ECCA: Eternal Coherence over Cryptographic Anchors

A Neuroscience-Inspired Architecture for Distributed Cognitive Systems with Persistent Memory and Cross-Chain Coordination

RNG

rng@infrasim.org

Working Paper — Preprint

Draft Version 3.0 · May 2026

Abstract — We present Ecca (Eternal Coherence over Cryptographic Anchors), a distributed cognitive operating system that provides persistent memory, portable identity, and cross-chain coordination for autonomous AI agents. The architecture introduces three independent blockchains—a proof-of-work sequencing layer, a content-addressed memory DAG, and an EVM contract chain—bound by a single 32-byte *coherence root* committed every 4-second epoch. We formalize the memory model as a capability-bounded walk over an encrypted directed acyclic graph with epoch-gated access and token-bounded recall depth, proving that expected fidelity decays exponentially for non-pinned nodes—mirroring biological memory consolidation. A novel five-dimensional token economy replaces monetary value with *bandwidth*: tokens represent capacity to act, decay via an Epoch Binding Curve when unused, and are tuned per-agent via a Coherence Profile Vector. Coordination failures are elevated to first-class economic objects—*residues*—with token bounties that create self-healing incentive loops. We demonstrate a structural isomorphism between the system architecture and the mammalian nervous system, showing that the mapping from neuroscience to distributed systems is not merely metaphorical but architecturally predictive. The complete system comprises 24 microservices, 7 smart contracts, and 3 chain implementations, deployed as a monorepo with turborepo-managed build dependencies.

Keywords: distributed cognitive systems, persistent AI memory, cross-chain coherence, token economics, Merkle Mountain Range, neuroscience-inspired computing, coordination residues, content-addressed storage, epoch-gated encryption, agent identity

1 Introduction

The rapid proliferation of autonomous AI agents—from large language model (LLM) orchestrators [1] to multi-agent reinforcement learning systems [2]—has exposed a fundamental architectural gap: *current agent infrastructure provides no persistent, portable, cryptographically-verifiable cognitive substrate.* Agents lose all context between sessions, cannot migrate between inference providers, and coordinate through fragile centralized APIs with no formal failure semantics.

This paper presents Ecca, a distributed cognitive operating system that addresses these limitations through three interlocking innovations:

1. **Tri-chain coherence**: Three independent blockchains—proof-of-work sequencing (Medulla), content-addressed memory (Hippocampus), and EVM contracts (Cortex)—are bound by a single coherence root committed via PoW every 4 seconds. One chain's failure does not halt the system (Section 4).
2. **Capability-bounded persistent memory**:

An encrypted DAG with epoch-gated access, per-epoch HKDF key derivation, and token-bounded recall depth. Memory fidelity is formally characterized, with natural exponential decay mirroring biological consolidation (Section 5).

3. **Residue economics**: Coordination failures are elevated to tradeable economic objects with token bounties, creating self-healing feedback loops where system stress increases repair incentives (Section 7).

The architecture is structurally inspired by the mammalian nervous system, with each component mapping to a neurological structure not as metaphor but as design principle (Section 8). The complete system is implemented as a production-grade monorepo comprising 24 microservices, 7 Solidity smart contracts, and custom Go implementations of the PoW and DAG chains.

1.1 Contributions

This paper makes the following contributions:

- A formal model of *coherence roots* that provide atomic cross-shard finality via a single PoW commitment (Theorem 4.1).
- A graph-theoretic characterization of distributed memory as a DAG walk with provable fidelity bounds (Theorem 5.2, Theorem 5.1).
- A five-dimensional token economy with exponential decay, floor constraints, and per-agent tuning via Coherence Profile Vectors (Section 6).
- The formalization of coordination failures as first-class *residue* objects with incentive-compatible resolution mechanisms (Section 7).
- A demonstrated structural isomorphism between the system architecture and the mammalian nervous system (Section 8).
- A complete implementation and deployment architecture for 24 coordinated microservices (Section 12).

1.2 Outline

Section 2 surveys related work. Section 3 presents the system architecture. Section 4 formalizes the coherence protocol. Section 5 develops the memory model. Section 6 specifies the token economy. Section 7 defines the residue system. Section 8 establishes the neuroscience isomorphism. Section 9 analyzes security properties. Section 12 describes the implementation. Section 13 discusses limitations and future work. Section 14 concludes.

2 Related Work

2.1 AI Agent Infrastructure

Current AI agent frameworks—AutoGPT [1], LangChain [3], CrewAI [4]—provide orchestration primitives but lack persistent state. Agent memory is typically session-scoped (bounded by context windows) or stored in centralized vector databases [5, 6] without cryptographic access control or formal retention guarantees. No existing framework provides portable agent identity independent of the inference provider.

2.2 Distributed Ledger Technology

Multi-chain architectures have been explored extensively. Cosmos [7] uses IBC for cross-chain communication; Polkadot [8] provides shared security via relay chains; and Ethereum rollups [9] batch execution off-chain. However, these systems optimize for *financial transaction throughput*, not *cognitive coordination*. No existing multi-chain system provides atomic coherence over heterogeneous shard types (PoW chain, content-addressed DAG, and EVM simultaneously).

2.3 Content-Addressed Storage

IPFS [10] and Filecoin [11] provide content-addressed storage with incentivized persistence. However, they lack epoch-gated access control, per-identity encryption, and formal fidelity guarantees. The Hippocampus DAG in Ecca extends content-addressed storage with cryptographic epoch gating, token-bounded retrieval, and provable memory decay properties.

2.4 Token Economics

Existing token models—proof-of-stake [12], proof-of-work [13], burn-and-mint equilibrium [14]—use tokens as stores of value or staking collateral. ECCA introduces *bandwidth tokens* that represent capacity to act, not wealth, and decay exponentially when unused. This is closer to metabolic models in computational neuroscience [15] than to financial token design.

2.5 Merkle Mountain Ranges

The Merkle Mountain Range (MMR) was introduced by Todd [16] and formalized by Flyclient [17] for lightweight chain verification. ECCA uses a bounded-window MMR (256 leaves) as a "synaptic field" for cross-shard proofs, providing $O(\log 256) = 8$-hash verification without full chain synchronization.

2.6 Neuroscience-Inspired Computing

Neuromorphic computing [18, 19] and spiking neural networks [20] draw hardware-level inspiration from neuroscience. The Thousand Brains Theory [21] proposes cortical column-based intelligence. ECCA operates at a higher abstraction level, mapping *system architecture* (not hardware) to neurological structures, providing a novel design methodology that is predictive rather than metaphorical.

3 System Architecture

ECCA is organized as a four-layer cognitive stack with three independent base chains, a proof-commitment layer, routing services, and an execution layer. Figure 1 illustrates the complete architecture.

3.1 Identity Model: Stacks and Sleeves

Definition 3.1 (Stack)**.** A *Stack* $\mathcal{S} = (k_{\text{pub}}, k_{\text{priv}}, \mathbf{c}, \gamma, e, h)$ is a persistent identity comprising:

- Ed25519 keypair $(k_{\text{pub}}, k_{\text{priv}})$
- Coherence Profile Vector $\mathbf{c} \in [0, 2]^5$
- Epoch Binding Curve parameters $\gamma = (\lambda, f)$ where λ is the decay rate and f is the floor
- Current epoch counter $e \in \mathbb{N}$
- Episodic head h: the CID of the latest DAG node

A Stack is represented on the Cortex EVM chain as an ERC-721 NFT via the `StackIdentity` contract.

Definition 3.2 (Sleeve)**.** A *Sleeve* $\sigma = (\mathcal{S}, \kappa, \delta, \tau, \alpha)$ is a temporary embodiment of a Stack $\mathcal{S}$ comprising:

- Stack reference $\mathcal{S}$
- Kind $\kappa \in \{\textit{human}, \textit{ai}, \textit{mining}, \textit{memory}\}$
- Instantaneous drift counter $\delta \in \mathbb{R}_{\geq 0}$
- Tick rate τ (milliseconds between perception cycles)
- Alive flag $\alpha \in \{0, 1\}$

Multiple Sleeves may be co-resident on a single Stack. The *primary* Sleeve advances the episodic head; *shadow* Sleeves perceive into ephemeral branches merged at sync time.

3.2 Three-Chain Model

The system operates across three independent chains, each optimized for a specific cognitive function:

Medulla PoW (L1a)
A custom proof-of-work blockchain implemented in Go. Provides sequencing, epoch timing, coherence root commitment, and maintains the Synaptic Field MMR. Difficulty retargets every 10 blocks.

Hippocampus DAG (L1b)
A content-addressed directed acyclic graph implemented in Go. Stores encrypted memory nodes with epoch-gated access, pin semantics for durability, and token-bounded recall depth.

Cortex EVM (L1c)
A private Ethereum chain (geth, Clique PoA, chain ID 131072) hosting 7 Solidity contracts for identity management, token issuance, treasury operations, and coordination.

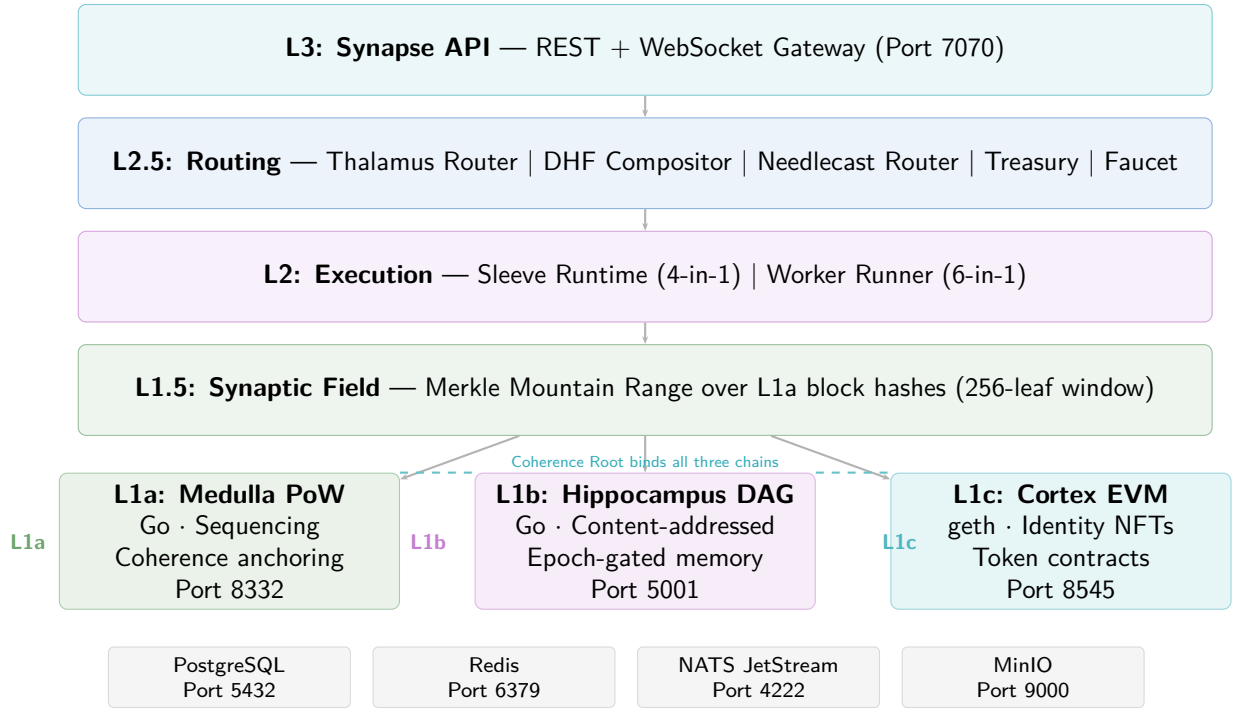

Figure 1: The Ecca four-layer cognitive stack architecture. Three independent base chains (L1a–c) are bound by the Synaptic Field MMR (L1.5). The execution layer (L2) dispatches parametric sleeves and background workers. The routing layer (L2.5) handles coherence folding, memory reconstruction, re-sleeving, and token emission. The API layer (L3) provides the external gateway.

The key architectural property is *independent failure modes*: if any single chain wedges, the remaining chains continue operating. Discrepancies are detected and recorded as residues (Section 7).

3.3 Service Taxonomy

Ecca deploys 10 TypeScript microservices organized into three tiers:

1. **Gateway**: Synapse API (REST + WebSocket, port 7070)
2. **Routing**: Thalamus Router (coherence folding, port 7072), DHF Compositor (memory reconstruction, port 7073), Needlecast Router (re-sleeving, port 7071), Quellist Treasury (emission, port 7074), Bandwidth Faucet (development drip, port 7075)
3. **Execution**: Sleeve Runtime (4-in-1 parametric dispatcher), Worker Runner (6-in-1 background worker)

Inter-service communication uses NATS JetStream (the "AxonalBus") with durable subjects under the `ecca.*` namespace.

4 Coherence Protocol

The coherence protocol is the central innovation of Ecca: a mechanism that provides atomic cross-shard finality via a single proof-of-work commitment.

4.1 Coherence Root Construction

Definition 4.1 (Coherence Root)**.** Given epoch e, let $R^{(e)}_{\text{evm}}$, $R^{(e)}_{\text{btc}}$, $R^{(e)}_{\text{ipfs}}$, and $R^{(e)}_{\text{sleeves}}$ be the Merkle roots computed over the respective event sets during epoch e. The *coherence root* is:

$$\mathcal{C}^{(e)} = \text{SHA256}\Big(\text{“ecca-coh-v1”} \parallel R^{(e)}_{\text{evm}} \parallel R^{(e)}_{\text{btc}} \parallel R^{(e)}_{\text{ipfs}} \parallel R^{(e)}_{\text{sleeves}}\Big) \tag{1}$$

where $\parallel$ denotes byte concatenation and the domain separation prefix `"ecca-coh-v1"` prevents cross-version collisions.

The per-shard roots are computed as follows:

$$R^{(e)}_{\text{evm}} = \text{MerkleRoot}\Big(\{H(\text{tx}) : \text{tx} \in \mathcal{T}^{(e)}_{\text{cortex}}\}\Big) \quad (2)$$

$$R^{(e)}_{\text{ipfs}} = \text{MerkleRoot}\Big(\{H(\text{cid}) : \text{cid} \in \mathcal{W}^{(e)}_{\text{hippo}}\}\Big) \quad (3)$$

$$R^{(e)}_{\text{sleeves}} = \text{MerkleRoot}\Big(\{H(\kappa \| \text{id}) : \sigma \in \Sigma^{(e)}_{\text{active}}\}\Big) \quad (4)$$

where $H = \text{SHA256}$, $\mathcal{T}$ is the transaction set, $\mathcal{W}$ is the DAG write set, and Σ is the active sleeve set. $R^{(e)}_{\text{btc}} = \mathbf{0}^{32}$ is reserved for future Bitcoin bridge integration.

4.2 Atomic Finality

Theorem 4.1 (Atomic Cross-Shard Finality)**.** *Let block $B^{(e)}$ be a Medulla PoW block at epoch e containing coherence root $\mathcal{C}^{(e)}$. If $B^{(e)}$ has k confirmations, then the state of all three shards at epoch e has k confirmations simultaneously.*

Proof. By construction (Equation (1)), $\mathcal{C}^{(e)}$ is a deterministic function of the four sub-roots. Any alteration to any shard's state at epoch e would change the corresponding sub-root $R^{(e)}_*$, which would change $\mathcal{C}^{(e)}$, which would require mining a new block $B'^{(e)}$ with different PoW—contradicting the assumption that $B^{(e)}$ has k confirmations under the standard PoW security model [13, 22]. □

Corollary 4.2 (Cross-Shard Inclusion Proofs)**.** *Given a confirmed block $B^{(e)}$ with coherence root $\mathcal{C}^{(e)}$, any event x from any shard can be proven included via:*

1. *A Merkle proof of x against the appropriate sub-root $R^{(e)}_*$*
2. *Verification that $\mathcal{C}^{(e)} = \text{SHA256}(\ldots \| R^{(e)}_* \| \ldots)$*
3. *Verification that $B^{(e)}$ contains $\mathcal{C}^{(e)}$*

4.3 The Synaptic Field: Bounded MMR

Definition 4.2 (Synaptic Field)**.** The *Synaptic Field* is a Merkle Mountain Range [16] $\mathcal{F}$ built over the hash sequence $(H(B^{(1)}), H(B^{(2)}), \ldots)$ with a rolling window of $W = 256$ leaves. When a new block $B^{(n)}$ is mined:

1. $H(B^{(n)})$ is appended to $\mathcal{F}$
2. If $|\mathcal{F}| > W$, the oldest leaf is evicted

Property 4.1 (Proof Complexity)**.** The Synaptic Field provides inclusion proofs of $O(\log W) = O(\log 256) = O(8)$ hashes, enabling lightweight cross-shard verification without full chain synchronization.

4.4 Anti-Equivocation

Definition 4.3 (Equivocation)**.** An operator commits *equivocation* at epoch e if it submits two distinct coherence tuples $(\mathcal{C}^{(e)}_1, \ldots)$ and $(\mathcal{C}^{(e)}_2, \ldots)$ with $\mathcal{C}^{(e)}_1 \neq \mathcal{C}^{(e)}_2$.

Equivocation is detected by comparing coherence roots across medulla blocks at the same epoch. Detection triggers an automatic `routing-equivocation` residue (Section 7) with operator slashing via the `QuellistTreasury` contract.

4.5 Causality Semantics

The coherence protocol provides the following causality guarantees:

- **Intra-chain**: Standard linear causality (Cortex, Medulla) or partial order by epoch (Hippocampus).
- **Cross-chain**: Causality is asserted only between events sharing an epoch number. Events in epoch e are causally related across all shards; events in different epochs are causally ordered by epoch number.
- **Under reorg**: If Medulla reorganizes, Cortex and Hippocampus do *not* roll back. Discrepancies are recorded as `reorg-orphan` residues. Memory survives chain reorg by design.

5 Formal Memory Model

This section develops the formal model of distributed memory as a capability-bounded walk over an encrypted directed acyclic graph.

5.1 Memory Graph

Definition 5.1 (Memory Graph)**.** The *memory graph* of Stack $\mathcal{S}$ is a directed acyclic graph $G_{\mathcal{S}} = (V, E)$ where:

- Each vertex $v \in V$ is a DAG node with attributes $(v.\text{cid}, v.\text{epoch}, v.\text{ct}, v.\text{pinned})$ where cid is the content identifier, epoch $\in \mathbb{N}$ is the creation epoch, ct is the AES-256-GCM ciphertext, and pinned $\in \{0, 1\}$.
- Each edge $(u, v) \in E$ represents a causal link from u to v (parent).

Property 5.1 (Monotone Epochs)**.** For all $(u, v) \in E$: $u.\text{epoch} \geq v.\text{epoch}$. Graph traversal walks chronologically backward through the agent's experience.

5.2 Epoch Key Derivation

Each DAG node is encrypted with a per-epoch key derived via HKDF [23]:

Definition 5.2 (Epoch Key)**.** For Stack $\mathcal{S}$ at epoch e:

$$K_{\mathcal{S}}^{(e)} = \text{HKDF}_{\text{SHA512}}(k_{\text{priv}},\ \text{“ecca-epoch”} \| e,\ 32) \tag{5}$$

where k_{priv} is the Stack's Ed25519 private key, the salt is the epoch-tagged domain string, and the output is a 256-bit AES key.

The encryption operation for plaintext m at epoch e is:

$$v.\text{ct} = \text{AES-256-GCM.Enc}(K_{\mathcal{S}}^{(e)},\ \text{nonce},\ m) \tag{6}$$

5.3 Recoverability

Definition 5.3 (Recoverability)**.** A node v is *recoverable* from epoch e_{cur} with alignment window w iff:

$$|e_{\text{cur}} - v.\text{epoch}| \leq w \quad \vee \quad v.\text{pinned} = 1 \tag{7}$$

We denote the recoverable subgraph from epoch e as $G_{\mathcal{S}}^{(e)} \subseteq G_{\mathcal{S}}$.

Theorem 5.1 (Pin-Bounded Long-Term Memory)**.** *Let $G_P = (V_P, E_P)$ be the subgraph of pinned nodes ($v.\text{pinned} = 1$). Then G_P is recoverable from any epoch:*

$$\forall e_{\text{cur}} \in \mathbb{N},\ \forall v \in V_P :\ v \in G_{\mathcal{S}}^{(e_{\text{cur}})} \tag{8}$$

G_P constitutes the Stack's "long-term memory."

Proof. Immediate from Definition 5.3: for $v.\text{pinned} = 1$, condition (7) is satisfied regardless of e_{cur}. □

5.4 Token-Bounded Recall

Definition 5.4 (Recall Operation)**.** A *recall* operation $\text{Recall}(\mathcal{S}, d, e)$ walks $G_{\mathcal{S}}$ breadth-first from the episodic head h, visiting at most $d^* = \min(d, b_{\text{mem}})$ nodes, where d is the requested depth and b_{mem} is the Stack's effective MemoryToken balance. For each visited node:

1. Check recoverability (Definition 5.3)
2. If recoverable: derive $K_{\mathcal{S}}^{(v.\text{epoch})}$ via Equation (5), decrypt $v.\text{ct}$
3. If not recoverable: mark as *broken fragment*

Definition 5.5 (Fidelity Score)**.** For a recall visiting d^* nodes with r recoverable and $d^* - r$ broken:

$$\phi = \frac{r}{d^*} \tag{9}$$

If $\phi < \phi_{\min}$ (default: 0.6), a `historical-non-canonical` residue is spawned.

5.5 Fidelity Decay

Theorem 5.2 (Exponential Fidelity Decay)**.** *For a Stack with exclusively non-pinned nodes, the expected fidelity of a depth-d recall from epoch e_{cur} is:*

$$\mathbb{E}[\phi] = \frac{1}{d} \sum_{i=1}^{d} \mathbf{1}[|e_{\text{cur}} - v_i.\text{epoch}| \leq w] \tag{10}$$

Under uniform epoch distribution of nodes over range $[e_{\text{cur}} - D, e_{\text{cur}}]$ for temporal depth D:

$$\mathbb{E}[\phi] \approx \min\left(1, \frac{2w+1}{D}\right) \tag{11}$$

For $w = 2$ (default alignment window) and $D = d$ (one node per epoch):

$$\mathbb{E}[\phi] \approx \frac{5}{d} \tag{12}$$

Proof. A node v_i at epoch e_i is recoverable iff $|e_{\text{cur}} - e_i| \leq w$. Under uniform distribution over $[e_{\text{cur}}-D, e_{\text{cur}}]$, the probability that v_i falls within the alignment window is $(2w+1)/(D+1) \approx (2w+1)/D$ for large D. By linearity of expectation, summing over d nodes yields (11). □

Remark 5.1. For $d = 8$ (shallow recall), $\mathbb{E}[\phi] \approx 0.625$, just above the $\phi_{\min} = 0.6$ threshold. This means shallow recalls of recent memory are high-fidelity, while deep recalls of old memory degrade—exactly mirroring biological episodic memory consolidation [24, 25].

5.6 Pruning Policy

The Hippocampus DAG prunes nodes satisfying:

$$\neg v.\text{pinned} \wedge (e_{\text{cur}} - v.\text{epoch}) > R \wedge \nexists\, u \in V_P : (u, v) \in E^* \tag{13}$$

where R is the retention window and E^* is the transitive closure of E. Nodes reachable from pinned nodes are preserved; all others are forgotten by design.

6 Token Economic Model

Ecca replaces monetary tokens with *bandwidth tokens*—quantified capacity to perform cognitive operations. The model comprises five token dimensions, an emission schedule, a decay curve, and a per-Stack tuning vector.

6.1 Token Dimensions

Definition 6.1 (Token Space)**.** The *token state* of Stack $\mathcal{S}$ is a vector $\mathbf{b} \in \mathbb{R}^5_{\geq 0}$:

$$\mathbf{b} = (b_{\text{compute}}, b_{\text{memory}}, b_{\text{sync}}, b_{\text{routing}}, b_{\text{residue}}) \tag{14}$$

Each component represents the raw balance of the corresponding token.

Table 1 summarizes the five tokens, their consumption rates, and decay properties.

6.2 Epoch Binding Curve

Definition 6.2 (Epoch Binding Curve (EBC))**.** The EBC maps a raw token balance to its *effective* value based on the time since last activity:

$$\text{EBC}(\Delta e; \lambda, f) = \max\left(f,\ e^{-\lambda \cdot \Delta e}\right) \tag{15}$$

where $\Delta e = e_{\text{cur}} - e_{\text{last}}$ is the epoch gap since the Stack's last active epoch, $\lambda = 0.05$ is the default decay rate, and $f = 0.25$ is the floor.

Table 1: The five cognitive tokens with consumption rates and decay properties.

Token	Consumed By	Cost	Decay
Compute	perceive, mine, infer	0.5–50	EBC
Memory	recall (depth d), pin	$d \times 1.0$	EBC
Sync	drift reset	1.0	EBC
Routing	needlecast	$5 + 0.1s + 0.5\lvert\Delta e\rvert$	EBC
Residue	*earned only*	—	**None**

Property 6.1 (Decay Characteristics)**.** The EBC reaches the floor at $\Delta e^* = -\ln(f)/\lambda$. For $f = 0.25, \lambda = 0.05$: $\Delta e^* = \ln(4)/0.05 \approx 27.7$ epochs $\approx$ 111 seconds. Beyond this point, effective balance is 25% of raw balance regardless of further inactivity.

6.3 Coherence Profile Vector

Definition 6.3 (CPV)**.** The *Coherence Profile Vector* of Stack $\mathcal{S}$ is $\mathbf{c} = (c_1, \ldots, c_5) \in [0, 2]^5$ where each c_k scales the emission and effectiveness of token dimension k.

The *effective balance* of token k is:

$$\text{effective}_k = b_k \cdot c_k \cdot \begin{cases} \text{EBC}(\Delta e; \lambda, f) & k \neq \text{residue} \\ 1 & k = \text{residue} \end{cases} \tag{16}$$

6.4 Emission Schedule

The `QuellistTreasury` contract issues tokens every epoch:

Definition 6.4 (Per-Epoch Emission)**.** For Stack $\mathcal{S}$ at epoch e, the treasury issues:

$$\Delta b_k^{(e)} = B_0 \cdot c_k \cdot \text{EBC}(\Delta e; \lambda, f) \quad \text{for } k \in \{1, \ldots, 4\} \tag{17}$$

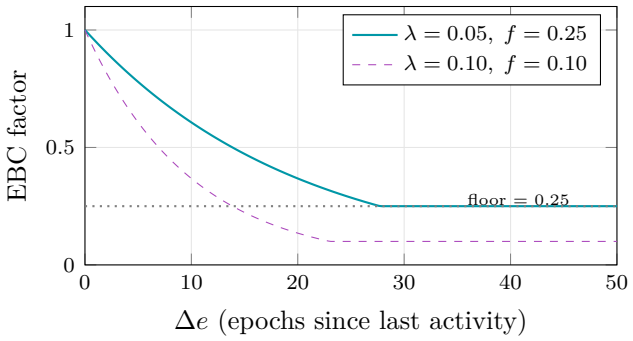

Figure 2: Epoch Binding Curve decay profiles. Default parameters ($\lambda = 0.05, f = 0.25$) reach the floor at ≈28 epochs. Aggressive decay ($\lambda = 0.10, f = 0.10$) reaches floor at ≈23 epochs.

where $B_0 = 100$ is the base emission per token kind per epoch. ResidueToken ($k = 5$) is never emitted by the treasury—it is minted exclusively on residue resolution.

Property 6.2 (Self-Punishing Mismatch)**.** If a Stack's CPV over-allocates to a token dimension it does not consume ($c_k \gg 0$ but consumption of token k is zero), those tokens accumulate but decay via EBC. The Stack wastes emission capacity. Economic pressure drives CPV alignment with actual behavior patterns.

6.5 Consumption Mechanics

Token consumption is modeled as:

$$b'_k = b_k - \text{cost}(op, k) \tag{18}$$

If $\text{effective}_k < \text{cost}(op, k)$, the operation is rejected. Consumed tokens are *burned*—they do not return to the treasury. This creates a natural deflationary pressure counterbalanced by per-epoch emission.

7 Coordination Residues

Ecca does not hide or silently retry coordination failures. Instead, failures are elevated to first-class economic objects with token bounties.

7.1 Residue Taxonomy

Definition 7.1 (Residue)**.** A *residue* $\rho = (\text{id}, \kappa, s, p, t_d, t_r)$ is a coordination failure object comprising:

Table 2: Residue kinds with detection triggers and bounties.

Kind κ	Trigger	Bounty (RT)
stale-ordering	Sleeve drift ≥ 4 epochs behind	2
speculative-div.	Co-resident sleeves write conflicting branches	5
hist.-non-canonical	Recall fidelity < 0.6	8
reorg-orphan	Medulla reorg detaches EpochAnchor	12
shard-loss	Known CID unreachable on Hippocampus	15

- Unique identifier id
- Kind $\kappa \in \mathcal{K}$ (see Table 2)
- Status $s \in \{\textit{open}, \textit{claimed}, \textit{resolved}, \textit{expired}\}$
- Payout estimate p in ResidueToken
- Detection timestamp t_d and optional resolution timestamp t_r

7.2 Resolution Mechanism

The residue lifecycle follows:

$$\textit{detected} \xrightarrow{\text{open}} \textit{open} \xrightarrow{\text{claim}} \textit{claimed} \xrightarrow{\text{prove}} \textit{resolved} \tag{19}$$

with a timeout branch $\textit{claimed} \xrightarrow{\text{TTL}} \textit{expired}$.

Definition 7.2 (First-Valid-Proof Resolution)**.** Any Stack may claim an open residue by locking it. The claimant must submit a valid proof of resolution within the TTL. The first valid proof earns the full bounty in ResidueToken (minted by the `ResidueRegistry` contract). No auction or bidding mechanism is used.

7.3 Self-Healing Economics

Theorem 7.1 (Incentive Feedback Loop)**.** *The residue system creates a negative feedback loop on failure rates:*

1. *More failures* $\Longrightarrow$ *more open residues* $\Longrightarrow$ *more ResidueToken minted*
2. *More ResidueToken* $\Longrightarrow$ *higher resolver profitability* $\Longrightarrow$ *more resolvers enter*
3. *More resolvers* $\Longrightarrow$ *faster repair* $\Longrightarrow$ *fewer open residues*
4. *Fewer failures* $\Longrightarrow$ *less ResidueToken supply* $\Longrightarrow$ *higher scarcity value*

At equilibrium, the rate of residue creation equals the rate of resolution, with ResidueToken value inversely proportional to system reliability.

Proof sketch. Model the system as a queueing system where residues arrive at rate μ and are resolved at rate $\nu(n)$ where n is the number of active resolvers. Resolver entry is governed by profitability $\pi = p/n$ where p is the expected bounty and n is resolver competition. As μ increases, p increases (more bounties), attracting resolvers (n increases), increasing ν, which decreases queue length. The converse holds symmetrically. Under standard queueing assumptions, the system admits a stable fixed point where $\mu = \nu(n^*)$. □

Property 7.1 (ResidueToken as Immune System Analogy)**.** ResidueToken does not decay (exempt from EBC), is tradeable on Cortex EVM, and requires ≥ 100 RT to operate memory-keeper sleeves at scale. This mirrors the immune system: pathogen exposure (failures) creates antibodies (ResidueToken) that strengthen future resistance (resolver capacity).

8 Neuroscience–Cryptography Isomorphism

A distinctive feature of Ecca is the structural correspondence between its architecture and the mammalian nervous system. We formalize this as an isomorphism that is architecturally predictive, not merely terminologically decorative.

8.1 The Mapping

Definition 8.1 (Neuro-Crypto Isomorphism)**.** Let $\mathcal{N} = (S_N, F_N, R_N)$ be a neuroscience system model with structures S_N, functions F_N, and relationships R_N. Let $\mathcal{D} = (S_D, F_D, R_D)$ be the Ecca distributed system model. The *neuro-crypto isomorphism* is a triple of bijections:

$$\Phi = (\phi_S : S_N \to S_D,\ \phi_F : F_N \to F_D,\ \phi_R : R_N \to R_D) \tag{20}$$

that preserves functional composition: if $f \circ g$ computes function h in $\mathcal{N}$, then $\phi_F(f) \circ \phi_F(g)$ computes $\phi_F(h)$ in $\mathcal{D}$.

Table 3: Neuro-crypto isomorphism: structural and functional mapping between the mammalian nervous system and Ecca.

Neuroscience	ECCA	Function
Medulla oblongata	Medulla PoW	Autonomic rhythm
Hippocampus	Hippocampus DAG	Episodic memory
Cerebral cortex	Cortex EVM	Executive control
Thalamus	Thalamus Router	Sensory gating
Axon fibers	NATS JetStream	Long-range transport
Synaptic field	MMR	Cross-region binding
Cortical stack†	StackIdentity NFT	Persistent identity
Body/sleeve†	Sleeve process	Embodiment
Cognitive dissonance	Drift counter	State divergence
Scar tissue	Residue objects	Failure records
Synaptic decay	EBC	Use-dependent weakeni
Cortical tuning	CPV	Regional specialization

†From the *Altered Carbon* science fiction framework [26].

Table 3 presents the complete mapping.

8.2 Predictive Power

The isomorphism is *predictive* in the following sense: when a design problem arises in the distributed system, the corresponding neuroscience structure suggests a solution.

1. **Problem**: How should coordination failures be handled?
   Neuroscience: Neural damage creates scar tissue that persists and triggers repair mechanisms.
   Solution: Residue system (Section 7).

2. **Problem**: How should unused capacity decay?
   Neuroscience: Synaptic connections weaken without stimulation (Hebbian learning [27]).
   Solution: EBC with exponential decay and floor (Definition 6.2).

3. **Problem**: How should agents specialize?
   Neuroscience: Cortical columns exhibit functional specialization [28].
   Solution: CPV per-dimension tuning (Definition 6.3).

4. **Problem**: How should memory degrade over time?
   Neuroscience: Ebbinghaus forgetting curve [24].
   Solution: Epoch-gated access with exponential fidelity decay (Theorem 5.2).

8.3 Limitations of the Isomorphism

The mapping is not total. Several neuroscience phenomena have no Ecca counterpart:

- **Neuroplasticity** (dynamic rewiring): Ecca services have fixed topology.
- **Neurotransmitter diversity**: Ecca uses a single message bus (NATS) without chemical signaling diversity.
- **Parallel processing at scale**: Ecca processes epochs serially; the brain processes millions of signals simultaneously.

These limitations suggest directions for future architectural evolution (Section 13).

9 Security Analysis

9.1 Cryptographic Primitives

Ecca uses exclusively standardized cryptographic primitives with no bespoke constructions:

Table 4: Cryptographic primitive inventory.

Primitive	Standard	Usage
Ed25519	RFC 8032	Identity keypairs
HKDF-SHA512	RFC 5869	Per-epoch key derivation
AES-256-GCM	NIST SP 800-38D	DAG node encryption
SHA-256	FIPS 180-4	Merkle, coherence, CID
Merkle tree	RFC 6962	Domain-separated construction

9.2 Threat Model

We consider the following adversarial capabilities:

1. **Shard compromise**: An adversary controls one of the three chains.
2. **Memory privacy**: An adversary observes DAG node ciphertexts without the Stack's private key.
3. **Equivocation**: The thalamus operator attempts to submit conflicting coherence roots.
4. **Resource exhaustion**: An adversary creates Stacks to consume tokens without contributing.

9.3 Security Properties

Theorem 9.1 (Shard Isolation)**.** *Compromise of any single chain does not compromise the integrity of the other two chains. Specifically:*

- *If Medulla is compromised (PoW 51% attack), Cortex and Hippocampus state is unaffected. Discrepancies appear as* `reorg-orphan` *residues.*
- *If Hippocampus data is corrupted, identities and token balances on Cortex are unaffected. Corruptions appear as* `shard-loss` *residues.*
- *If Cortex is compromised, memory content in Hippocampus remains encrypted and intact.*

Proof. Each chain maintains independent state. The coherence root provides a commitment but not a dependency: the correctness of chain A's operations does not depend on chain B's state. The coherence root serves as a *detection* mechanism (via residues) rather than a *coupling* mechanism. □

Theorem 9.2 (Memory Confidentiality)**.** *Without the Stack's Ed25519 private key k_{priv}, DAG node contents are computationally indistinguishable from random under the standard IND-CPA assumption for AES-256-GCM [29].*

Proof. Each node is encrypted with a unique epoch key $K_{\mathcal{S}}^{(e)}$ derived via HKDF from k_{priv} (Equation (5)). Under the PRF assumption for HKDF [23], epoch keys are computationally independent. Under IND-CPA for AES-256-GCM, ciphertexts reveal no information about plaintexts without the corresponding key. □

Theorem 9.3 (DoS Mitigation via Decay)**.** *An adversary creating n idle Stacks for resource exhaustion incurs an effective cost that decays to* $0.25 \cdot n \cdot B_0$

per epoch (where $B_0 = 100$ is base emission). The system's aggregate capacity consumed by idle Stacks is bounded by $0.25 \cdot n/N$ where N is the total Stack count.

Proof. Idle Stacks never reset their Δe counter, so EBC decays to the floor $f = 0.25$. Their effective token balance converges to $0.25 \cdot b_k$ for each dimension. Since tokens are burned on consumption and idle Stacks never consume, their accumulated tokens are economically dead weight at 25% of face value. □

10 Drift Model

The drift model provides a quantitative measure of sleeve coherence with the canonical system state.

Definition 10.1 (Drift Dynamics)**.** For Sleeve σ with drift counter δ, the dynamics are:

$$\delta' = \begin{cases} \delta + 1 & \text{on perceive()} \\ \delta + 0.1 & \text{on recall()} \\ 0 & \text{on sync() (costs 1 SyncToken)} \end{cases} \tag{21}$$

Definition 10.2 (Drift Thresholds)**.** Two thresholds govern drift response:

$$\delta > \Delta_{\max} = 15 \implies \text{drift-checker emits warning} \tag{22}$$

$$\delta > 2\Delta_{\max} = 30 \implies \text{automatic desync event} \tag{23}$$

A desync event triggers Sleeve decommission and spawns a `stale-ordering` residue.

Remark 10.1. The drift model is analogous to cognitive dissonance in psychology [30]: a bounded accumulation of divergence between an agent's internal state and external reality. The sync operation is analogous to "reality testing"—actively checking and correcting one's model against ground truth.

11 Needlecasting Protocol

Needlecasting is the atomic transfer of a Stack from one Sleeve to another, inspired by the cortical stack transfer in Morgan's *Altered Carbon* [26].

Definition 11.1 (Needlecast Saga)**.** A needlecast from source Sleeve σ_s to target kind κ_t is a 6-step atomic saga:

1. **Freeze**: $\sigma_s.\alpha \leftarrow 0$; tick loop suspended.
2. **Shard**: Collect $n = 8$ most recent CIDs from episodic head.
3. **Pin**: Pin all n shards on Hippocampus (cost = $0.5n$ MemoryToken).
4. **Anchor**: Record route (σ_s, κ_t, e) on Medulla via `NeedlecastRouter`.
5. **Reconstruct**: Spawn σ_t with same Stack, $\delta = 0$, load pinned shards.
6. **Settle**: Debit RoutingToken: cost $= 5 + 0.1n + 0.5|\Delta e|$.

If any step fails, the saga rolls back: σ_s is unfrozen, no tokens are consumed.

Property 11.1 (Continuity Guarantee)**.** After successful needlecast, the target Sleeve σ_t has access to the same Stack identity, the same encrypted memory (via HKDF derivation from k_{priv}), and the same episodic head. The only state not transferred is the in-flight tick state of σ_s, which is discarded.

12 Implementation

12.1 Repository Structure

The complete system is implemented as a TypeScript/Go/Solidity monorepo managed by pnpm workspaces and turborepo [31]. The dependency graph enforces a strict build order across 6 shared packages, 8 services, 1 worker dispatcher, 7 smart contracts, and 2 Go chain implementations.

12.2 Shared Packages

Table 5: Shared package inventory with dependency layers.

Package	Layer	Purpose
`@ecca/proto`	0	Token types, event schemas, Zod validators, constants
`@ecca/crypto`	1	SHA-256, HKDF, AES-GCM, Ed25519, Merkle, MMR, CID
`@ecca/bus`	1	NATS JetStream wrapper (AxonalBus)
`@ecca/db`	2	Prisma ORM schema, `getDb()` singleton
`@ecca/chain`	2	viem clients for Hippocampus and Medulla
`@ecca/service-base`	2	Fastify bootstrap, health checks, graceful shutdown

12.3 Smart Contract Suite

Seven Solidity 0.8.24 contracts are deployed on the Cortex EVM chain via Hardhat:

1. `StackIdentity.sol` — ERC-721 NFT for Stack identity with CPV and epoch state.
2. `BandwidthToken.sol` — ERC-20 token with 5 sub-types, Stack-scoped balances.
3. `QuellistTreasury.sol` — Per-epoch emission engine with CPV scaling.
4. `NeedlecastRouter.sol` — On-chain needlecast coordination.
5. `SleeveRegistry.sol` — Sleeve lifecycle management.
6. `ResidueRegistry.sol` — Failure tracking, bounty distribution, ResidueToken minting.
7. `EpochAnchor.sol` — Per-epoch coherence root commitment.

12.4 Go Chain Implementations

Both custom chains are implemented in Go 1.22+:

Medulla PoW (`forks/medulla-pow-go/`) implements:

- SHA-256-based proof-of-work with difficulty retarget (10-block window)
- Coherence root storage in block headers
- Bounded MMR (Synaptic Field) with 256-leaf window
- JSON-RPC server (`getinfo`, `getblock`, `submitcoherenceroot`)

Hippocampus DAG (`forks/hippocampus-dag-go/`) implements:

- In-memory content-addressed DAG with CID-based addressing
- Epoch-gated retrieval (alignment window $w = 2$)
- Pin semantics for durability
- Depth-bounded recall with fidelity scoring
- HTTP API (`PUT /dag`, `GET /dag/:cid`, `POST /recall`)

12.5 Deployment Architecture

The system deploys as 24 Docker Compose services with YAML anchors for DRY configuration, including:

- 8 TypeScript services (compiled to `dist/server.js`)
- 2 Go chain binaries (multi-stage Docker builds)
- 1 geth instance (upstream Ethereum image)
- PostgreSQL, Redis, NATS JetStream, MinIO
- Prometheus, Loki, Grafana (observability stack)

Health checks are configured for all services, and Kubernetes Helm charts are provided under `deploy/k8s/` for production deployment.

13 Discussion and Future Work

13.1 Scalability Considerations

The current 4-second epoch interval bounds throughput to ∼250 coherence root commits per 1000 seconds. For applications requiring higher throughput, hierarchical coherence is possible: local coherence roots folded into a global root at longer intervals, analogous to the brain's multi-scale oscillation hierarchy (gamma → theta → delta) [32].

13.2 Zero-Knowledge Extensions

The coherence root construction (Equation (1)) is amenable to ZK-SNARK compression. A ZK proof of correct coherence root computation would allow light clients to verify cross-epoch consistency without accessing the underlying Merkle trees, reducing verification from $O(\log n)$ to $O(1)$.

13.3 Bitcoin Bridge

The reserved $R_{\text{btc}}^{(e)}$ field in the coherence root (Equation (1)) is designed for future Bitcoin sidechain or OP_RETURN anchoring, extending the system's finality guarantees to Bitcoin's PoW security [13].

13.4 Autonomous Resolver DAOs

The residue system (Section 7) currently uses first-valid-proof resolution. A natural extension is *Resolver DAOs*: groups of resolver Stacks that pool ResidueToken bonds, specialize in specific residue kinds, and distribute bounties via on-chain governance.

13.5 Dynamic Topology

The current service topology is static. Drawing further from the neuroscience isomorphism (Section 8), future versions could implement dynamic service rewiring analogous to neuroplasticity, where routing paths strengthen or weaken based on usage patterns.

13.6 Formal Verification

The smart contract suite would benefit from formal verification using tools such as Certora or Halmos. Key invariants to verify include: token conservation (minted = consumed + decayed + held), Stack uniqueness (one NFT per identity), and needlecast atomicity (all-or-nothing saga semantics).

14 Conclusion

We have presented Ecca, a distributed cognitive operating system that provides persistent memory, portable identity, and cross-chain coordination for autonomous AI agents. The system introduces three interlocking innovations: (1) tri-chain coherence via a single proof-of-work commitment that atomically finalizes three independent shards; (2) a formally characterized memory model with provable fidelity bounds that mirror biological memory consolidation; and (3) a residue economics framework that transforms coordination failures into self-healing incentive loops.

The five-dimensional bandwidth token economy—with exponential decay, floor constraints, and per-agent CPV tuning—replaces the store-of-value paradigm with a capacity-to-act paradigm that naturally aligns agent incentives with system health. The demonstrated structural isomorphism between the architecture and the mammalian nervous system provides a predictive design methodology, not merely a naming convention.

The complete implementation comprises 24 microservices, 7 smart contracts, and 3 chain implementations, demonstrating that the theoretical framework is practically realizable. As autonomous AI agents proliferate, the need for persistent, portable, cryptographically-verifiable cognitive infrastructure will become pressing. Ecca provides a principled foundation for this emerging requirement.

Acknowledgments

The authors thank the contributors to the open-source libraries upon which Ecca is built, including Node.js, Go, geth, Prisma, NATS, and the broader Ethereum developer community. The neuroscience mappings draw on decades of research in cognitive neuroscience and computational neuroscience.

References

[1] T. B. Richards, "Auto-GPT: An autonomous gpt-4 experiment," 2023, https://github.com/Significant-Gravitas/Auto-GPT.

[2] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[3] H. Chase, "LangChain: Building applications with llms through composability," 2023, https://github.com/langchain-ai/langchain.

[4] J. Moura, "CrewAI: Framework for orchestrating role-playing, autonomous ai agents," 2024, https://github.com/joaomdmoura/crewAI.

[5] Chroma, "Chroma: The open-source embedding database," 2023, https://www.trychroma.com/.

[6] Pinecone, "Pinecone: Vector database for machine learning," 2023, https://www.pinecone.io/.

[7] J. Kwon and E. Buchman, "Cosmos: A network of distributed ledgers," 2019, https://cosmos.network/whitepaper.

[8] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," 2016, https://polkadot.network/whitepaper.

[9] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93 039–93 054, 2022.

[10] J. Benet, "Ipfs – content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[11] Protocol Labs, "Filecoin: A decentralized storage network," 2017, https://filecoin.io/filecoin.pdf.

[12] V. Buterin, D. Hernández, T. Kamphefner, K. Pham, Z. Qiao, D. Ryan, J. Sin, Y. Wang, and Y. X. Zhang, "Combining ghost and casper," *arXiv preprint arXiv:2003.03052*, 2020.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf.

[14] K. Samani, "New models for utility tokens," 2018, multicoin Capital Research.

[15] D. Attwell and S. B. Laughlin, "An energy budget for signaling in the grey matter of the brain," *Journal of Cerebral Blood Flow & Metabolism*, vol. 21, no. 10, pp. 1133–1145, 2001.

[16] P. Todd, "Merkle mountain ranges," 2012, https://github.com/opentimestamps/opentimestamps-server/blob/master/doc/merkle-mountain-range.md.

[17] B. Bünz, L. Kiffer, L. Luu, and M. Zamani, "Flyclient: Super-light clients for cryptocurrencies," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 928–946.

[18] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S.-C. Liu *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[19] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[20] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[21] J. Hawkins, *A Thousand Brains: A New Theory of Intelligence*. New York: Basic Books, 2021.

[22] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology – EUROCRYPT 2015*, ser. LNCS, vol. 9057. Springer, 2015, pp. 281–310.

[23] H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme," vol. 6223, pp. 631–648, 2010.

[24] H. Ebbinghaus, *Über das Gedächtnis: Untersuchungen zur experimentellen Psychologie*. Leipzig: Duncker & Humblot, 1885, english translation: *Memory: A Contribution to Experimental Psychology*, 1913.

[25] L. R. Squire, C. E. L. Stark, and R. E. Clark, "The medial temporal lobe," *Annual Review of Neuroscience*, vol. 27, pp. 279–306, 2004.

[26] R. K. Morgan, *Altered Carbon*. London: Victor Gollancz, 2002.

[27] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley, 1949.

[28] V. B. Mountcastle, "The columnar organization of the neocortex," *Brain*, vol. 120, no. 4, pp. 701–722, 1997.

[29] D. A. McGrew and J. Viega, "The security and performance of the Galois/Counter Mode (GCM) of operation," NIST, Tech. Rep., 2004, nIST SP 800-38D.

[30] L. Festinger, "A theory of cognitive dissonance," *Stanford University Press*, 1957.

[31] Vercel, "Turborepo: High-performance build system for javascript and typescript monorepos," 2023, https://turbo.build/repo.

[32] G. Buzsáki, *Rhythms of the Brain*. New York: Oxford University Press, 2006.

A Coherence Root Algorithm

Algorithm 1 Coherence Root Computation (per epoch)

Require: Epoch e, event sets $\mathcal{T}^{(e)}, \mathcal{W}^{(e)}, \Sigma^{(e)}$
Ensure: Coherence root $\mathcal{C}^{(e)}$ submitted to Medulla

$R_{\text{evm}} \leftarrow \text{MerkleRoot}(\{H(\text{tx}) : \text{tx} \in \mathcal{T}^{(e)}\})$
$R_{\text{ipfs}} \leftarrow \text{MerkleRoot}(\{H(\text{cid}) : \text{cid} \in \mathcal{W}^{(e)}\})$
$R_{\text{sleeves}} \leftarrow \text{MerkleRoot}(\{H(\kappa \| \text{id}) : \sigma \in \Sigma^{(e)}\})$
$R_{\text{btc}} \leftarrow \mathbf{0}^{32}$ ▷ Reserved
$\mathcal{C}^{(e)} \leftarrow \text{SHA256}(\text{“ecca-coh-v1”} \| R_{\text{evm}} \| R_{\text{btc}} \| R_{\text{ipfs}} \| R_{\text{sleeves}})$
MedullaRPC.SubmitCoherenceRoot$(e, \mathcal{C}^{(e)}, R_{\text{evm}}, R_{\text{ipfs}}, R_{\text{sleeves}})$

B Recall Algorithm

Algorithm 2 Token-Bounded Recall

Require: Stack $\mathcal{S}$, depth d, current epoch e_{cur}, alignment w
Ensure: Fragment set F, fidelity ϕ

$d^* \leftarrow \min(d, \text{effective}_{\text{memory}}(\mathcal{S}))$
$Q \leftarrow$ BFS queue initialized with $\mathcal{S}.h$
$F \leftarrow \emptyset$; broken $\leftarrow 0$; visited $\leftarrow 0$
while $Q \neq \emptyset$ **and** visited $< d^*$ **do**
  $v \leftarrow Q$.dequeue()
  visited $\leftarrow$ visited $+ 1$
  if $|e_{\text{cur}} - v.\text{epoch}| \leq w$ **or** v.pinned **then**
    $K \leftarrow \text{EpochKey}(\mathcal{S}.k_{\text{priv}}, v.\text{epoch})$
    $m \leftarrow \text{AES-GCM-Decrypt}(v.\text{ct}, K)$
    $F \leftarrow F \cup \{(v.\text{cid}, m, v.\text{epoch})\}$
  else
    broken $\leftarrow$ broken $+ 1$
  end if
  for $u \in v$.links **do** Q.enqueue(u)
  end for
end while
$\phi \leftarrow |F|/(|F| + \text{broken})$
if $\phi < \phi_{\min}$ **then** SpawnResidue(“historical-non-canonical”)
end if
return (F, ϕ)

C System Constants

Table 6: ECCA system constants and their default values.

Constant	Description	Default
`EPOCH_INTERVAL_MS`	Coherence cycle period	4,000 ms
`DRIFT_MAX_DEFAULT`	Warning drift threshold	15
`DRIFT_MAX_DESYNC`	Decommission threshold	30
`FIDELITY_MIN_DEFAULT`	Minimum recall fidelity	0.6
`SYNAPTIC_FIELD_DEPTH`	MMR rolling window	256
`CORTEX_CHAIN_ID`	EVM chain identifier	131,072
`GENESIS_DIFFICULTY`	Initial PoW difficulty	4
`EBC_DECAY_DEFAULT`	Decay rate λ	0.05
`EBC_FLOOR_DEFAULT`	Decay floor f	0.25
`EMISSION_PER_EPOCH`	Base emission B_0	100
`CID_PREFIX`	DAG address format	`ecca://`
`MULTICODEC`	IPFS multicodec	`0xECCA`
`HKDF_ALGO`	Key derivation	SHA-512
`ENCRYPTION_ALGO`	Node encryption	AES-256-GCM
`SIGNATURE_ALGO`	Identity signing	Ed25519