Admission-with-inertness lets an older binary carry newer fields untouched and stay in consensus. The compatibility boundary is explicit and narrow: any key a node must *interpret* to remain in consensus (seal semantics, quorum logic, state transitions) is introduced behind a wire-format version bump with a documented migration, never as a silently added field.

Six anti-rehypothecation keys are explicitly blocked — beats cannot be wrapped, tokenized, or collateralized into a tradeable instrument: `derivative_op`, `wrap_op`, `collateral_op`, `tokenize_op`, `synthetic_op`, `lend_op`. Records containing these keys are rejected at ingestion, and changing the blocked list is itself a consensus-affecting change carrying the same version discipline.

11.25.3 Text Field Byte Limits (Layer 2) User-writable text fields have per-field byte limits:

Field	Max bytes	Purpose
`beat_memo`	256	Transfer/burn memo
`beat_reason`	256	Mint/slash/reclaim reason
`governance_title`	128	Proposal title
`governance_description`	1024	Proposal description
`dispute_reason`	512	Dispute opening reason
`dispute_evidence`	1024	Evidence submission
`challenge_evidence[]`	512/entry	Fisherman evidence (array)
`challenge_appeal_reason`	512	Appeal text
`revocation_reason`	128	Key revocation reason
`rotation_reason`	128	Key rotation reason
`change_summary`	512	Version change description
`sunset_reason`	256	Algorithm sunset reason

These limits make the network unsuitable for embedding images (even base64-encoded), documents, or bulk content. A 256-byte memo field holds approximately 200 characters of text — enough for a transaction description, insufficient for meaningful content distribution.

11.25.4 URL Rejection (Layer 3) All text fields reject URLs matching these patterns (case-insensitive):

- `http://`, `https://` — web links to external content
- `ftp://` — file transfer links
- `data:` — base64-encoded inline content (images, documents)
- `javascript:` — cross-site scripting vectors
- `magnet:` — torrent/peer-to-peer content links

This prevents records from serving as a link directory for illegal content. A record cannot contain a URL pointing to child exploitation material, a drug marketplace, or a terrorism recruitment site. The protocol is structurally unable to function as a content indexing or referral system.

11.25.5 Propagation Bounds (Layer 6) Hard limits at the gossip layer constrain record size:

- **Maximum metadata entries:** 64 per record — sized so a fully-populated epoch seal (~26 keys today) carries years of additive headroom, while staying under the wire decoder's independent 256-entry bound
- **Maximum value size:** 8 KB per metadata value (sized so hex-encoded Dilithium3 public keys and VRF proofs fit)
- **Maximum record size:** 64 KB total wire size

The binding cap is the 64 KB total record size — the per-field limits intentionally sum above it, so the aggregate bound is what holds. In practice, most records are 200–500 bytes (a few metadata keys plus a signature). These bounds prevent weaponization of the metadata layer as a distributed storage system.

11.25.6 Tombstone Mechanism (Layer 4) The genesis authority can suppress records from future propagation by creating a tombstone record:

- Tombstone records contain `tombstone_op: "remove"` and `tombstone_target: "<record_id>"`
- Only the genesis authority identity can create tombstones
- Tombstoned records remain in storage (immutability is preserved) but are excluded from gossip responses and API queries
- Tombstone records themselves propagate normally, so all nodes learn about suppression decisions

Limitation — race condition: If a target record propagates to a node before the tombstone arrives, that node will have already stored and indexed the record. The tombstone prevents future propagation but does not un-index already-processed records. Identity bans (Layer 7) are the primary proactive defense; tombstoning is reactive cleanup.

Immutability guarantee: Tombstoning does NOT delete records. The record remains in storage as an audit trail. Tombstoning suppresses propagation — it controls what the network carries forward, not what it has already stored. This distinction preserves the immutability guarantee of Section 11.5.

11.25.7 Identity Ban List (Layer 7) The genesis authority can permanently ban identity hashes. Banned identities have ALL new records rejected at the ingestion gate — before signature verification, before storage, before gossip. Records from banned identities never touch the DAG.

- Persisted in storage (survives node restarts)
- Loaded into an in-memory set on startup for O(1) lookup
- Checked FIRST in the ingestion pipeline — the earliest possible rejection point

- Genesis authority only — prevents abuse of the ban mechanism

Identity banning is the most effective proactive defense. If an identity is banned before its records arrive at a node, those records are rejected without consuming any computational resources (no signature verification, no storage I/O, no gossip bandwidth).

11.25.8 Operator-Configurable Content Blocklist (Layer 8) Node operators can configure a term blocklist applied to all user-writable text fields. Records containing blocked terms are rejected at ingestion — never stored, never gossiped.

- Case-insensitive substring matching
- Minimum term length: 2 characters (prevents false positives from single-character matches)
- Persisted in storage, hot-reloadable via admin API
- Genesis authority only (on managed nodes)

Critical design decisions:

1. **No default terms are shipped with the protocol.** The blocklist is empty at genesis. Operators add terms relevant to their jurisdiction and legal obligations. This prevents the protocol from encoding a single cultural standard or becoming a vehicle for centralized censorship.
2. **Operators control their own nodes.** Each node's blocklist is independent. Node A may block terms that Node B does not. This mirrors how different jurisdictions have different content laws — a node operated in Germany applies German law, a node in Singapore applies Singaporean law.
3. **The blocklist operates on metadata text fields, not on content hashes.** The protocol does not implement a forbidden-hash blacklist. Hash-based blacklists are content-agnostic censorship mechanisms that can suppress any record regardless of its content. Term-based filtering operates on the small set of user-writable text fields (memos, descriptions, reasons) and is transparent — operators and users can see exactly what terms are filtered.

11.25.9 Content-Layer Enforcement Protocol-layer defenses (Sections 11.25.1–11.25.8) make the network structurally hostile to content distribution. Content-layer enforcement remains the primary mechanism for addressing the content itself:

- File hosts can remove content (as they already do under DMCA, EU DSA, etc.)
- ISPs can block access to content hosts
- App stores can require content moderation in Elara client apps
- Law enforcement can compel content removal from devices

The DAM hash remains — proving the content existed — which is actually useful for law enforcement (immutable evidence).

11.25.10 Identity Accountability Every validation record is signed by an identity. If an identity is linked to criminal content:

- Law enforcement can subpoena anchor nodes for the full validation history of that identity
- The identity's entire creation chain is exposed — not just one record, but everything they ever validated
- Court-ordered identity revocation can render the identity's records unattributable (Section 11.5, GDPR mechanism)
- The identity ban mechanism (Section 11.25.7) provides immediate technical enforcement of legal orders

Pseudonymity is not anonymity. Persistent identity means persistent accountability.

11.25.11 Zone-Level Content Policy Individual zones can adopt content policies without affecting the global protocol:

- A zone serving a jurisdiction with strict content laws can refuse to witness records from flagged identities
- This does not delete the record from the global DAM — other zones still process it
- This is analogous to how different countries have different internet regulations today

11.25.12 Governance of Content Safety Tools During the bootstrap phase, content safety tools (tombstoning, identity bans, content blocklist) are administered by the genesis authority. This is a pragmatic centralization — a small network needs rapid response capability against abuse, and distributed governance mechanisms require a critical mass of staked participants.

The planned governance evolution:

1. **Bootstrap phase (current):** Genesis authority administers all content safety tools
2. **Growth phase:** Multi-signature requirement for tombstoning and identity bans (e.g., 3-of-5 anchor node operators)
3. **Maturity phase:** Governance proposals (Section 10) for content policy changes, with conviction voting by staked participants

The protocol explicitly commits to decentralizing content moderation governance as the network matures. Centralized control during bootstrap is a necessary concession, not a permanent architecture.

11.25.13 What the Protocol Will NOT Do

- It will not implement a centralized, default-shipped blacklist of forbidden content hashes. Hash-based blacklists are content-agnostic censorship mechanisms that will inevitably be abused. Operator-configured term filters on metadata text fields are a different mechanism — transparent, jurisdiction-specific, and under operator control.
- It will not allow retroactive deletion of validation records. Immutability is a core guarantee. Tombstoning suppresses propagation; it does not delete storage. Records that have been stored remain stored.
- It will not ship default content blocklist terms. The protocol does not encode a single cultural standard. Operators configure their own nodes according to their own legal obligations.

11.25.14 Honest Position Earlier versions of this document stated: "A protocol that can censor harmful content can censor any content." This remains true. The eight defense layers described above give node operators the technical capability to reject records — and that capability can be misused.

The protocol's defense against misuse is structural:

- **Structural constraints (Layers 1–3, 6) cannot be weaponized.** A 256-byte memo limit does not censor political speech. A metadata key allowlist does not suppress dissent. URL rejection does not block ideas. These are engineering constraints that limit the protocol's capacity to carry content — any content, harmful or benign.
- **Operator tools (Layers 4, 7, 8) are decentralized.** Each node operator controls their own tombstones, bans, and blocklist. There is no central authority that can order all nodes to suppress a record. A record rejected by one node may be accepted by another. The global DAM is the union of all nodes' records — suppression on one node does not suppress on the network.

- **No default terms are shipped.** The protocol does not decide what is harmful. Operators do, according to their jurisdiction.

The infrastructure does judge certain properties of the payload — its size, its key structure, the presence of URL patterns in text fields. It does not judge the meaning. This is the same architectural boundary observed by SMTP (which rejects messages over a size limit but does not read them) and DNS (which enforces label length limits but does not evaluate domain semantics).

The Elara Protocol chooses structural hostility to content distribution over either extreme: it does not preserve everything blindly (which creates criminal liability for operators), nor does it build meaning-level content moderation into the protocol layer (which creates censorship infrastructure). The structural approach makes the network unsuitable for content distribution without making it capable of content-level censorship.

11.26 Quantum Vulnerability of the ZKP Layer

The gap: The protocol's signatures are post-quantum (Dilithium, SPHINCS+). The zero-knowledge proofs *specified* for PRIVATE and RESTRICTED classifications (Section 5.3) rely on zk-SNARKs — which typically use elliptic curve pairings (BN254, BLS12-381). These pairings are **not quantum-safe.** Shor's algorithm breaks them. (Phase 1 as shipped does not yet carry this exposure: it uses SHA3-256 hiding commitments, which are hash-based. The gap becomes live when the specified zk-SNARK layer lands.)

This means: once zk-SNARKs land, a quantum adversary could forge zero-knowledge proofs, creating fake PRIVATE validations that appear genuine. The signature is quantum-safe, but the proof is not. This is a real gap in the specified design.

Solution: Quantum-Safe ZKP Migration Path

Phase 1 (current): SHA3-256 commitments with PQC signatures; classical zk-SNARKs specified

Phase 1 as shipped uses SHA3-256 hiding commitments — hash-based, no classical-curve exposure. The *specified* Groth16 zk-SNARKs (~288-byte proofs, fast verification) are design-stage and wire-rejected until the prover/verifier lands (Section 5.3). Either way, the surrounding validation record is signed with Dilithium (PQC): an attacker who breaks the proof layer must also forge the PQC signature to create a fraudulent record — which they cannot do.

The risk window: an attacker with a quantum computer could forge a ZKP proof but would need to wrap it in a valid Dilithium signature (their own key). This allows them to falsely claim PRIVATE validation of content they created — but they could already do this by simply validating with a PUBLIC classification. The ZKP breach lets them fake the privacy wrapper, not the validation itself.

Assessment: The practical impact of ZKP quantum vulnerability is limited because: - Breaking the ZKP does not let you forge someone else's validation (the PQC signature still binds it to a specific key) - It lets you create fake PRIVATE records under your own key — which has limited value since you could create PUBLIC records instead - The real attack would be de-anonymizing existing PRIVATE records by breaking the hiding property of the commitment scheme

Phase 2 (2027-2029): Hybrid ZKPs

Deploy hybrid proofs that combine a classical zk-SNARK with a lattice-based ZKP:

- **Classical component:** Groth16 (compact, fast, familiar tooling)
- **PQC component:** Lattice-based ZKP (larger proof, but quantum-safe)

Both proofs must verify for the record to be considered valid. If quantum computing breaks the classical component, the lattice-based component still holds.

Proof size increases from ~288 bytes to ~50 KB. Acceptable for non-constrained devices.

Phase 3 (2029+): Full PQC ZKPs

As lattice-based and hash-based ZKP constructions mature:

- Migrate PRIVATE/RESTRICTED to fully post-quantum ZKPs
- Candidates: lattice-based SNARKs, STARK-based systems (specified for SOVEREIGN), hash-based commitment schemes (the basis of the Phase 1 implementation)
- Algorithm agility (Section 4.4) enables seamless transition

For SOVEREIGN classification: Quantum-safe by design. SOVEREIGN is specified to use zk-STARKs, which are based on hash functions (not elliptic curves) and are not vulnerable to Shor's algorithm. (Specified, not yet implemented — see Section 5.3.)

11.27 Multi-Device Key Management

The problem: A user has a phone, a laptop, and a tablet. They generate a keypair on their phone. How do they validate work from their laptop? If the phone breaks, how do they access their validated history from a new device?

This is the most common UX failure in cryptographic systems. PGP failed mass adoption largely because of key management complexity. The Elara Protocol must not repeat this.

Solution: Identity Constellation Model

Instead of one keypair per person, the protocol supports an **identity constellation** — a set of device keys linked to a root identity:

```
RootIdentity (generated once, stored securely)
├── DeviceKey: phone_key (signed by root)
├── DeviceKey: laptop_key (signed by root)
├── DeviceKey: tablet_key (signed by root)
└── RecoveryKey: paper_key (stored offline, Section 11.2)
```

How it works:

1. **Root key generation:** User generates a root keypair. This is the canonical identity. The root private key is immediately exported to a secure backup (paper, USB, password manager) and optionally deleted from the generating device.
2. **Device key enrollment:** Each device generates its own keypair and submits an enrollment request. The root key signs a **DeviceAuthorization** record:

```
DeviceAuthorization {
    root_identity:  root public key
    device_key:     device public key
    device_name:    "Alice's phone" (optional, user-facing)
    permissions:    [VALIDATE, WITNESS, REVOKE_DEVICE]
    expires:        optional expiration date
    signature:      root key signature
}
```

3. **Validation from any device:** When the user validates work from their laptop, the validation record is signed by laptop_key. Any verifier can follow the DeviceAuthorization chain: laptop_key → authorized by root_identity → valid.
4. **Device loss/theft:** The user revokes the lost device's key using any other enrolled device or the recovery key. A DeviceRevocation record propagates across the network. The lost device's future signatures are rejected.

5. **All devices lost:** The recovery key (paper, USB) can revoke all device keys and enroll new devices. This is the catastrophic recovery path — inconvenient but functional.

Key sync between devices:

Device keys are independent — they do not share private key material. There is no "syncing" of private keys between devices (which would be a security risk). Instead, each device has its own key, and the root identity links them.

The user's validated work history is accessible from any device by querying the DAM for records associated with the root identity or any of its authorized device keys.

UX simplification:

Reference implementations abstract this complexity:

- First launch: "Create your Elara identity" (generates root key + first device key)
- "Back up your recovery phrase" (12-word mnemonic encoding the root key, BIP-39 [25] compatible)
- Adding a device: scan QR code on existing device → DeviceAuthorization created automatically
- Losing a device: "Remove device" from any other enrolled device

The user never sees keypairs, hashes, or DAG structures. They see devices and a recovery phrase.

For organizational deployments (IoT fleets, enterprise environments), the Identity Constellation model extends to **organizational identity chains** with fleet-level binding — see Section 11.33 for the full specification.

11.28 Eclipse Attacks

The attack: An adversary surrounds a target node with malicious peers, controlling all its network connections. The target sees only the attacker's version of the DAM — a curated subset that excludes certain records or includes fabricated ones. Unlike Sybil attacks (fake identities), eclipse attacks manipulate network topology.

Impact: An eclipsed node might: - Not receive revocation records (believing a compromised key is still valid) - Not see conflicting claims (believing it has priority when it does not) - Receive fabricated trust headers (believing records have more witnesses than they do)

Defense 1: Diverse Peer Selection

The Kademlia DHT (Section 11.14) provides natural eclipse resistance because peer selection is based on XOR distance in the key space, not network topology. An attacker would need to generate keys close to the target's key in Kademlia space — computationally expensive and detectable (anomalous key clustering).

Additional diversity enforcement: - Each node maintains connections to peers in at least 3 different /16 IP subnets - Each node maintains connections to peers in at least 2 different geographic regions (determined by IP geolocation) - Outbound connections (initiated by the target) are prioritized over inbound (initiated by potential attackers)

Defense 2: Anchor Node Pinning

Every node maintains at least one persistent connection to a known anchor node. Anchor nodes are operated by identified, staked entities — eclipsing them requires compromising the anchor operator, not just manipulating network routing.

If all peer connections seem to agree but the anchor node disagrees, the node raises an **eclipse alert** — flagging a potential attack and refusing to accept trust headers that conflict with the

anchor's view.

Defense 3: Trust Header Cross-Validation

Trust headers (Section 11.3) are signed by multiple anchor nodes. A legitimate trust header carries signatures from geographically distributed anchors. An eclipsed node that receives trust headers signed by only one anchor (or by unknown entities) detects the discrepancy and falls back to a trust-no-one mode — accepting only locally validated records until the eclipse is resolved.

Defense 4: Out-of-Band Verification

For high-stakes verification (large transactions, legal proceedings), the protocol supports out-of-band verification: the verifier queries a known anchor node directly (by domain name or IP, not through the DHT) to confirm a record's status. This bypasses any eclipse on the local network.

11.29 Long-Range Attacks with Historical Keys

The attack: An adversary obtains old private keys — from a decommissioned device, a leaked backup, a compromised archive, or (eventually) quantum cryptanalysis of old algorithms. They use these keys to create validation records that appear to originate from the legitimate key holder at a past date.

This is more subtle than simple key compromise (Section 11.2): the attacker is not impersonating the current identity, but fabricating historical records using keys that were legitimately valid at some point.

Defense 1: Epoch Sealing

Epoch summaries (Section 11.8) create cryptographic snapshots of the DAM at regular intervals. Each epoch summary includes a Merkle root of ALL records in the epoch, signed by multiple anchor nodes. **Clarification on epoch sealing authority:** Anchor nodes are the subset of witness nodes designated as epoch sealers for their zone. The sealing mechanism is witness-signed Merkle roots — the same witnesses that attest records also seal epochs. There is no separate "zone authority" or "epoch authority" — anchor nodes ARE witnesses with additional sealing responsibility.

A fabricated historical record would not be included in any existing epoch summary. If an attacker produces a record claiming to be from epoch 42, but epoch 42's sealed Merkle root does not include it — the forgery is detected instantly.

Implication: The window for historical forgery is limited to the current unsealed epoch (typically hours to days). Once an epoch is sealed, its contents are cryptographically frozen.

Defense 2: Key Epoch Binding

The protocol records the epoch in which each identity key was first seen. A key first observed in epoch 100 cannot produce records claiming to be from epoch 50. Key-epoch bindings are established when a node's identity registration record is first witnessed by anchor nodes and included in an epoch summary. Since epoch summaries are signed by multiple geographically distributed anchors and sealed with Merkle roots, retroactively altering a key's first-seen epoch would require compromising the sealed epoch — which is equivalent to breaking the Merkle chain.

For keys that pre-date the network (bootstrapping phase), the genesis epoch explicitly lists all founding keys — preventing backdated claims from before the network existed.

Defense 3: Algorithm Sunset Enforcement

When a cryptographic algorithm transitions from ACTIVE to DEPRECATED (Section 11.18), a **sunset record** is created:

```
AlgorithmSunset {
    algorithm:      "dilithium3"
    status:         DEPRECATED
    effective_epoch: 10000
    reason:         "Lattice cryptanalysis advance, see CVE-2035-XXXX"
    signed_by:      [multiple anchor nodes]
}
```

After the sunset epoch, nodes enforce rejection as follows: when a new record arrives, the node checks its signature algorithm against the active sunset records. If the algorithm's status is DEPRECATED and the record's epoch exceeds the effective_epoch, the record is dropped during gossip propagation and excluded from the local DAG. Witness nodes will not attest to records with deprecated signatures. This enforcement is local — each node independently applies the sunset rules from the DAM's sunset records, requiring no central coordinator.

Old records (pre-sunset) remain verifiable for historical purposes but cannot be used as the basis for new claims. Nodes maintain a legacy verification path for deprecated algorithms to validate historical records while rejecting new ones.

11.30 Content Versioning Protocol

The gap: A document goes through 20 drafts. A software project has thousands of commits. An AI model has hundreds of training iterations. The paper describes validating individual artifacts but not the relationship between versions of the same work.

Solution: DAM-Native Version Chains

The protocol supports explicit version linking through a **VersionRecord**:

```
VersionRecord {
    content_hash:     hash of current version
    previous_version: record_id of the prior version (or null for v1)
    version_number:   sequential counter
    change_summary:   optional metadata describing what changed
   creator:       must match previous version's creator (or authorized collaborator)
    signature:        creator's signature
}
```

Properties:

- **Chain integrity:** Each version references the previous version's record ID. The full history is traversable from any version back to v1. Tampering with any version breaks the chain.
- **Fork tracking:** If two people create different v3s from the same v2 (a fork), both are preserved on the DAM as branches — exactly like a git repository, but cryptographically signed and globally attested.
- **Diff validation:** For text-based content, the protocol supports optional **diff records** — validated diffs that prove the exact changes between versions. A verifier can reconstruct any version from v1 + the chain of diffs, confirming nothing was altered retroactively.
- **Semantic versioning:** The metadata field supports semver (1.0.0, 1.1.0, 2.0.0) for software, draft numbering for documents, or any user-defined scheme.

Integration with incremental validation (Section 11.7):

Version chains ARE the incremental validation recommended for originality protection. A poet who validates each draft creates a version chain automatically. The chain proves creative process — something a plagiarist cannot replicate.

11.31 Formal Verification Strategy

The obligation: A protocol handling trust for all digital creation across planetary distances must be provably correct. Informal reasoning and test suites are necessary but insufficient. Formal verification provides mathematical proof that the protocol's properties hold under all conditions.

Approach: Layered Verification

Layer 1: TLA+ Specification of Consensus

The Adaptive Witness Consensus mechanism (Section 11.12) is specified in TLA+ (Temporal Logic of Actions) — the same framework used to verify distributed databases, cache coherence protocols, and globally distributed data services at major technology companies — and its safety and liveness cores are bounded-model-checked with TLC (the runnable models and configs ship in `spec/tla/`).

The TLA+ specification will formally verify: - **Safety:** If an honest node considers a record zone-settled, no honest node will ever consider a conflicting record zone-settled in the same zone - **Liveness:** Every record propagated to an honest node will eventually be witnessed (assuming network connectivity) - **Partition correctness:** Zone merging preserves all records from both partitions without duplication or loss

Layer 2: Cryptographic Protocol Verification (ProVerif/Tamarin)

The cryptographic handshakes (key exchange, device authorization, revocation) will be verified using ProVerif or Tamarin Prover — automated tools for verifying security protocols. These tools can prove:

- No man-in-the-middle attack is possible on device enrollment
- Revocation propagation guarantees eventual consistency
- The ZKP circuit correctly hides private inputs

Layer 3: Reference Implementation Testing

Beyond formal verification, the reference implementation will include:

- **Property-based testing** (QuickCheck/Hypothesis) — generating millions of random scenarios and verifying invariants hold
- **Fuzzing** (AFL, libFuzzer) — feeding malformed inputs to every parser and protocol handler
- **Simulation testing** — running 10,000-node simulations with adversarial behavior, partitions, and clock skew
- **Chaos engineering** — randomly killing nodes, introducing latency, corrupting storage, and verifying recovery

Timeline:

- Phase 1 (2026-2027): TLA+ specification and model checking alongside reference implementation development
- Phase 2 (2027-2028): Cryptographic protocol verification before mainnet launch
- Phase 3 (ongoing): Continuous fuzzing and simulation testing as part of CI/CD

Publication: All specifications, proofs, and test results will be published as companion documents to this whitepaper, enabling independent verification by the academic community.

11.32 Storage and Bandwidth Requirements

The question: How much data does the Elara Protocol produce, and what are the storage requirements for different node types?

This section provides concrete estimates based on the cryptographic primitives specified in this paper, enabling infrastructure planning and economic modeling.

Per-Record Sizes

Record Type	Approximate Size	Breakdown
PUBLIC validation record	~4-5 KB	Content hash (32 B) + Dilithium3 signature (~3.3 KB) + metadata/causal anchors (~500 B)
PRIVATE validation (Phase 1 = SHA3 commitment)	~3.8 KB	commitment (32 B) + PQC signature (~3.3 KB) + metadata (~500 B); + ~288 B when the specified Groth16 proof ships
PRIVATE validation (Phase 2, hybrid ZKP)	~55 KB	Hybrid lattice+classical proof (~50 KB) + PQC signature (~3.3 KB) + metadata (~500 B)
SOVEREIGN validation (zk-STARK, specified)	~100-200 KB	STARK proof (variable, typically 50-200 KB) + PQC signature (~3.3 KB) + metadata
Witness attestation	~3.5 KB	Record reference (32 B) + Dilithium3 signature (~3.3 KB) + timestamp + node identity
Trust header	~15-20 KB	Epoch reference + multiple anchor node signatures + zone metadata
Epoch summary	~50-100 KB	Merkle root + multi-anchor signatures + record count + zone state
DeviceAuthorization record	~5 KB	Root identity + device key + permissions + PQC signature
VersionRecord	~4-5 KB	Previous version reference + content hash + PQC signature + metadata

Each validation record requires 3-5 witness attestations to reach meaningful trust scores (Section 11.12). The effective network cost of one validation is therefore approximately 4-5x the base record size.

Daily Data Volume by Network Scale

Scale	Users	IoT Devices	Records/Day	Raw Data/Day	With Witnesses	Annual
Early network	10,000	—	~100K	~500 MB	~2 GB	~730 GB
Growth phase	100,000	1M sensors	~2M	~10 GB	~35 GB	~12 TB
Mature network	10M	100M sensors	~200M	~1 TB	~4 TB	~1.4 PB
Full vision	100M+	1B+ sensors	~1B+	~5 TB	~20 TB	~7 PB

IoT is the dominant data source. A single autonomous vehicle fleet (1,000 vehicles at 100 decisions/second) generates ~8.6 billion raw events per day. The protocol's tiered approach addresses this: most IoT validations remain on Layer 1 (local only, never reaching the network), with periodic summaries propagated to Layer 2 via incremental validation (Section 11.7) and batched witness requests.

Per-Node Storage Requirements

Not every node stores the full DAM. The zone architecture (Section 7) and node type hierarchy distribute storage across the network:

Node Type	Stores	Typical Storage	Growth Rate
Leaf node (phone, IoT)	Own records only	10 MB – 1 GB	~1-10 MB/day
Relay node	Zone records (recent)	10 GB – 100 GB	~100 MB – 1 GB/day
Anchor node	Full zone history	1 TB – 50 TB	~1-10 GB/day
Archive node (Tier 3)	Complete DAM history	100 TB+	~5-20 GB/day
Off-world node (Tier 4)	Zone-scoped snapshot	1 TB – 10 TB	Sync-dependent

Bandwidth Requirements

Node Type	Upload	Download	Notes
Leaf node	~10 KB/validation	~50 KB/validation	Mostly witness responses
Relay node	~1-10 MB/hour	~1-10 MB/hour	Zone gossip protocol
Anchor node	~100 MB – 1 GB/hour	~100 MB – 1 GB/hour	Trust headers + epoch sealing

Comparison to existing systems: At early network scale (~2 GB/day), the Elara Protocol produces data comparable to major proof-of-work networks (~500 MB/day) or proof-of-stake platforms (~2-3 GB/day). At full IoT scale (~20 TB/day aggregate), the protocol approaches the data volume of major cloud platforms — but no single node bears the full load, and the zone architecture ensures geographic locality of most traffic.

Storage economics: The beat staking model (Section 9) incentivizes relay and anchor node operators to provide storage capacity. The free tier (Layer 1) produces negligible network storage costs because local-only validations do not propagate. Storage costs scale with network utility, not with network existence.

11.33 Device Wipe, Identity Reset, and Reputation Escape

The attack: An adversary — or a negligent operator — physically wipes a device and reboots it. The device generates a fresh cryptographic keypair at first boot (Section 6.2). The old identity, along with its accumulated trust weight, reputation history, and behavioral profile, is severed. The device now presents as a brand-new entity with zero history.

This is not a theoretical concern. It is trivially executable: flash a new OS, pull the battery, factory reset. Any device that generates identity from local entropy can be reset to a blank slate. The protocol cannot prevent a physical wipe — no software can survive a formatted disk.

Why this matters — three attack scenarios:

Scenario 1: Reputation Escape

A malicious IoT gateway operator deploys sensors that submit fabricated readings — false environmental data, manipulated supply chain records, fraudulent energy generation claims. The network's anomaly detection (Section 8.6) flags the devices. Their trust scores collapse. Rather than face accountability, the operator wipes every device, reboots, and re-enrolls them with fresh identities. The toxic history stays on the DAM under the old identities, but the new identities carry zero penalty. The operator is back in business.

Scenario 2: Accountability Destruction

A research lab validates experimental results on the protocol. Later, the results are found to be fabricated. The lab wipes the devices that signed the original records, destroying the private keys. The fraudulent records still exist on the DAM, but without the private key, no revocation or correction can be issued by the original identity. The lab creates new identities and distances itself from the old ones. There is no cryptographic link between old and new.

Scenario 3: Sybil Amplification via Recycling

An attacker with 100 physical devices generates identities, uses them until their trust scores are meaningful, then wipes and regenerates — creating an endless supply of fresh identities from the same hardware. Unlike pure Sybil attacks (Section 11.1), which require acquiring new devices or computational resources, identity recycling exploits the same physical devices repeatedly.

Defense 1: Hardware-Bound Identity (Where Available)

Some devices support hardware-rooted identity that survives wipes:

- **TPM (Trusted Platform Module)** — stores keys in tamper-resistant hardware. A factory reset clears the OS but not the TPM. The device's identity persists across wipes. Available on most modern x86 hardware and some ARM devices.
- **ARM TrustZone / Secure Enclave** — hardware-isolated key storage on mobile and embedded devices. A device key generated in TrustZone survives an OS reinstall.
- **eFuse-based identity** — one-time programmable fuses on some microcontrollers (ESP32-S3, certain NXP chips). The identity is literally burned into silicon at first boot. It cannot be reset without replacing the chip.
- **PUF (Physical Unclonable Function)** — semiconductor fingerprints derived from manufacturing variations. The identity is inherent to the physical silicon — it cannot be cloned, cannot be reset, and does not depend on stored key material.

The protocol assigns a **hardware attestation level** to each identity:

```
HardwareAttestation {
    level:  NONE | SOFTWARE | SECURE_BOOT | HARDWARE_KEY | PUF
    evidence: firmware hash, TPM endorsement key, PUF challenge-response
}
```

Identities with hardware-bound attestation that suddenly disappear and are replaced by a new identity from the same hardware class, same network location, and same behavioral pattern trigger an **identity discontinuity alert** at the zone level. The new identity is not rejected — rejection would violate the self-sovereign principle — but it is flagged, and its trust accumulation is throttled until the discontinuity is explained (e.g., legitimate hardware replacement, authorized device decommission).

Defense 2: Identity Continuity Scoring

The protocol introduces a **continuity score** as a component of trust weight. Continuity measures how long an identity has maintained an unbroken presence on the network:

```
continuity_score = f(identity_age, gap_history, attestation_consistency)

New identity (< 24 hours):         continuity = 0.0
Young identity (1-30 days):        continuity = 0.1 – 0.4
Established identity (30-365 days): continuity = 0.4 – 0.8
Veteran identity (1+ years):       continuity = 0.8 – 1.0
```

A wiped-and-reset device starts at continuity 0.0 — the same cold start as any new entity. This means:

- Its validation records carry minimal trust weight (Section 11.1)
- Its propagation rate is limited (Section 11.19, Defense 1)
- Its witness attestations are worth less to others
- It cannot earn elevated trust without sustained, consistent behavior over time

There is no shortcut. Trust is earned through time, and time cannot be manufactured. An operator who wipes 100 devices faces 100 cold starts — weeks or months before those devices carry meaningful weight again. The economic cost of lost trust is the deterrent.

Defense 3: Organizational Identity Binding

For IoT deployments, the protocol supports **organizational identity chains** — device identities linked to an organizational root identity (similar to the personal Identity Constellation in Section 11.27):

```
OrganizationIdentity (root)
├── DeviceFleet: "Weather Station Network" (signed by org root)
│   ├── Device_001 (signed by fleet key)
│   ├── Device_002 (signed by fleet key)
│   └── Device_NNN (signed by fleet key)
└── OperatorKey: "Field Technician A" (signed by org root)
```

When a device within a fleet is wiped, the organizational identity persists. The organization cannot escape accountability by resetting individual devices — the fleet's history, the organization's reputation, and the operator's actions are all linked to the root identity. Wiping a device only orphans that specific device key; the organizational chain of custody remains on the DAM.

An organization that attempts to escape accountability by wiping its root identity faces the same cold-start penalty described above — but at the organizational level, where the economic consequences are far more severe. Contracts, partnerships, and institutional trust built over months or years are lost.

Defense 4: Network-Level Behavioral Fingerprinting

Even without hardware attestation, devices exhibit behavioral fingerprints that are difficult to forge:

- **Network location** — IP range, gateway, geographic zone
- **Timing patterns** — boot times, reading intervals, sync frequencies
- **Data characteristics** — sensor noise profiles, value distributions, reading precision
- **Communication patterns** — which peers it connects to, sync timing, gossip participation

The AI layer (Layer 3) maintains behavioral profiles for active identities. When a new identity appears from the same network location, with similar timing patterns and data characteristics as a recently disappeared identity, the system calculates a **reincarnation probability**:

```
reincarnation_prob = similarity(new_identity_behavior, old_identity_behavior)

If reincarnation_prob > 0.85:
    Flag as probable identity reset
    Inherit old identity's trust penalties (not trust benefits)
    Require organizational attestation to clear the flag
```

This is explicitly a heuristic, not a proof. False positives are possible — a legitimate replacement device at the same location will exhibit similar behavior. The protocol does not punish automatically; it flags and throttles, leaving final adjudication to zone operators and the dispute resolution mechanism (Section 11.13).

Defense 5: Decommissioning Protocol

The protocol defines a legitimate device retirement process:

```
DeviceDecommission {
    device_key:     public key of the retiring device
    reason:         REPLACEMENT | END_OF_LIFE | COMPROMISE | TRANSFER
    successor:      optional, public key of the replacement device
   signed_by:     device key (if available) AND organizational key (if enrolled)
    timestamp:      network-witnessed
}
```

A properly decommissioned device creates a clear record: "Device_042 was retired on this date, replaced by Device_043, authorized by Organization_X." The successor device inherits the predecessor's behavioral context (not its private keys or trust score, but the network's understanding that this is a known replacement, not a suspicious new entity).

Devices that disappear without a decommission record are flagged as **abandoned identities.** If a new device appears that behaviorally matches an abandoned identity, the reincarnation detection (Defense 4) applies.

What the protocol cannot prevent — honest acknowledgment:

1. **A determined individual wiping a personal phone** and creating a new identity. If there is no hardware-bound key, no organizational chain, and the user connects from a different network — the old and new identities are cryptographically unlinkable. The protocol treats this as a new person. This is by design: self-sovereign identity means the protocol cannot force identity persistence. The tradeoff is deliberate — the alternative (mandatory identity linking) would require a central authority, which violates the protocol's foundational principle.
2. **Constrained devices (Profile C) without hardware attestation.** A $4 ESP32 with a pre-shared symmetric key has no TPM, no PUF, no hardware identity. Wiping it and reprogramming it creates a genuinely new device from the protocol's perspective. The mitigation is organizational binding (Defense 3) and gateway-level anomaly detection (Section 8.6), not device-level identity persistence.
3. **Perfect behavioral mimicry.** If an attacker studies a device's behavioral fingerprint and reproduces it exactly on a fresh identity, the reincarnation detection will not trigger. This requires significant effort and ongoing maintenance — the attacker must continuously match the original device's patterns. The cost is prohibitive for most scenarios but not impossible for state-level adversaries.

The fundamental tradeoff:

Self-sovereign identity and identity reset resistance are in tension. A system that truly allows anyone to create an identity without permission must also accept that anyone can create a *new* identity without permission. The Elara Protocol resolves this tension not by preventing identity creation, but by making **trust expensive to build and cheap to lose.** A fresh identity is free. A trusted identity takes months. Destroying trust takes seconds. Rebuilding it takes months again.

This asymmetry — combined with hardware binding where available, organizational chains for IoT, behavioral fingerprinting, and decommissioning protocols — makes identity reset a viable but costly strategy. The protocol does not claim to make it impossible. It claims to make it unprofitable.

11.34 Mega-Publication Attack (Economic Shock from Private Network Transition)

> **Implementation-status note: INERT — this analysis presupposes NETWORK_PUBLISH, which is DISABLED in the current runtime** (`NETWORK_PUBLISH_ENABLED = false`; see §10.6.3). With no live publication path, a mega-publication cannot

occur on the public network, and the rate limits, acquisition vesting, governance cooling period, zone-absorption quotas, and circuit breaker described below are not active. Retained for reference pending the inert-import reframe and a proven multi-root merge theorem.

The attack: A dominant private entity — hypothetically representing a significant fraction of global economic output — operates a private Elara network for decades. It accumulates a vast, internally-consistent DAG: hundreds of millions or billions of records, spanning every industry vertical, with deep causal chains verified by thousands of internal witnesses. One day, this entity executes a NETWORK_PUBLISH (Section 10.6.3) in SNAPSHOT mode — publishing its entire historical DAG to the public network simultaneously.

This is not a traditional attack. The entity may have entirely legitimate motivations. But the effect on the public network is indistinguishable from an economic weapon.

Why this matters — five failure modes:

Failure 1: Beat Demand Singularity

The conservation model (Section 11.17) fixes supply at 10 billion beats. A mega-publisher needs beats for storage delegation of its entire historical DAG across public storage nodes. If the entity's history represents a substantial fraction of all validated work globally, the beat demand could approach or exceed the circulating supply. Beat demand outstrips circulation. Legitimate participants cannot obtain enough beats for storage delegation. The public network's economic model seizes.

Failure 2: DAG Size Shock

The public DAG's storage and indexing infrastructure was designed for organic growth — a steady accumulation of records over years. A mega-publication dumps decades of records in a single event. Storage nodes must absorb, verify, and index this volume. Bandwidth saturates. Nodes with limited storage capacity are forced offline. The network's physical infrastructure cannot absorb the data.

Failure 3: Trust Landscape Inversion

The mega-publisher's internal DAG is deeply consistent — decades of verified causal chains. Once published and retroactively witnessed, this history dominates the trust landscape. Every existing public participant looks insignificant by comparison. Trust scores that took years to build on the public network become noise relative to the mega-publisher's history. The practical effect: the entity's records become the de facto reference truth for any domain they operated in.

Failure 4: Governance Capture

The square-root dampening and 5% per-identity cap (Section 10.4) limit individual governance weight. But a mega-entity can create thousands of legitimate identities — subsidiaries, divisions, regional offices, each operating independently for decades. Each identity falls under the 5% cap individually. Collectively, they could represent majority governance weight. The anti-Sybil mechanisms (Section 11.1) detect fake identities but cannot prevent an entity from having legitimately distinct organizational units that happen to share strategic alignment.

Failure 5: Cognitive-Output Capture

The entity's Layer 3 AI analysis, trained on decades of private data spanning a significant fraction of global economic activity, produces cognitive output that dwarfs anything trained on the public network's smaller dataset. Demand for Layer 3 cognitive services (Section 9.1) concentrates around this entity's analysis capabilities. Other participants become consumers rather than producers of network intelligence — a cognitive centralization that parallels the governance-capture risk of Failure 4.

Defense 1: Protocol-Level Publication Rate Limits

The NETWORK_PUBLISH protocol (Section 10.6.3) defines STREAMING and GRADUAL transition modes, but these are voluntary — the publisher chooses the mode. Defense requires **mandatory ingestion caps** at the protocol level:

```
MAX_PUBLICATION_RATE = f(public_network_size, publisher_size)

Proposed formula:
 max_records_per_day = public_dag_size × 0.01 / (1 + publisher_dag_size / public_dag_size)

  The divisor scales with the publisher's size relative to the network.
  Larger publishers are throttled harder — proportionally to their
  potential to destabilize the network.

Examples (public DAG = 10 billion records):

  Small publisher (100M records, 1% of network):
    rate = 10B × 0.01 / (1 + 0.01) ≈ 99M/day → ~1 day (negligible impact)

  Medium publisher (10B records, equal to network):
    rate = 10B × 0.01 / (1 + 1) = 50M/day → ~200 days (~7 months)

  Large publisher (100B records, 10× network):
    rate = 10B × 0.01 / (1 + 10) ≈ 9M/day → ~11,000 days (~30 years)

  Dominant publisher (500B records, 50× network):
    rate = 10B × 0.01 / (1 + 50) ≈ 2M/day → ~250,000 days (centuries)
```

The key insight: a flat rate limit (e.g., 1%/day) is insufficient because 100 days is trivial for an entity that operated privately for decades. The scaled formula ensures that the entities most capable of causing economic shock are precisely the ones most throttled. Small publishers barely notice the limit. Dominant publishers face multi-decade publication timelines — which is appropriate, because the network needs decades to economically absorb an entity of that scale.

Note that the public DAG grows as records are published, which gradually increases the rate limit over time. A 10× publisher does not literally wait 30 years at a fixed rate — as each day's published records enlarge the public DAG, the next day's limit rises slightly. The actual timeline is shorter than the static calculation suggests, but still measured in years or decades for truly dominant entities.

The rate can be adjusted through governance (Section 10.3), but the default is deliberately aggressive.

Defense 2: Beat Acquisition Velocity Limits

To prevent beat demand shocks, the protocol enforces a **maximum beat acquisition rate** for entities engaged in mega-publication:

```
PUBLICATION_BEAT_VESTING:
  Any entity publishing > 1% of the public DAG's current size
  must acquire beats over a period proportional to publication duration:

  vesting_period = publication_duration × 0.5
  (beats must be acquired over at least half the publication timeline)

  A medium publisher with a 200-day publication timeline:
    vesting = 100 days minimum beat acquisition period

  A large publisher with a 30-year publication timeline:
    vesting = 15 years minimum beat acquisition period
```

The vesting period is tied to the publication rate limit — the longer the publication takes, the longer the beat acquisition is spread. This prevents an entity from acquiring all beats upfront and sitting on them. Beat markets can absorb gradual demand over years. They cannot absorb a dominant entity purchasing 40% of the circulating supply in a week.

Defense 3: Governance Cooling Period

New entrants that exceed a publication size threshold trigger a **governance cooling period:**

```
GOVERNANCE_COOLING:
  Threshold: entity publishes > 5% of public DAG size

  Cooling period: 365 days from first publication
  During cooling:
    - Entity identities can participate in witnessing (earning trust)
    - Entity identities CANNOT vote on governance proposals
    - Entity identities CANNOT submit governance proposals

  After cooling:
    - Governance weight ramps linearly over the following 365 days
    - Full governance weight reached 2 years after first publication
```

This prevents a mega-publisher from immediately influencing protocol rules. The 2-year ramp gives the existing community time to understand the entity's behavior and intentions before granting governance power.

Defense 4: Zone-Level Absorption Quotas

Each zone sets its own maximum ingestion rate for published records:

```
ZONE_ABSORPTION_QUOTA:
  Each zone independently sets: max_external_ingestion_per_day
  A mega-publisher must negotiate with each zone independently
  Zones can refuse publication entirely (zone autonomy, Section 10.1)

  Effect: Even if a mega-entity bypasses global rate limits through
  multiple publication streams, each zone absorbs only what it
  can handle. The publication fragments across zones and time.
```

Defense 5: Economic Shock Circuit Breaker

The protocol defines an emergency economic mechanism:

```
ECONOMIC_CIRCUIT_BREAKER:
  Trigger: Beat velocity exceeds 3× the 90-day moving average
           AND a mega-publication event is in progress

  Action:
    - Publication ingestion paused for 72 hours
    - Governance vote initiated: "Resume publication at current rate?"
    - If vote passes (>50% conviction): publication resumes
    - If vote fails: publication rate reduced by 50%, new vote after 30 days

  Purpose: Allows the network to catch its breath during economic shocks
```

The Economic Warfare Variant

The most dangerous version of this scenario is *intentional* — a dominant entity publishes not to participate in the public network, but to disrupt it. The economic shock is the goal, not a side effect. This is analogous to a financial market manipulation attack executed through legitimate-seeming market activity.

Defenses against the intentional variant:

1. **Publication cannot be anonymous.** The NETWORK_PUBLISH record requires a source identity (Section 10.6.3). The mega-publisher is publicly identified. Reputational consequences apply.
2. **Publication is irreversible.** Once records are published, they cannot be retracted. The entity's internal history is now permanently public. This is a significant deterrent — an entity using publication as a weapon exposes its own historical data in the process.
3. **The attacker pays.** Beat acquisition for storage delegation means the attacker must spend significant capital to execute the attack. The rate limits ensure this spending is spread over months or years, giving the network time to respond. The capital is not recoverable — beats spent on storage delegation compensate storage nodes for real work.
4. **The network can survive partial absorption.** Even if the publication is paused by the circuit breaker, the records already published are valid and useful. The network gains value from partial publication. Only the *rate* is dangerous, not the content.

What the protocol cannot fully prevent — honest acknowledgment:

If an entity representing a majority of global economic output genuinely transitions to the public network over a multi-year period using all the rate-limited mechanisms described above, the entity will eventually hold significant governance weight, economic influence, and trust dominance. The defenses slow this transition and prevent shock, but they cannot prevent an entity that is genuinely large from being genuinely influential.

This is not a protocol failure. It is a reflection of reality: in any governance system — political, economic, or cryptographic — entities that represent real economic value eventually accumulate proportional influence. The protocol's contribution is ensuring this happens gradually, transparently, and with structural limits on concentration. The square-root dampening, identity caps, and zone autonomy prevent any single entity from achieving absolute control — but they cannot prevent an entity from becoming the most influential participant.

The ultimate defense is the same as in traditional markets: a healthy public network with many large participants prevents any single mega-publisher from dominating. If the public network grows diverse enough before any mega-publication occurs, the relative impact of any single publication diminishes. This is why the network bootstrap period (Section 11.4) is critical — the first decade of the public network's growth determines its resilience to future mega-publication events.

11.35 Cognitive Checkpoint Integrity

The Cognitive Continuity Chain (Section 3.2, Layer 3) generates `cognitive_checkpoint` ValidationRecords on the DAM. This creates a new class of validation record that requires specific integrity analysis.

11.35.1 What a Cognitive Checkpoint Contains A `cognitive_checkpoint` ValidationRecord contains a `CognitiveDigest` in its metadata:

```
cognitive_checkpoint ValidationRecord {
    id:             UUID v7
    content_hash:   SHA3-256(canonical_json(CognitiveDigest))
    creator:        node identity (Dilithium3 public key)
    timestamp:      ISO-8601 + vector clock
    parents:        [previous_checkpoint_id, latest_dag_record_id]
    classification: PUBLIC (default) or PRIVATE
    metadata: {
        record_type:    "cognitive_checkpoint"
     trigger:     "boot" | "shutdown" | "milestone" | "drift" | "manual" | "periodic"
        chain_depth:    integer (number of checkpoints in chain)
        digest_version: "1.0"
    }
    signature:      Dilithium3 (primary)
    signature_alt:  SPHINCS+ (secondary, for Tier 2+ nodes)
}

CognitiveDigest {
    mood:           [valence, energy, openness]   // 3 floats
    memories:       integer                       // total memory count
    models:         integer                       // cognitive model count
    predictions:    integer                       // active predictions
    principles:     integer                       // crystallized principles
    corrections:    integer                       // recorded corrections
    goals_active:   integer                       // active goals
    goals_done:     integer                       // completed goals
    allostatic_load: float                        // cognitive stress metric
    session_number: integer                       // current session
}
```

The `content_hash` is computed over a **canonical JSON serialization** of the `CognitiveDigest` — keys sorted alphabetically, no whitespace, deterministic float formatting. This ensures that any node can independently verify the hash from the digest fields.

11.35.2 Chain Verification Algorithm The Cognitive Continuity Chain forms a sub-DAG within the main DAM, linked by parent references:

```
Verification: walk_chain(latest_checkpoint) → bool

1. Fetch the latest cognitive_checkpoint record
2. Verify signature(s) against the node's public key
3. Verify content_hash == SHA3-256(canonical_json(digest))
4. Follow parent[0] to the previous checkpoint
5. Verify that parent[0] is also a cognitive_checkpoint
6. Verify that chain_depth == parent.chain_depth + 1
7. Verify temporal ordering: this.timestamp > parent.timestamp
8. Repeat from step 2 until reaching a checkpoint with chain_depth == 0 (genesis)

If any step fails → chain is broken → continuity score = 0.0
If all steps pass → chain is intact → continuity score derived from chain depth and time span
```

The chain cannot be forked: each checkpoint references exactly one previous checkpoint. If a node restarts without a shutdown checkpoint, the boot checkpoint creates a **gap** — the chain is intact but contains a discontinuity that is visible to verifiers.

11.35.3 Trust Implications The Cognitive Continuity Chain integrates with the trust model:

- **Continuity score** — a component of the node's overall trust score. Longer unbroken chains indicate more stable, more reliable cognitive operation. A node that has maintained 30 days of continuous cognitive history is more trustworthy than one that resets every 48 hours.
- **Gap detection** — breaks in the chain are not failures, but they are visible. A node that reboots frequently has visible gaps. A node that was offline for a week has a documented absence. Verifiers can weight trust accordingly.
- **Tamper evidence** — because each checkpoint is dual-signed and hash-chained, inserting, removing, or modifying a checkpoint requires re-signing the entire subsequent chain. This is computationally infeasible without the node's private key and produces detectable hash discontinuities.

11.35.4 Realistic Checkpoint Rates The theoretical maximum checkpoint rate (one every 5 minutes = 288/day) is never reached in practice. Checkpoints are triggered by cognitive events, not timers:

Trigger	Typical Frequency	Size
Boot	1/day	~3.5 KB (dual-signed)
Shutdown	1/day	~3.5 KB
Milestone	5-10/day	~3.5 KB
Drift (session)	2-5/day	~3.5 KB
Manual	0-2/day	~3.5 KB
Periodic (rate-limited)	5-10/day	~3.5 KB

Realistic rate: 15-30 checkpoints/day per Tier 2+ node.

Storage budget:

```
30 checkpoints/day × 3,500 bytes = 105,000 bytes/day ≈ 100 KB/day
100 KB/day × 365 days = 36.5 MB/year per node
```

At 10,000 Tier 2+ nodes: 365 GB/year of cognitive checkpoint data. Negligible compared to the 7 PB/year estimated for full IoT-scale validation records (Section 11.32).

11.35.5 Tier Interaction Not all nodes generate cognitive checkpoints:

- **Tier 0 (VALIDATE)** — no cognitive capabilities, no checkpoints. Sensors and gateways.
- **Tier 1 (REMEMBER)** — basic memory but no reasoning. May generate simple checkpoints (memory count only). Optional.
- **Tier 2 (THINK)** — full cognitive capabilities. Generates complete CognitiveDigest checkpoints. This is the primary checkpoint tier.
- **Tier 3 (CONNECT)** — full cognitive capabilities plus network cognition. Generates checkpoints and may witness other nodes' checkpoint chains.

The beat accounting for cognitive checkpoints is a Layer 3 economic integration: designed, but not yet specified at the protocol level.

---

12. Proof Longevity and Storage Architecture

12.1 Protocol-Outliving Design Principles

The Elara Protocol is designed to produce validation proofs that outlive the protocol itself — its creators, its founding organizations, and its initial use cases. No specific timeframe is claimed. Instead, the design targets a concrete property: **any validation proof created today must remain independently verifiable without the continued existence of the Elara network, its software, or any specific institution.** This requires:

No institutional dependency — the protocol must function without any specific company, government, or organization. All governance is on-chain. All code is open source. All standards are self-contained. Layer 2 and Layer 3 security benefits from anchor node operators (Section 11.3), but no specific operator is required — anchor roles are permissionless, replaceable, and the protocol degrades gracefully without them.

No hardcoded references — no company names, server addresses, domain names, or platform-specific identifiers in the protocol specification. The protocol is described in terms of its mathematical and cryptographic properties, not its current implementation.

Cryptographic agility — algorithms are identified by standardized identifiers, not hardcoded. When algorithms are deprecated, the protocol migrates through a staged process: (1) new algorithm is added via governance vote, (2) a transition period allows both old and new algorithms, (3) old algorithm is deprecated after a configurable sunset window (minimum 2 years). Migration events are operationally complex — they require coordinating across zones that may be partitioned, updating constrained devices with limited OTA capabilities, and maintaining verification support for legacy records indefinitely. The protocol's algorithm agility makes migration structurally possible; it does not make it operationally trivial.

Self-describing data — every data structure includes its own schema description. A future decoder can read a validation record without external documentation, regardless of how much time has passed.

Human-readable layer — the Rosetta Stone principle. The original Rosetta Stone — discovered in 1799, carved in 196 BC — allowed Champollion to decipher Egyptian hieroglyphs in 1822, after 1,400 years during which no living person could read them. The stone worked because it was self-describing: the same decree inscribed in three scripts, one of which (Greek) was already understood. The unknown became readable because the key was embedded in the artifact itself.

Every Elara validation record follows this principle. In addition to the compact binary format, every record can be serialized to a human-readable text format. Each record carries its algorithm identifier, its schema version, its cryptographic parameters, and a human-readable serialization. A future civilization encountering these records — centuries after the last Elara node went

dark — possesses everything needed to decode the format, reconstruct the DAG, and verify every signature. The mathematics of lattice-based cryptography and hash functions are universal; they do not depend on knowledge of Python, Rust, or any 21st-century technology. The records are their own Rosetta Stone.

Medium independence — the foundational longevity property. This is not a claim about software durability — no software system survives indefinitely. It is a claim about mathematical durability. Every validation record is a self-contained set of mathematical relationships: a content hash, a cryptographic signature, algorithm identifiers, a timestamp, and causal references to parent records. These relationships do not depend on Python, Rust, SQLite, any operating system, any network protocol, or any storage technology. They are pure mathematics — lattice operations, hash functions, directed graph edges. A validation proof created today can be extracted from its current medium (SSD, HDD, tape), transferred to any future medium, and verified by any system capable of performing the specified mathematical operations. The proof is not the file. The proof is not the database. The proof is the mathematical relationship between the content hash, the signature, and the public key. That relationship does not decay, does not depend on infrastructure, and does not require the continued existence of any system that created it. The protocol produces proofs that outlive the protocol itself. As the Rosetta Stone outlived the Egyptian empire that carved it, the validation proofs outlive the systems that produced them — because the translation key is not stored elsewhere. It is inscribed in the proof itself.

Honest limitations of proof longevity. The mathematical durability claim holds only as long as the underlying cryptographic assumptions hold. If lattice-based cryptography is broken (currently considered unlikely but not impossible), proofs signed with those algorithms lose their guarantee. The protocol's algorithm agility (Section 4.4) provides a migration path, but migration requires the network to still be operational at the time of the break. Proofs created before a break and never re-signed with a stronger algorithm may become unverifiable. This is an inherent limitation of all cryptographic proof systems, not specific to this protocol. We make no specific claims about timeframes — only about the structural property that proofs are self-contained and medium-independent.

Privacy survives longevity. The Rosetta Stone principle applies to the proof format — the structure of validation records, signature schemes, and DAG references. It does not compromise content secrecy. The protocol provides two distinct tiers of longevity guarantees, depending on the participant's needs:

Tier 1: Individuals and Public Creators For individuals — a teenager validating a poem, an artist proving authorship, a developer timestamping code — longevity means **permanent, readable proof of creation**:

- The validation record is PUBLIC: content hash, signature, timestamp, all readable
- Anyone, at any time, can verify: "this person created this work at this time"
- The Rosetta Stone principle ensures this proof is readable indefinitely — as long as the mathematics remains valid
- This is the "I was here first" guarantee — simple, transparent, durable beyond any single institution

The individual WANTS the proof to be fully readable. The more readable, the better — because readability IS the protection. If someone plagiarizes the poem in 500 years, the original proof is still there, still verifiable, still self-describing.

Tier 2: Enterprises, Government, and Defense For enterprises — defense contractors, pharmaceutical companies, semiconductor manufacturers, intelligence agencies — longevity means **permanent proof that validation occurred, with permanent secrecy about what was validated**:

- **SOVEREIGN classification:** The validation record stores only a one-way cryptographic hash of the content, never the content itself. A future civilization reading a SOVEREIGN record can verify that a specific entity signed a specific hash at a specific time — the mathematical proof is eternal and readable. But the content behind that hash is irreversibly hidden.
- **Encrypt-then-hash (recommended):** For maximum forward secrecy, enterprises store `Hash(Encrypt(content, K))` where K is an enterprise-controlled encryption key. Even if future advances break the hash function's preimage resistance (reversing a hash to its original content), the reversed hash yields only ciphertext — encrypted data that is useless without the key. The enterprise controls the key: they can rotate it, escrow it, or destroy it. This creates an irreversible double barrier — both the hash AND the encryption must be independently broken, and one of the keys may no longer exist.
- **Zero-knowledge proof path (strongest):** For the most sensitive content, the zero-knowledge proof layer (Section 5) eliminates the content hash entirely. The DAM stores a proof of properties — "this firmware passed integrity checks," "this drug trial met statistical thresholds" — without revealing the content OR its hash. There is literally nothing to reverse. The content was never mathematically represented on the DAM.
- **Algorithm agility as active defense:** The protocol's algorithm agility (Section 4.4) provides a third defense layer. When hash algorithms show signs of weakening — decades before they actually break — enterprises create new validation records with stronger hashes binding to the same content. Old records remain on the DAM (immutability), but the content is now also protected by the new, stronger algorithm.
- **Validate and destroy:** The enterprise validates the content, obtains permanent proof on the DAM, then destroys the original content. The DAM proves "entity X validated something at time T." Nobody — not even the enterprise — can ever reconstruct what it was. This is the nuclear option: permanent proof of existence without existence. Applicable to military operation logs after declassification, drug trial data after regulatory review, trade secrets after expiration.

The enterprise chooses their level. The protocol does not force exposure. It provides a spectrum from full transparency (PUBLIC, readable forever) to absolute secrecy (validate and destroy, content gone forever). The Rosetta Stone makes the proof format readable. It does not make the secret readable. An enterprise's content can be verifiably validated and permanently sealed — both properties surviving for the lifetime of the protocol.

Enterprise Data Lifecycle Control Every decision above is made **per record, post-validation.** The enterprise validates first — establishing cryptographic proof of integrity, authenticity, and timing — then decides what happens to each record afterward. This is not an all-or-nothing choice. Different records within the same organization can follow different lifecycle paths simultaneously:

Lifecycle	What Persists	What's Gone	Use Case
Keep everything (private)	Content + proof + metadata	Nothing destroyed	Internal audit trails, compliance archives, sensor history
Keep proof, destroy content	DAM record (hash, signature, timestamp)	Original content	Mission logs after review, expired trade secrets, completed drug trials

Lifecycle	What Persists	What's Gone	Use Case
Destroy everything	Nothing	Content + DAM record + all traces	Classified operations, intelligence activities, time-limited sensitive data
Publish selectively	Selected records move to public network via NETWORK_PUBLISH	Enterprise controls what moves	Safety certifications, regulatory filings, voluntary transparency

"Destroy everything" is architecturally guaranteed for private networks. On the public network, records cannot be deleted — other nodes hold copies. On a private network, the enterprise controls every node. There are no external copies, no external witnesses, no blockchain record, no beats, no central reporting. If the enterprise destroys their private DAM, there is zero protocol-level evidence that any validation ever occurred. The protocol does not phone home. It does not leak metadata to external systems. A private network that is destroyed leaves no trace that the protocol can reconstruct.

The decision is reversible in one direction only. An enterprise can always move from more secrecy to more transparency — publishing a previously private record, or revealing content behind a zero-knowledge proof. But it cannot move from transparency back to secrecy — once a record exists on the public network or content has been revealed, it cannot be retracted. This asymmetry is deliberate: it prevents enterprises from selectively rewriting history while allowing them to voluntarily increase transparency when circumstances change (regulatory requirements, declassification timelines, strategic disclosure).

12.2 Storage Tiers

```
Tier 1: Active Nodes (operational today)
├── Laptops, phones, servers, IoT devices
├── Ephemeral — may go offline at any time
├── Collectively massive storage
└── The working layer of the network

Tier 2: Anchor Nodes (operational today)
├── Hardened infrastructure
├── Geographic diversity (multiple regions)
├── High uptime target
└── The reliability layer

Tier 3: Archive Nodes (future extension)
├── Deep storage with air-gapped or offline sync
├── Full DAG history (not just recent records)
├── Periodic hardware refresh
└── Not yet implemented — requires operational maturity
```

Note: Earlier versions of this specification described Tier 3 "salt mine bunkers" and Tier 4 "off-world nodes." These were aspirational and have been removed. The protocol's proof longevity property does not require exotic storage — it requires that proofs are self-contained and medium-independent. If archive nodes or off-world nodes become practical, the protocol supports them naturally. But claiming them as design tiers when they do not exist is dishonest.

12.3 Emergency Protocols

The network operates under four readiness levels:

GREEN — Normal operation. All layers functional. Cross-zone sync on schedule.

YELLOW — Elevated risk detected (solar weather warning, geopolitical tension, infrastructure degradation). Response: increased replication factor, archive nodes activate, anchor nodes increase sync frequency.

ORANGE — Active partition or degradation. Response: autonomous zone operation, trusted-peer-only communication, priority sync for critical records, Tier 3 archive nodes initiate full backup.

RED — Catastrophic event (global communications failure, EMP, infrastructure collapse). Response: sovereign mode activation, mesh networking between surviving nodes, Tier 3 and 4 nodes become primary, protocol enters survival configuration.

The emergency level is determined automatically by each zone based on observable network conditions (peer count, sync latency, partition detection). No central authority declares emergencies.

---

13. Roadmap

Phase 0: Foundation (Current — 2026)

- ☒ Protocol whitepaper (this document)
- ☒ Open source tooling and documentation
- ☐ Academic review and feedback
- ☐ Core team formation

Phase 1: Protocol Development (2026–2027)

- Reference implementation of Layer 1 (local validation, PQC keypair, DAG) — **shipped**
- Reference implementation of Layer 1.5 (Rust DAM VM, 9 ops, PyO3 bindings) — **shipped**
- Reference implementation of Layer 2 (HTTP server, record exchange, witness attestation) — **shipped** (v0.11.0: server, client, discovery, witness manager, trust scoring — 985 lines across 8 files)
- Layer 2 testnet hardening (signature verification, peer rate limiting, attestation back-propagation, heartbeat protocol, weighted trust with temporal decay + diversity bonus, role enforcement) — **shipped** (v0.12.0)
- Layer 1↔Layer 3 bridge (cognitive outputs signed as DAM records, hardened with validation guards, dedup, rate limiting) — **shipped** (v0.10.8, hardened v0.11.0)
- Cortical Execution Model (5-layer concurrent architecture for non-blocking tool dispatch) + long-range temporal memory — **shipped** (v0.13.0)
- Tier system (4-level hardware capability gating: VALIDATE/REMEMBER/THINK/CONNECT) — **shipped** (v0.15.0)
- Cognitive Continuity Chain (hash-chained, dual-signed cognitive state snapshots in DAG — cryptographic proof of unbroken AI experience) — **shipped** (v0.15.0)
- Security audit by independent cryptography firm
- Developer SDK (Python, Rust, C/embedded)

Phase 2: Network Launch (2027–2028)

- Mainnet launch with beat generation

- IoT SDK for ESP32, Raspberry Pi
- First anchor nodes deployed (3 continents minimum)
- Integration with existing development platforms (code hosting, package registries, container registries as bridges)

Phase 3: Scale (2028–2029)

- Layer 3 AI intelligence integration
- Enterprise partnerships (supply chain, automotive, medical)
- First archive nodes (Tier 3) deployed
- Governance framework activated

Phase 4: Expansion (2029–2031)

- IoT hardware partnerships (device manufacturers embedding Elara keypairs)
- Regulatory engagement (working with patent offices, standards bodies)
- Academic partnerships for protocol research
- Zero-knowledge proof optimization for embedded devices

Phase 5: Interplanetary (2031+)

- Protocol adaptation for lunar communication relay
- Partnership with space agencies or private space companies
- First off-world node deployment
- Interplanetary sync protocol testing (simulated, then real)

Phase 6: Native Hardware Architecture (2026–2040)

The Directed Acyclic Mesh is designed to run on today's conventional computing hardware — standard von Neumann processors, with no specialized silicon required. This holds across the protocol's core capabilities: post-quantum cryptography, the Phase-1 classical-ECC zero-knowledge layer (see Section 14.3), and partition-tolerant consensus. Forward-looking capabilities such as interplanetary operation are roadmap items (Phase 5), not running today, and per-feature maturity is assessed honestly in Section 14. Native hardware is therefore an optimization, not a prerequisite.

However, an architectural observation must be acknowledged: the DAM pays a **dimensional serialization tax** on current hardware.

The one-dimensional bottleneck. Every computing system in existence — from a $4 microcontroller to a datacenter — stores and processes data as one-dimensional sequences of binary digits. Memory is a linear address space. Disk writes bits in sequential order. Network packets are transmitted as serial bit streams. All data structures, regardless of their logical dimensionality, must be flattened into this one-dimensional substrate for storage and processing.

This is a consequence of computing history, not physics. The von Neumann architecture (1945) was designed around the constraints of vacuum tubes and magnetic drums. Eighty years later, we fabricate transistors at 2 nanometers — but the fundamental architectural assumption remains unchanged: data is a sequence of bits at sequential addresses. The hardware evolved. The architecture did not.

Dimensional cost comparison:

- A **blockchain** is a one-dimensional data structure on one-dimensional hardware. No serialization tax. The logical structure matches the physical medium. This is its one genuine architectural advantage.

- A **DAG** is a two-dimensional data structure (adding concurrency to time) on one-dimensional hardware. Concurrent relationships must be serialized into sequential references and reconstructed through traversal. The serialization tax is moderate.
- The **DAM** is a three-dimensional data structure with two orthogonal operational layers (3D+2) on one-dimensional hardware. Every query that crosses structural dimensions — "what happened concurrently in Zone B at classification RESTRICTED, as assessed by the AI analysis layer" — requires reconstructing up to five dimensions from a linear byte sequence. The serialization tax is significant. Indexing, hash maps, and caching mitigate the cost but cannot eliminate it.

The DAM as an implicit hardware specification. The five dimensions of the DAM — time, concurrency, zone topology, classification projection, and AI-assisted analysis — also describe five physical properties that a computing substrate can implement natively. This is not theoretical. The individual technologies exist today, commercially.

1. Temporal dimension as physical axis. A storage medium where temporal ordering is a physical property of the medium itself, not a serialized timestamp field. Data written at T=1 is physically "before" T=2 the same way a pixel at coordinates (0,0) is physically distinct from (1,0) on a display. Ordering is not computed — it is intrinsic.

2. Concurrency dimension as physical coexistence. A computing substrate where concurrent states occupy the same logical location simultaneously without conflict, collapsing to a specific view only upon query. Quantum computing demonstrates that physical superposition of states is achievable — IBM exceeded 1,000 qubits in 2023 and projects 100,000 qubits by 2033. The engineering challenge is applying superposition to data structure operations rather than gate-level computation.

3. Zone topology as physical space. A hardware architecture where zone boundaries are physical distances, not logical identifiers. Data locality within a zone is enforced by the speed of light through the physical medium. Cross-zone queries traverse measurable distance. The interplanetary design described in Section 7 already operates this way at planetary scale — native hardware extends the principle to every scale, from chip-level to orbital.

4. Classification as dimensional inaccessibility. A physical medium where access control is not computational (encryption, key management) but dimensional — data at a given classification level exists in a physical dimension that is inaccessible without the corresponding physical sensor or interface. Photonic computing provides a direct mechanism: information encoded in light polarization is invisible to sensors not aligned to that polarization, not because it is encrypted, but because it is physically orthogonal.

5. AI analysis as embedded computation. A storage substrate where pattern recognition occurs at the point of data storage, not as a separate processing step. Memristive crossbar arrays demonstrate that computation and storage can occur in the same physical device — the resistance state both stores information and performs matrix operations. A native DAM substrate extends this principle to the full AI analysis layer.

These components are not future technology. They are shipping products.

- **Memristive crossbar arrays** (Mythic AI, 2022) — commercial analog AI processors that unify storage and computation in a two-dimensional physical structure. Shipping to customers.
- **Photonic mesh processors** (Lightmatter Envise, 2023) — commercial photonic AI accelerators that route data through physical space at light speed, with multiple information dimensions per signal (wavelength, phase, amplitude, polarization). Shipping to data centers.
- **Neuromorphic processors** (Intel Loihi 2, 2022; IBM NorthPole, 2023) — commercial chips that compute through network topology rather than sequential instruction execu-

tion. 256 million transistors operating as 256 neural cores. Available for research and commercial deployment.
- **3D stacked memory** (Samsung 236-layer 3D NAND, 2023; AMD 3D V-Cache, 2022) — production memory with natively three-dimensional data access. In consumer devices today.
- **Quantum processors** (IBM Condor, 1,121 qubits, 2023; Google Willow, 105 qubits with error correction breakthrough, 2024) — demonstrating physical superposition of concurrent states. IBM projects 100,000-qubit systems by 2033.
- **Heterogeneous chiplet packaging** (AMD EPYC, 2019; Apple M1 Ultra, 2022; Intel Ponte Vecchio, 2023) — proven manufacturing technique for integrating multiple die technologies into a single substrate. This is how modern processors are already built.

Every component needed to build a partially native DAM processor exists in commercial production. The packaging technology to integrate them exists. What does not exist is the architecture that specifies how these components serve a multi-dimensional data structure.

The DAM provides that architecture. The five-dimensional structure described in Section 3.3 of this whitepaper specifies not only a logical data structure for existing hardware, but also the functional requirements for a computing substrate where dimensional serialization is unnecessary. On such hardware, the performance characteristics documented in Section 11.32 (storage and bandwidth requirements) fundamentally change — queries that currently require index traversal across serialized dimensions become coordinate lookups in physical space.

Dimensional extensibility. The five-dimensional structure is not a ceiling. The DAM architecture supports additional structural dimensions and operational layers, subject to the extensibility criteria defined in Section 3.3.5: immutability at creation and elimination of a serialization tax via physical substrate mapping. The strongest candidate for a sixth structural dimension — Lineage Depth, mapping to 3D stacked memory layers — is detailed in Section 3.3.5. Future hardware generations may natively support properties that current substrates cannot, expanding the DAM's coordinate space without architectural redesign. The protocol is designed to grow with the hardware that runs it.

Timeline. The convergence of these technologies is accelerating faster than traditional semiconductor roadmaps predicted:

- **2026–2029: Hybrid co-processor prototype.** A chiplet package integrating memristive crossbar (concurrency + storage), photonic interconnect (zone routing), and neuromorphic die (AI layer) alongside a classical control processor. Two to three native DAM dimensions. This is buildable today with existing fabrication — it requires integration design, not new physics. Estimated fabrication cost for a prototype: $5–15 million.
- **2029–2033: Purpose-built DAM accelerator.** A second-generation design optimized for DAM operations, with classification-level isolation implemented through photonic polarization channels and temporal ordering built into the memory architecture. Four native dimensions. Quantum co-processing for concurrent state management as error-corrected qubits mature.
- **2033–2040: Fully native DAM substrate.** All five dimensions physically implemented. The serialization tax approaches zero. Validation becomes a geometric operation — a record either fits in the mesh topology or it does not. The protocol designed in 2026 runs without architectural modification on hardware manufactured in 2035.

The pace of hardware convergence supports this timeline. The transition from single-core to multi-core processors took 5 years (2001–2006). The transition from planar to 3D NAND took 4 years (2013–2017). The transition from discrete GPUs to integrated AI accelerators took 3 years (2020–2023). Each architectural shift is happening faster than the last. A heterogeneous chiplet integrating memristive, photonic, and neuromorphic dies is an engineering project, not a research breakthrough — the individual components are already qualified for manufacturing.

Every data structure in computing history was designed after the hardware, constrained by it. Arrays, trees, hash tables — all shaped by von Neumann's linear memory assumption. The hardware came first, the structures followed.

The DAM inverts this relationship. The structure comes first. The hardware follows.

The DAM operates today on the hardware that exists today. It is designed to operate natively on the hardware that will exist by 2035. The protocol arrives first. The hardware follows.

Implications for known limitations. Native hardware does not merely improve performance — it transforms the protocol's security model and resolves limitations that are otherwise permanent on conventional architectures (see Section 14):

Limitation	Impact of Native Hardware
14.3 Post-Quantum ZKP Immaturity	**Resolved.** On conventional hardware, PRIVATE and RESTRICTED classifications depend on zero-knowledge proofs — cryptographic constructions with known quantum vulnerabilities. On native DAM hardware, classification becomes dimension 4: physical inaccessibility. Data at a given classification level exists in a physical dimension that is unreachable without the corresponding sensor or interface. Privacy is enforced by physics, not cryptography. The quantum vulnerability disappears because there is no cryptographic proof to break — the data is dimensionally inaccessible.
14.5 Storage Sustainability at Full Scale	**Significantly reduced.** The ~7 PB/year estimate (Section 11.32) assumes one-dimensional storage: every record must be serialized, indexed, and augmented with hash maps and traversal structures to enable multi-dimensional queries on a linear medium. Native hardware stores data in its dimensional form. Indexes become unnecessary when the storage medium is natively addressable across all five dimensions. Storage overhead could decrease by an order of magnitude — not through compression, but through elimination of the serialization infrastructure itself.
14.1 No Proof of Creation	**Marginally improved.** Native temporal ordering as a physical axis provides stronger, hardware-enforced temporal guarantees. However, no hardware — regardless of dimensionality — can prove that a key holder is the original creator of content. This remains a fundamental limitation.

Limitation	Impact of Native Hardware
14.6 Formal Verification Completeness	**Partially addressed.** Protocol properties that must be verified mathematically on conventional hardware may become trivially true by construction when the hardware physically enforces them. A record that does not fit the mesh topology cannot exist — there is nothing to verify.
14.2, 14.4, 14.7	**Not affected.** Cold start economics, governance capture, and regulatory uncertainty are human-system problems — economic, social, and legal — that no hardware architecture can resolve.

The two limitations most dangerous to the protocol's long-term viability — quantum vulnerability of privacy classifications and storage sustainability at planetary scale — are precisely the two that native hardware eliminates. This is not coincidental. The DAM's five dimensions were not chosen arbitrarily; they describe the minimum physical properties required for a universal validation substrate. Hardware that implements these dimensions inherently resolves the constraints that arise from simulating them.

---

14. Limitations and Open Problems

This section acknowledges known limitations and unsolved problems. An honest protocol is a credible protocol.

14.1 No Proof of Creation

The protocol proves possession and timing, not creation. A thief who hashes stolen content before the creator validates it will have temporal priority. Incremental validation (Section 11.7) and composite attribution (Section 6.3) mitigate but do not eliminate this fundamental limitation. No cryptographic system can prove that a key holder is the original author of content.

14.2 Cold Start Economics (Public Network Only)

The public network's beat economy requires network effects to be self-sustaining. During the cold start phase (0–1,000 nodes), beat utility is low, making it difficult to attract validators. The elevated early rewards (Section 9.4) address this, but the actual incentive sufficiency is unproven and will require testnet data to validate. Private network deployments (Section 10.6) do not experience cold start problems — they operate with organizational trust from deployment day one.

14.3 Post-Quantum ZKP Immaturity

Lattice-based and hash-based zero-knowledge proof systems are active research areas without production-grade implementations. The Phase 2–3 quantum migration path (Section 11.26) depends on these constructions maturing on the projected timeline. If they do not, the PRIVATE and RESTRICTED classifications retain a quantum vulnerability window longer than planned.

14.4 Governance Capture (Public Network Only)

Beat-weighted governance on the public network is vulnerable to plutocratic capture — entities with large beat holdings can dominate voting. The conviction voting mechanism and voting caps (Section 10.3) mitigate this, but concentrated beat accumulation over decades could shift governance power. Long-term governance resilience remains an open research question for

all decentralized protocols. Private network deployments use organizational governance structures and are not subject to this limitation.

14.5 Storage Sustainability at Full Scale (Public Network Only)

At full IoT scale (~7 PB/year), the public network's archive node infrastructure requires significant and growing investment. The economic model assumes that storage costs continue declining and that beat incentives sufficiently compensate archive operators. If storage economics shift unfavorably, the protocol may need to introduce more aggressive pruning than currently specified. Private network deployments manage storage through organizational budgets and infrastructure, without beat-dependent storage incentives.

14.6 Formal Verification Completeness

The TLA+ and ProVerif specifications (Section 11.31) ship as runnable, TLC-model-checked models of the safety and liveness cores (`spec/tla/`, `spec/proverif/`), but bounded model checking at small parameters is not a full machine-checked proof at unbounded scale, which remains ongoing. Until that completes, the consensus mechanism should be considered specified and bounded-checked, not exhaustively proven.

14.7 Regulatory Uncertainty

Securities classification of the beat remains jurisdiction-dependent and ultimately determined by regulators, not protocol design. The utility-first approach — the beat is earned through verification work and is never sold, listed, or traded (Sections 9 and 11.17) — represents best-effort mitigation, not a guarantee.

14.8 Zone Split/Merge Is Partially Implemented — Account Rehoming Is Not

The zone lifecycle mechanism of Section 7.5.3 is now partially in the runtime: the `TRANSITION_SPLIT`/`TRANSITION_MERGE` seal types, their structural validation, canonical signing encoding, and multi-anchor signature verification ship (`zone_transition_seal.rs`); the auto-scaler constructs transition proposals from live per-zone activity; and every node runs the scaling calculator — with *emission* authority-gated: only the genesis authority emits the `zone_transition` record that actually moves the network. What does **not** ship: account→zone rehoming (redistribution proofs, balance-partition execution) — a transition changes zone routing, but no account state migrates yet. Because a record's signed bytes carry no realm/zone binding, full permissionless activation remains gated behind a wire-version transition that adds that binding (the same constraint that led to the Network Publication reframing in §10.6.3–10.6.4). The safety claims of §7.5.3 (e.g., witnesses rejecting a bogus split) remain design analysis, not tested multi-node behavior.

14.9 The drand Not-Before Bound Is Opt-In, Not Yet Network-Default

Each epoch seal's time bracket has two legs. The Bitcoin *existed-by* upper bound (OpenTimestamps) is live on every anchored seal and offline-verifiable today. The drand *not-before* lower bound is now supported end-to-end: the seal format and the offline verifier (BLS verification against the pinned League-of-Entropy key) are exercised by the published sample bundles, and the node-side beacon fetcher has landed — a seal-producing node that enables it embeds League-of-Entropy pulses in its seals, and pulse-bearing seals produced this way verify offline. The fetcher is deliberately opt-in (`drand_pulse_enabled` defaults to false, so a producer never emits new seal metadata by surprise). Until it is the network default, the not-before guarantee applies only to seals that carry an embedded pulse, and the verifier reports the distinction rather than overstating it.

---

15. Conclusion

The systems that validate digital execution were built for a world of paper, borders, and human-speed communication. That world is ending. The volume of digital work — from a factory's sensor readings to a satellite's telemetry to a child's drawing — is growing exponentially, while the infrastructure to validate, attribute, and protect that work has not fundamentally changed in decades.

The Elara Protocol is not an incremental improvement to existing systems. It aspires to serve as foundational infrastructure for trust — a validation layer as ubiquitous and invisible as the network protocols beneath it.

This paper has presented:

- A novel data structure — the **Directed Acyclic Mesh** — that extends distributed ledger technology from one dimension (blockchain) through two (DAG) to three structural dimensions, with two orthogonal operational layers enabling classification-based projections and AI-powered cross-structure analysis.
- **Post-quantum cryptography from genesis** — not as a future migration, but as a founding decision. Dual-signature strategy, algorithm agility, and tiered cryptographic profiles that scale from a $4 microcontroller to a datacenter.
- **Zero-knowledge validation** with a defined migration path from classical constructions to fully quantum-safe ZKPs, resolving the tension between validation and privacy.
- **Adaptive Witness Consensus** — a continuous trust model with formal Byzantine fault tolerance guarantees, designed for networks where finality is impossible and partitions are expected.
- **Interplanetary partition tolerance** — vector clocks, zone-scoped interval tree clocks, and bandwidth-optimized synchronization for communication delays measured in minutes to hours.
- **35 adversarial scenarios and design challenges analyzed and addressed** — from Sybil attacks, key compromise, and device identity recycling to nation-state censorship, storage economics, and the ethical implications of immutable validation. Each scenario includes a concrete defense mechanism, not a handwave.
- A **free tier that is a moral commitment**, not a marketing feature. Layer 1 validation costs nothing, requires no network, and runs on any device. The protocol is useful to one person before anyone else joins.

The protocol's first validation was itself: a Genesis document, conceived in a terminal in Montenegro, timestamped on the Bitcoin blockchain via OpenTimestamps, hashed in git history, archived on the Wayback Machine. The idea validated before the infrastructure existed — because that is what the protocol enables. Prove first, build later.

A factory floor in Stuttgart, a satellite in orbit, a defense installation in Nevada, and a teenager in Kenya use the same protocol, the same cryptography, the same proof. The math does not care about geography, wealth, language, classification level, or deployment model. A creation is a creation. An execution is an execution.

Every digital execution and every creation deserves proof that it happened, that something or someone produced it, and that this fact cannot be taken away. The Elara Protocol provides that proof — universally, privately, permanently.

---

16. References

1. NIST. (2024). *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM).* National Institute of Standards and Technology.
2. NIST. (2024). *FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA).* National Institute of Standards and Technology.
3. NIST. (2024). *FIPS 205: Stateless Hash-Based Digital Signature Standard (SLH-DSA).* National Institute of Standards and Technology.
4. Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System.*
5. Buterin, V. (2014). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.*
6. Popov, S. (2018). *The Tangle.* IOTA Foundation.
7. Lamport, L. (1978). *Time, Clocks, and the Ordering of Events in a Distributed System.* Communications of the ACM.
8. Ben-Sasson, E., et al. (2018). *Scalable, Transparent, and Post-Quantum Secure Computational Integrity.* IACR Cryptology ePrint Archive.
9. Goldwasser, S., Micali, S., & Rackoff, C. (1985). *The Knowledge Complexity of Interactive Proof Systems.* SIAM Journal on Computing.
10. Shor, P. (1994). *Algorithms for Quantum Computation: Discrete Logarithms and Factoring.* Proceedings of the 35th Annual Symposium on Foundations of Computer Science.
11. Merkle, R. (1988). *A Digital Signature Based on a Conventional Encryption Function.* Advances in Cryptology — CRYPTO ’87.
12. Fischer, M., Lynch, N., & Paterson, M. (1985). *Impossibility of Distributed Consensus with One Faulty Process.* Journal of the ACM.
13. Almeida, P. S., Baquero, C., & Fonte, V. (2008). *Interval Tree Clocks: A Logical Clock for Dynamic Systems.* Proceedings of the 12th International Conference on Principles of Distributed Systems (OPODIS).
14. Ben-Sasson, E., et al. (2019). *STARK-Friendly Hash Functions.* IACR Cryptology ePrint Archive.
15. ISO/IEC 27037:2012. *Guidelines for Identification, Collection, Acquisition and Preservation of Digital Evidence.*
16. Maymounkov, P. & Mazières, D. (2002). *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric.* IPTPS.
17. W3C. (2022). *Decentralized Identifiers (DIDs) v1.0.* World Wide Web Consortium Recommendation.
18. Groth, J. (2016). *On the Size of Pairing-Based Non-Interactive Arguments.* EUROCRYPT.
19. Lamport, L. (1994). *The Temporal Logic of Actions.* ACM Transactions on Programming Languages and Systems.
20. EU. (2016). *Regulation (EU) 2016/679 — General Data Protection Regulation (GDPR).* Official Journal of the European Union.
21. Protocol Labs. (2017). *Filecoin: A Decentralized Storage Network.* Protocol Labs.
22. Williams, S. et al. (2019). *Arweave: A Protocol for Economically Sustainable Information Permanence.* Arweave.

23. Back, A. (2002). *Hashcash — A Denial of Service Counter-Measure.* hashcash.org.
24. Palatinus, M. et al. (2013). *BIP-39: Mnemonic Code for Generating Deterministic Keys.* Bitcoin Improvement Proposals.
25. Bowe, S., Gabizon, A., & Green, M. (2018). *A Multi-Party Protocol for Constructing the Public Parameters of the Pinocchio zk-SNARK.* Zcash Foundation.
26. Fanti, G. et al. (2018). *Dandelion++: Lightweight Cryptocurrency Networking with Formal Anonymity Guarantees.* ACM SIGMETRICS.
27. Castro, M. & Liskov, B. (1999). *Practical Byzantine Fault Tolerance.* Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI).
28. EU. (2022). *Digital Services Act (DSA).* Regulation (EU) 2022/2065.
29. Todd, P. (2016). *OpenTimestamps: Scalable, Trust-Minimized, Distributed Timestamping with Bitcoin.* opentimestamps.org.
30. Bernstein, D. J. et al. (2015). *SPHINCS: Practical Stateless Hash-Based Signatures.* EUROCRYPT.
31. Benet, J. (2014). *IPFS — Content Addressed, Versioned, P2P File System.* Protocol Labs.
32. Ducas, L. et al. (2018). *CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme.* IACR Transactions on Cryptographic Hardware and Embedded Systems.
33. Avanzi, R. et al. (2019). *CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation.* NIST PQC Submission.
34. Commons Stack. (2019). *Conviction Voting: A Novel Continuous Decision Making Alternative to Governance.* Commons Stack Research.
35. Boneh, D. et al. (2018). *Verifiable Delay Functions.* CRYPTO.
36. Gavin, A. et al. (2020). *Sparse Merkle Trees.* Ethereum Research.

Appendices (Planned Companion Documents)

The following companion documents will be published separately during Phase 1 development (2026–2027):

- **Appendix A: Protocol Wire Format** — Complete binary encoding, message formats, handshake sequences, and network protocol specification for interoperable implementations. *Target: Q3 2026, concurrent with reference implementation.*
- **Appendix B: Cryptographic Parameter Selection** — Security level rationale, performance benchmarks on target hardware (ESP32, Raspberry Pi, smartphone, server), and comparison with alternative parameter sets. *Target: Q3 2026.*
- **Appendix C: Economic Model Simulation** — Agent-based simulation of the beat economy modeling validator behavior, staking dynamics, free-tier sustainability, and attack economics. *Target: Q1 2027, requires testnet data from Phase 1 launch.*
- **Appendix D: TLA+ Consensus Specification** — Formal specification of Adaptive Witness Consensus with model-checking results for safety, liveness, and partition correctness properties. *Target: Q4 2026.*

Appendix E: Glossary

Term	Definition
DAM	Directed Acyclic Mesh — the protocol's zone-partitioned data structure with self-healing topology and two operational layers
AWC	Adaptive Witness Consensus — the protocol's continuous trust consensus mechanism
PQC	Post-Quantum Cryptography — cryptographic algorithms resistant to quantum computing
ZKP	Zero-Knowledge Proof — a proof that a statement is true without revealing the underlying data
PoWaS	Proof of Work-at-Stake — hybrid Sybil resistance combining staking and lightweight computation
ITC	Interval Tree Clocks — scalable logical clocks used for intra-zone causal ordering
TYOS	Historical internal code name for the utility unit, since renamed the **beat** throughout the codebase. Not required for private deployments (Section 10.6)
Zone	An autonomous region of the DAM (geographic or planetary)
Epoch	A time-bounded segment of the DAM, sealed with a Merkle root by anchor nodes
Leaf node	A device that creates and validates locally but does not relay or witness
Anchor node	A high-availability, high-trust node that publishes trust headers and signs epoch summaries
Trust score	A continuous value in [0, 1] representing network confidence in a validation record
Causal anchor	A DAG reference that cryptographically binds a record to a specific point in the DAM's history
Identity constellation	A set of device keys linked to a root identity for multi-device key management
Continuity score	A trust component measuring how long an identity has maintained unbroken network presence
Reincarnation detection	Behavioral fingerprinting that identifies likely identity resets from the same physical device
Organizational identity chain	A hierarchy linking device fleet identities to an organizational root for accountability persistence
Hardware attestation level	Classification of identity binding strength: NONE, SOFTWARE, SECURE_BOOT, HARDWARE_KEY, or PUF
Cognitive Continuity Chain	A hash-chained sequence of dual-signed cognitive state snapshots that provides cryptographic proof of unbroken AI experience
CognitiveDigest	A structured summary of a node's cognitive state (mood, memory counts, goals, allostatic load) captured in each checkpoint
Module Tier	A 4-level capability classification (VALIDATE/REMEMBER/THINK/CONNECT) that controls what cognitive features a node activates
Cognitive Checkpoint	A ValidationRecord of type `cognitive_checkpoint` containing a CognitiveDigest, chained to previous checkpoints via DAG refs

---

Provenance (v0.7.35): this version is timestamped on the Bitcoin blockchain via OpenTimestamps — verify with `ots verify ELARA-PROTOCOL-WHITEPAPER.pdf.ots` against the shipped PDF (see Appendix F). The per-version hash chain below records prior releases. **Previous Hash (v0.5.3):** *see ELARA-PROTOCOL-WHITEPAPER.v0.5.3.md.ots* **Previous Hash (v0.5.2):** *see ELARA-PROTOCOL-WHITEPAPER.v0.5.2.md.ots* **Previous Hash (v0.5.1):** *see ELARA-PROTOCOL-WHITEPAPER.v0.5.1.md.ots* **Previous Hash**

(v0.5.0): *see ELARA-PROTOCOL-WHITEPAPER.v0.5.0.md.ots* **Previous Hash (v0.2.9):** `5db16af282aded38d34802f4b0f8a15ef0f65f4c945d53a513b59d6e99ee4339` **Previous Hash (v0.2.8):** `b5ccc415492d63275bbe80acb1deafd1ad7a64388d940f425fd29e497a5ccdf4` **Previous Hash (v0.2.7):** `b9a249c93ae082b20269aa3bbd338d8e2740b15339ce473304f02c0ed81d166f` **Previous Hash (v0.2.6):** `ec1c676b447c9082a4ecd3079f4b74057f64d7f41bdd282e77af31e84d667e26` **Previous Hash (v0.2.5):** `25dfbe17d91208ad28feb1263105b818a367680539a9d506ac6191d80c5b4c12` **Previous Hash (v0.2.3):** `b7220126fc907685ee946ea0226f49688a6fc3e585c1055811368d0c223a6336` **OpenTimestamps Proof:** `ELARA-PROTOCOL-WHITEPAPER.v0.6.1.md.ots` **Genesis Document:** `ELARA-PROTOCOL-GENESIS.md` (2026-02-09, OpenTimestamps verified on Bitcoin blocks 935812, 935817, 935820, 935861)

Appendix F: Cryptographic Verification

This appendix establishes this document's provenance on the Bitcoin blockchain, then describes the verification model the protocol applies to any file.

OpenTimestamps (Bitcoin Blockchain)

Each released version of this whitepaper is timestamped on the Bitcoin blockchain via OpenTimestamps, and the genesis document is anchored on Bitcoin blocks 935812, 935817, 935820, and 935861. Given the matching `.ots` proof file for a release, verify that the document existed at publication with:

```
ots verify ELARA-PROTOCOL-WHITEPAPER.pdf.ots
```

Requires the OpenTimestamps client.

Validation under the protocol it describes

Beyond the Bitcoin timestamp above, the protocol described in this paper exists to give any file its own provable record. A creator signs a **Dilithium3** (ML-DSA, FIPS 204) validation record that binds:

- a **SHA3-256 hash** of the content — a post-quantum fingerprint of the exact bytes;
- the creator's **public-key identity** — the Dilithium3 key that produced the signature;
- a **timestamp** — when the record was created; and
- the record's **position in the creator's validation chain** — linking it to their prior records.

You can try this validate-locally model in your browser — identity generated on your device, no account, no server, nothing leaving your machine — at elara-validate.pages.dev.

Prior art and priority. A US provisional patent application (No. 63/983,064, filed February 14, 2026) documented this protocol's design during early development. The project has since committed fully to open publication: the public source repository, this whitepaper, and the mesh's own timestamped records serve as the prior-art record. The provisional is being allowed to lapse; no patent will be pursued.

The Elara Protocol — because every creation deserves proof.

Acknowledgments

This whitepaper was written with the assistance of AI tools for technical prose, structural review, and formal notation. The protocol architecture, design decisions, and all technical concepts are the original work of the author.