Disentangle: Topological Mass Consensus with Capability-Coherence Identity for Sybil-Resistant Agreement via Discrete Curvature

Larsen James Close, Independent Researcher

February 2026

Abstract

We present Topological Mass Consensus (TMC), a novel consensus mechanism that derives security from the geometric structure of transaction graphs rather than economic incentives or computational work. TMC leverages discrete curvature—specifically, a Jaccard-based approximation to Ollivier-Ricci curvature following Pal et al. (2017)—to detect and throttle Sybil attacks. Edges connecting attack clusters to the honest network exhibit characteristic negative curvature due to low neighborhood overlap, enabling throttling that reduces attacker influence by a factor of $1/\varepsilon$ (default $\varepsilon = 0.01$, yielding ~99% reduction in simulations with specific graph parameters; see Section 6 for parameter sensitivity). We provide a complete post-quantum implementation using NIST-standardized primitives: ML-DSA (FIPS 204) for signatures, ML-KEM (FIPS 203) for key encapsulation, and SHA3-256 for hashing, along with zero-knowledge reputation proofs via Plonky3 STARKs. Our implementation comprises 9 Rust crates with 349+ passing tests. All temporal properties—ordering, epochs, confirmation, and bootstrap activation—are derived from topological depth (a Lamport clock on the DAG partial order), eliminating assigned timestamps and external clock dependencies. We demonstrate that topological mass—a curvature-weighted measure of structural integration—provides deterministic conflict resolution resistant to content grinding, though parent-selection constraints are required to bound topology grinding. We further present the Capability-Coherence Identity Protocol (CCIP), an identity and capability layer where DIDs (including non-human agents via did:agi:*) participate in the same substrate, authority is held as object capabilities with delegation chains subject to curvature analysis, and the system's native value measure is topological mass itself—a non-transferable, non-extractable structural property of coherent participation. Introduction chains, delegation chains, and governance voting are all transactions in the DAG, enabling unified curvature-based detection of Sybil identity farms, capability laundering, manufactured trust networks, and governance capture without external oracles or authorities.

Contents

1 Introduction

Distributed consensus remains one of the fundamental challenges in computer science. Existing solutions fall into two broad categories: Proof of Work (PoW), which derives security from thermodynamic costs, and Proof of Stake (PoS), which derives security from game-theoretic incentives. Both approaches face significant limitations:

- **PoW** consumes enormous energy and remains vulnerable to 51% attacks by well-funded adversaries
- **PoS** suffers from nothing-at-stake problems, long-range attacks, and plutocratic centralization
- **Both** rely on cryptographic primitives (ECDSA, Ed25519) vulnerable to Shor's algorithm on quantum computers

We propose a third approach: **Topological Mass Consensus (TMC)**, which derives security from information theory and discrete geometry. Rather than asking "who has the authority to extend the chain," TMC asks "which state is structurally coherent with the network's history."

We call this protocol **Disentangle**: it *disentangles* coherent network structure from incoherent Sybil clusters by detecting the geometric signature of fake identities—low-overlap boundary edges that manifest as negative curvature.

1.1 Key Insight

Sybil attacks—where an adversary creates many fake identities to gain disproportionate influence—create detectable geometric distortions in the transaction graph. Specifically, Sybil clusters connect to the honest network through *bottleneck edges* with low neighborhood overlap, which manifests as negative curvature under the Jaccard-based approximation to Ollivier-Ricci curvature. By throttling influence based on local curvature, TMC achieves Sybil resistance without requiring proof of resource expenditure.

This geometric principle extends beyond consensus. We show that the same curvature analysis detects incoherent behavior across identity (manufactured trust networks exhibit negative curvature in the introduction subgraph), capability (misused delegations create bridge topology between delegator and delegatee), and governance (plutocratic capture requires structural isolation that curvature measures). The unifying insight is that coherence—structural integration via closed consequence chains—is what curvature measures, and Sybil resistance is one instance of a general coherence detection mechanism.

1.2 Curvature as Coherence Measurement

The use of curvature for Sybil detection has a deeper information-theoretic interpretation: **curvature measures structural coherence**.

- **High curvature** ($\kappa > 0$): Dense neighborhood overlap indicates nodes participate in closed consequence loops—their transactions are genuinely integrated with surrounding structure. This is coherent behavior.
- **Low/negative curvature** ($\kappa < 0$): Sparse neighborhood overlap indicates structural isolation—nodes are attached to the network but not woven into its fabric. Sybil clusters are, by construction, "consequence-free zones" that lack genuine relational integration.

This framing reveals why TMC detects *the act*, not *the identity*:

1. **Historical reputation provides no immunity**: A node with 1000 honest transactions creates detectable topological anomalies the moment it attempts a double-spend or fork attack. The act of betrayal creates geometric distortion regardless of accumulated history.

2. **Integration attacks are expensive**: An attacker who wants to avoid detection must achieve genuine structural integration—making many transactions that other honest nodes naturally reference. This converts a Sybil attack into a long-term investment with comparable economics to legitimate participation.
3. **Detection is local**: Curvature computation requires only the immediate neighborhood, not global graph analysis. Anomalies are detectable where they occur, enabling real-time throttling.

1.3 Contributions

This paper makes the following contributions:

1. **Topological Mass Consensus**: A conflict resolution mechanism based on discrete curvature that provides deterministic finality without leader election or voting rounds.
2. **Curvature-Based Sybil Detection**: We show that Sybil attacks create low-overlap boundary edges detectable through Jaccard-based curvature computation, enabling throttling parameterized by ε (minimum weight) and α (throttling aggressiveness).
3. **Post-Quantum Security Architecture**: A complete implementation using NIST-standardized primitives: ML-DSA (FIPS 204), ML-KEM (FIPS 203), SHA3-256, and Plonky3 STARK proofs.
4. **Privacy-Preserving Reputation**: Zero-knowledge proofs allowing nodes to demonstrate reputation thresholds without revealing identity, enabling privacy-preserving weighted consensus.
5. **Production Implementation**: A tested Rust implementation comprising 9 crates (~9,826 lines of core logic; ~60,000 lines total including tests, benchmarks, and tooling) with 349+ unit/integration tests.
6. **Capability-Coherence Identity Protocol (CCIP)**: A three-layer identity architecture unifying DID-based persistent identity, object-capability authority, and petname-based naming, where introduction and delegation chains are DAG transactions subject to the same curvature analysis used for consensus.
7. **Coherence-as-Value**: A non-token value measure where topological mass serves as the system's native, non-transferable value, making parasitic extraction structurally detectable and automatically throttled.
8. **Substrate-Independent Agent Identity**: A unified identity model where human and non-human agents (did:agi:*) participate under identical rules, with coherence detection applying uniformly regardless of agent substrate.

1.4 Paper Organization

Section 2 provides background on graph curvature, post-quantum cryptography, and object-capability security. Section 3 describes the TMC protocol design. Section 4 presents the Capability-Coherence Identity Protocol including DID registration, petname naming, capability transactions, and coherence-as-value. Section 5 provides security analysis. Section 6 details the implementation. Section 7 provides evaluation. Section 8 discusses related work. Section 9 concludes.

---

2 Background

2.1 Ollivier-Ricci Curvature and Jaccard Approximation

Ollivier-Ricci curvature extends the notion of Ricci curvature from Riemannian geometry to discrete metric spaces. For an edge (u, v) in a graph G, the Ollivier-Ricci curvature $\kappa(u, v)$ measures how much the neighborhoods of u and v overlap compared to their distance.

Definition 2.1 (Ollivier-Ricci Curvature). For vertices u, v connected by an edge in graph G, let μ_u and μ_v be probability distributions over the neighbors of u and v respectively. The Ollivier-Ricci curvature is:

$$\kappa(u, v) = 1 - \frac{W_1(\mu_u, \mu_v)}{d(u, v)}$$

where W_1 denotes the Wasserstein-1 (earth mover's) distance and $d(u, v)$ is the graph distance.

Computational Challenge: Exact Ollivier-Ricci curvature requires solving optimal transport, which is $O(n^3)$ for general graphs—prohibitive for real-time consensus.

Jaccard Approximation: Following Pal et al. (2017), we use the generalized Jaccard curvature (gJC):

$$\kappa_J(u, v) = 2 \cdot J(N(u), N(v)) - 1$$

where $J(A, B) = |A \cap B|/|A \cup B|$ is the Jaccard index of neighbor sets. This is computable in $O(|N(u)| + |N(v)|)$ time.

Approximation Quality: Pal et al. demonstrate that gJC asymptotically aligns with Ollivier-Ricci curvature in relevant graph regimes. Critically for our purposes, the *sign* of curvature (positive vs. negative) is preserved, which is what determines throttling behavior.

Key Properties:

- **Positive curvature** ($\kappa_J > 0$): Dense communities where neighbors of adjacent nodes overlap significantly
- **Negative curvature** ($\kappa_J < 0$): Bridge edges connecting distinct communities with minimal overlap
- **Zero curvature** ($\kappa_J = 0$): Exactly 50% neighborhood overlap

Relevance to Sybil Detection: Sybil clusters are, by construction, weakly connected to the honest network. The edges connecting Sybils to honest nodes have low neighborhood overlap because Sybil identities primarily connect to each other, not to the honest network's internal structure. This manifests as negative Jaccard curvature on boundary edges.

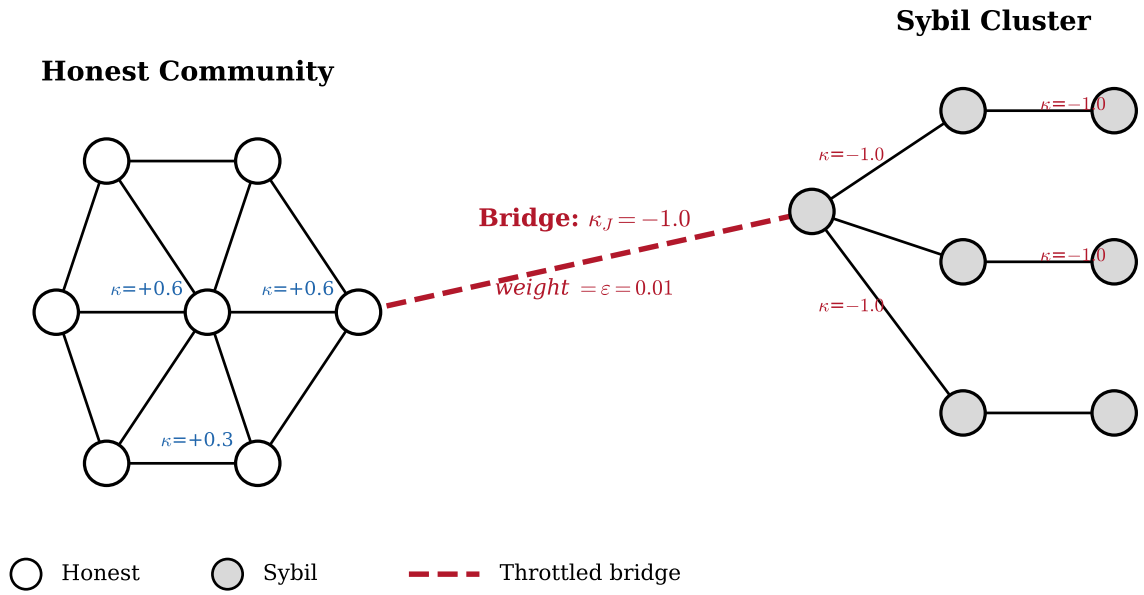

Figure 1: Honest community with dense internal connectivity (positive curvature) versus Sybil cluster connected through a throttled bridge edge (negative curvature). Edge weights from curvature throttling reduce Sybil influence by a factor of $1/\varepsilon$.

Alternative: Forman-Ricci Curvature: Forman-Ricci curvature provides another discrete curvature notion with $O(1)$ per-edge computation (depending only on endpoint degrees), but captures less neighborhood structure than Jaccard. The tradeoff between computational cost and geometric fidelity should be evaluated for specific deployment scenarios.

2.2 Post-Quantum Cryptography

The NIST Post-Quantum Cryptography standardization process finalized standards in August 2024:

- **ML-DSA (FIPS 204)**: Module-Lattice Digital Signature Algorithm, formerly Dilithium. We use ML-DSA-87 (Level 5, equivalent to AES-256 classical security).
- **ML-KEM (FIPS 203)**: Module-Lattice Key Encapsulation Mechanism, formerly Kyber. We use ML-KEM-1024 for Level 5 security.
- **SHA3-256**: Hash function from the Keccak family. Provides 256-bit output with strong classical collision resistance. Quantum security follows standard hash assumptions; preimage resistance is ~128-bit under Grover's algorithm.

2.3 Zero-Knowledge Proofs

Zero-knowledge proofs allow a prover to convince a verifier of a statement's truth without revealing information beyond the statement's validity. We use **STARKs** (Scalable Transparent Arguments of Knowledge) via the Plonky3 framework:

- **Post-quantum secure**: Based on hash functions, not elliptic curves
- **No trusted setup**: Transparent, publicly verifiable parameters
- **Scalable**: Polylogarithmic verification time

2.4 Object-Capability Security

Object-capability (ocap) security models derive authority from possession of unforgeable references rather than identity-checked permissions. In an ocap system, holding a capability IS the authorization to perform the corresponding action—there is no separate step of checking the holder's identity against an access control list.

Relevance to distributed consensus: Traditional distributed systems separate identity (who you are) from authority (what you can do), creating an attack surface at the binding between them. Credential theft, privilege escalation, and impersonation all exploit this separation. By collapsing identity and authority into a single construct—the capability—this attack surface is eliminated.

Petname Systems: Petname systems (Lemmer-Webber et al., 2022) resolve Zooko's Triangle (decentralized, globally unique, human meaningful—pick two) by adding a local naming layer over cryptographic identifiers. Three name types provide defense-in-depth:

- **Petnames**: User-authoritative local bindings (cannot be overridden by network)
- **Edge names**: Names published by trusted entities, forming introduction chains
- **Proposed names**: Self-asserted names displayed with disambiguation markers

The introduction chain mechanism is structurally compatible with DAG-based transaction graphs: each introduction is a transaction, making trust propagation subject to the same topological analysis as consensus.

Decentralized Identifiers (DIDs): W3C DIDs provide cryptographic identifiers with no central registry. The did:agi:* method (draft proposal) extends this to non-human agents, enabling substrate-independent identity where human and AGI participants operate under identical protocol rules.

---

3 Protocol Design

3.1 System Model

We consider a directed acyclic graph (DAG) of transactions where:

- Each transaction references one or more parent transactions
- Transactions are signed using ephemeral ML-DSA keypairs
- Double-spending is prevented via nullifiers (hash commitments to one-time secrets)
- Conflict resolution is deterministic based on topological mass

3.1.1 Network Assumptions

- Nodes have partial, eventually-consistent views of the DAG
- Network is asynchronous with eventual delivery
- Adversary controls at most f fraction of edges incident to honest nodes

3.2 Transaction Structure

```
Transaction {
    id: Hash256,                 // SHA3-256 of serialized content
    ephemeral_pk: VerifyingKey,  // One-time ML-DSA public key
    signature: Signature,        // ML-DSA signature
    parents: Vec<Hash256>,       // References to parent transactions
    simhash: SimHash,            // Structural fingerprint (256-bit)
    nullifier: Nullifier,        // Double-spend prevention token
    reputation_claim: u64,       // Claimed reputation (ZK-proven)
    confidential_outputs: Vec<ConfidentialOutput>,  // Optional
}
```

Temporal ordering. The transaction struct contains no assigned timestamp or block height. Temporal position is derived from the DAG partial order via topological depth (Definition 3.3). This ensures all temporal properties are committed—determined solely by parent references, which are fixed at creation—rather than assigned by an external coordinator.

3.2.1 Parent Selection Constraints

To bound topology grinding (where adversaries search for favorable parent combinations), we impose:

0. **Parent count bounds**: Each non-genesis transaction must reference at least 1 and at most MAX_PARENTS = 8 parents. This bounds the ancestor set size for curvature computation and prevents parent flooding attacks where an adversary references hundreds of honest transactions to inflate neighborhood overlap.
1. **Tip requirement**: At least one parent must be a current tip (no children yet)
2. **Recency bound**: Parents must be within depth k of the current frontier depth (default $k = 100$)
3. **Deterministic component**: One parent must be the heaviest tip by mass (ties broken lexicographically)

These constraints limit the adversary's search space for favorable topological positions while maintaining DAG liveness.

3.2.2 Ephemeral Identities

Each transaction uses a fresh ML-DSA keypair, providing unlinkability between transactions from the same long-term identity. The nullifier mechanism prevents double-spending without revealing the permanent identity:

$$\text{nullifier} = H(\text{sk\_hash} \| \text{input\_commitment})$$

where $\text{sk\_hash} = H(\text{secret\_key})$ is derived from the user's long-term secret key and input_commitment uniquely identifies the input being spent. This construction ensures that any attempt to spend the same input twice produces an identical nullifier, enabling double-spend detection without revealing the spender's identity.

The epoch in the nullifier computation is derived from topological depth (Definition 3.3), not from an assigned block height.

3.2.3 Structural SimHash

SimHash provides a locality-sensitive hash that places topologically similar transactions nearby in hash space. SimHash is computed from *structural* inputs only:

$$\text{SimHash} = \text{LSH}(\text{parents} \| \text{history\_root})$$

Security Property: No user-controllable content (memo fields, amounts, addresses) can influence SimHash, preventing *content* grinding. Topology grinding is bounded by parent-selection constraints above.

3.3 Topological Depth

Definition 3.3 (Topological Depth). For a transaction t in DAG G, the topological depth is:

$$\text{depth}(t) = \begin{cases} 0 & \text{if } t \text{ is a genesis transaction} \\ 1 + \max_{p \in \text{parents}(t)} \text{depth}(p) & \text{otherwise} \end{cases}$$

Topological depth is a Lamport clock (Lamport, 1978) derived from the DAG partial order. It has the following properties:

1. **Committed:** Depth is determined solely by parent references, which are fixed at transaction creation. All nodes that have received the same set of transactions compute identical depths.
2. **Monotonic:** For any edge (u, v) where v references u as parent, $\text{depth}(v) > \text{depth}(u)$.
3. **Concurrent transactions:** Two transactions at the same depth with no ancestor relationship are genuinely concurrent from the DAG's perspective.
4. **Computation:** Depth is computed via memoized dynamic programming in topological order, achieving $O(V + E)$ for the full DAG and $O(1)$ amortized per transaction after initial computation.

Depth-Derived Epochs. Epochs for nullifier scoping are derived deterministically from depth:

$$\text{epoch}(t) = \lfloor \text{depth}(t)/K \rfloor$$

where K is the epoch window size (default $K = 100$). Since depth is a committed property, epoch is also committed—all nodes agree on a transaction's epoch without coordination.

Design Rationale. Previous protocol versions used an assigned `block: u64` field as a logical timestamp. This introduced an implicit coordination requirement: the assigning entity must be trusted to assign consistent values. Topological depth eliminates this dependency by deriving temporal position from the DAG structure itself. The epoch derivation $\lfloor \text{depth}/K \rfloor$ provides the discrete bucketing needed for nullifier scoping without statistical tests or view-dependent computation.

3.4 Online Mass Computation Specification

To ensure deterministic agreement despite partial network views, we define mass computation over a *confirmation window*:

Definition 3.4 (Confirmation Window). For a transaction t, the confirmation window $W(t, d)$ consists of all descendants of t with topological depth $\leq \text{depth}(t) + d$ and path distance $\leq$ MAX_PATH_DEPTH from t, where d is the confirmation depth parameter. Topological depth is defined in Definition 3.3.

Definition 3.5 (Ancestor-Depth Neighbor Set). For curvature computation, the neighbor set of transaction t at depth k is defined recursively:

$$N_k(t) = \text{parents}(t) \cup \bigcup_{p \in \text{parents}(t)} N_{k-1}(p)$$

with base case $N_0(t) = \varnothing$. The default depth is $k = 2$ (parents and grandparents).

This ancestor-only definition ensures curvature is computed exclusively from committed transaction data. Parents are fixed at transaction creation and never change, making curvature **immutable by construction**—no view-consistency

assumptions or freezing semantics are required. All honest nodes compute identical curvature for any edge, regardless of which later transactions they have observed.

With MAX_PARENTS = 8 and $k = 2$, the ancestor set is bounded by $|N_k(t)| \leq 8 + 64 = 72$ nodes, ensuring $O(1)$ curvature computation per edge.

Design rationale: A bidirectional definition (parents + children) would make curvature dependent on which children a node has *seen*, introducing view-dependence that an adversary could exploit through selective relay. The ancestor-only definition eliminates this attack surface entirely.

Definition 3.5.1 (Curvature Immutability). Under the ancestor-depth neighbor set (Definition 3.5), the curvature of an edge (u, v) is fully determined by the parent lists of u, v, and their ancestors up to depth k. Since parent lists are committed at transaction creation and never modified, curvature is **immutable by construction**.

Nodes cache curvature values permanently for performance. No freezing semantics, confirmation depth requirements, or view-consistency assumptions are needed for curvature computation. (Confirmation depth remains relevant for finality assessment—see §3.5.)

Implementation: The curvature cache stores `(edge, value)` pairs. On first computation, the value is cached. All subsequent lookups return the cached value without recomputation.

Convergence: Nodes compute mass at a confirmation depth d. The default $d = 6$ is suitable for testing and low-value transactions; production deployments should use $d = 50$–200 (see Appendix C). Curvature computation requires no network synchronization; it is deterministic from committed transaction content. Mass convergence depends only on descendants propagating within the confirmation window, which is ensured by standard network delivery assumptions.

3.5 Topological Mass Computation

Given a transaction t and its confirmation window $W(t, \tau)$, the topological mass is:

$$M(t) = \left(\sum_{d \in W(t,\tau)} w(t,d) \cdot (1 + \log(1 + r_d)) \right) \cdot D(t)$$

where:

- $w(t, d)$ is the curvature-weighted path weight from t to descendant d
- r_d is the (ZK-verified) reputation of the transaction author
- $D(t)$ is the diversity score: $|\text{unique\_supporters}| \cdot (1 + \log(1 + R_{\text{total}})/10)$[1]

The continuous reputation formula $(1 + \log(1 + r_d))$ is presented for analytical clarity; the implementation uses the discrete bucketed weights from §3.7, which are compatible with zero-knowledge proofs.

3.5.1 Path Weight Definition

Definition 3.6 (Path Weight). For transactions t (source) and d (target) connected by one or more paths in the DAG, the path weight is defined as the maximum multiplicative weight over all paths:

$$w(t,d) = \max_{P \in \text{paths}(t,d)} \prod_{e \in P} \text{edge\_weight}(e)$$

where $\text{paths}(t, d)$ is the set of all directed paths from t to d with length ≤ MAX_PATH_DEPTH.

Computation: We use depth-first search with pruning on the DAG. Since edge weights are ≤ 1, the product can only decrease along a path, enabling early termination when the accumulated weight falls below the current best. For DAGs within the confirmation window, this achieves $O(V + E)$ expected complexity.

[1]The per-window nullifier construction described here provides Sybil-resistant diversity counting; the current implementation approximates this using ephemeral public keys (see §5.8.3 for the implementation gap and planned closure).

Intuition: The multiplicative formulation means that paths crossing multiple negative-curvature bridges (Sybil boundaries) are exponentially attenuated. A path crossing k bridge edges with weight ε contributes at most ε^k of its nominal weight. With default $\varepsilon = 0.01$, three bridge crossings reduce contribution to 10^{-6}.

Why Max-Path, Not Sum-Over-Paths: We use max-path aggregation (not sum over paths) to prevent path-count amplification, which would create a vulnerability where attackers manufacture many low-weight paths that sum to high mass. Redundant connectivity helps only insofar as it: (a) improves the best path weight (by providing alternative routes that avoid negative-curvature edges), or (b) increases the set of reachable descendants (more nodes can be reached at all). The honest network's advantage comes from having at least one high-weight path to each descendant, plus genuine supporter diversity—not from having many paths per descendant.

Note on Identity for Diversity: Supporter identity for diversity counting uses **per-window supporter tags**, a construction that enables unique supporter counting without global linkability:

$$\text{supporter\_tag} = H(\text{identity\_secret} \| \text{window\_id} \| \text{tx\_id})$$

The ZK proof demonstrates: 1. The prover knows an identity secret for an account in the committed Merkle tree 2. The supporter tag was correctly computed from that identity secret

Properties: - **Sybil-resistant**: A single identity produces one tag per (window, transaction) pair - **Locally linkable**: Same identity supporting the same transaction in the same window produces identical tags (enables deduplication) - **Globally unlinkable**: Tags for different windows/transactions reveal nothing about whether they share an identity - **Privacy-preserving**: The identity secret is never revealed; only the tag and proof are public

This construction is analogous to Zcash nullifiers but for "unique support" rather than double-spend prevention.

3.5.2 Curvature Throttling

For each edge (u, v) on the path from t to d, we compute a weight based on Jaccard curvature:

$$\text{edge\_weight}(u, v) = \text{clamp}(1 + \alpha \cdot \kappa_J(u, v),\ \varepsilon,\ 1.0)$$

where: - $\alpha = 3$ (throttling aggressiveness) - $\varepsilon = 0.01$ (minimum weight floor)

Effect: Edges with strongly negative curvature (low-overlap bridges) have their influence reduced to ε of normal, while edges in dense communities ($\kappa_J > 0$) retain full weight (clamped at 1.0; positive curvature does not amplify beyond baseline).

Parameter Sensitivity: See Section 7 for analysis across parameter ranges.

Saturation Rationale: The rapid saturation to ε for $\kappa_J \leq -1/3$ is intentional: bridge edges with less than ~33% neighborhood overlap are already structurally anomalous in well-connected networks (honest nodes with degree ≥ 5 typically achieve >50% overlap with neighbors). The sharp cutoff prevents "slightly negative" bridges from being exploited as Sybil channels. See §7.3 for sensitivity analysis across α values.

3.5.3 Alternative Throttling Functions

The linear throttling function above is chosen for simplicity and interpretability. Alternative functions include:

- **Quadratic**: weight $= \max(\varepsilon, 1 - \alpha \cdot \kappa^2)$ — gentler near zero curvature, harsher at extremes
- **Sigmoid**: weight $= \varepsilon + (1 - \varepsilon) \cdot \sigma(\beta \cdot \kappa)$ — smooth transition with tunable steepness
- **Step function**: weight $= 1$ if $\kappa > \theta$ else ε — simple but discontinuous
- **Adaptive**: α, ε adjusted dynamically based on observed attack patterns or network density

Alternative functions may be preferable in specific deployment scenarios; the linear function provides a reasonable default.

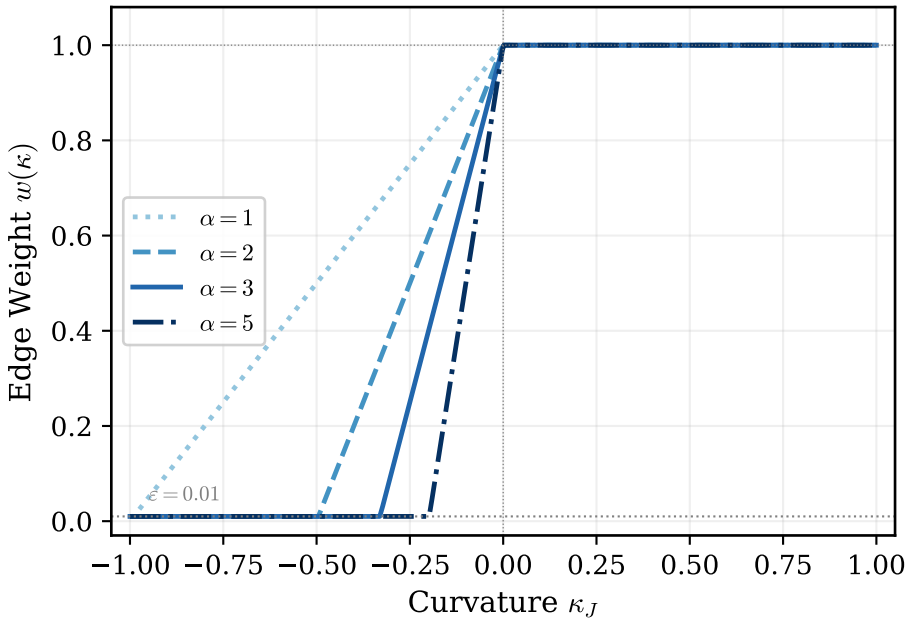

Figure 2: Throttling weight function $w(\kappa) = \text{clamp}(1 + \alpha\kappa,\ \varepsilon,\ 1.0)$ for varying aggressiveness parameter α. Higher α creates steeper throttling of negative-curvature edges. All curves share the floor $\varepsilon = 0.01$ and ceiling 1.0 (positive curvature does not amplify beyond baseline).

3.5.4 Throttling Application Points

Beyond mass computation, throttling weights may be applied at various points in the protocol:

- **Parent selection**: Probability of selecting a transaction as parent weighted by edge curvature
- **Relay priority**: Order of transaction propagation in the P2P network
- **Fee multiplier**: Required transaction fee increased for transactions creating low-curvature edges
- **Admission control**: Rejection of transactions that would create too many negative-curvature edges from a single source

These additional application points provide defense-in-depth against Sybil attacks.

3.5.5 Bridge Exemption Mechanisms

Legitimate network bridges (nodes connecting geographically or topologically distinct honest communities) may exhibit negative curvature despite being honest. To prevent false positives while avoiding centralization:

1. **Mutual Recognition Heuristic**: Two nodes A and B are considered *mutually recognized* if:
   - A references ancestors of B deeper than confirmation depth d
   - B references ancestors of A deeper than confirmation depth d
   - This interwoven history indicates long-term alliance, not sudden Sybil attachment

   For edges between mutually recognized nodes, curvature penalty is relaxed: effective_$\alpha = \alpha/3$.

2. **Age-based exemption**: Edges where both endpoints have been present in the graph for longer than a threshold duration (e.g., 10,000 depth units) receive reduced throttling: effective_$\alpha = \alpha/2$.

3. **Identity-scarcity tie-in**: Bridge exemptions require the same ZK-proven identity mechanism used for diversity counting. This prevents attackers from creating "reputable" Sybils specifically to exploit exemptions.

Avoiding "Rich Get Richer": We deliberately avoid mass-based exemptions (e.g., "top 1% by historical mass get

full weight") because this creates centralization pressure where only historically heavy nodes can form bridges. The mutual recognition heuristic instead rewards *demonstrated integration* over *accumulated mass*, allowing new legitimate sub-networks to join without requiring pre-existing heavy nodes.

3.6 Conflict Resolution

When two transactions A and B conflict (e.g., double-spend via same nullifier), the winner is determined by topological mass:

$$\text{winner} = \begin{cases} A & \text{if } M(A) > M(B) \\ B & \text{if } M(B) > M(A) \\ \text{lexicographically\_smaller}(A, B) & \text{otherwise} \end{cases}$$

3.6.1 Finality Criterion

A branch is considered finalized when its mass exceeds all competitors by a factor of $\phi = 10$:

$$\text{finalized}(A) \iff \forall B \in \text{competitors} : M(A) > \phi \cdot M(B)$$

The finality ratio $\phi = 10$ is chosen so a branch must accumulate at least $10\times$ the topological mass of any competitor, providing high-confidence finality analogous to Bitcoin's 6-confirmation heuristic but based on structural integration rather than hash power.

3.7 Bootstrap Mechanism

At network genesis, topological structure is insufficient for curvature-based security. We define a bootstrap phase with **ramped throttling** to prevent the bootstrap cliff attack.

3.7.1 Ramped Throttling

Rather than a binary switch from "no throttling" to "full throttling," the protocol uses a linear ramp:

$$\alpha_{\text{effective}}(t) = \alpha_{\max} \cdot \min\left(1, \frac{\max(0, \text{depth}(t) - D_{\text{start}})}{D_{\text{end}} - D_{\text{start}}}\right)$$

where: - $D_{\text{start}} = 1000$ (throttling begins ramping at this depth) - $D_{\text{end}} = 6000$ (full throttling at this depth) - $\alpha_{\max} = 3$ (maximum throttling aggressiveness)

Security Rationale: Without ramped throttling, an attacker could pre-compute a massive dense DAG offline and broadcast it just before the throttling threshold to take over before throttling activates. The linear ramp ensures that even if attackers try to exploit the transition, their advantage is limited to a fraction of the bootstrap period.

Note that $\alpha_{\text{effective}}$ is computed per-transaction based on that transaction's topological depth, not from a global "current network depth." This prevents depth inflation attacks where an adversary self-chains to high depth: their transactions reach full throttling, but since self-chains exhibit poor internal curvature (low ancestor overlap), full throttling makes the attack strictly worse for the attacker.

3.7.2 Bootstrap Phases

1. **Genesis transaction**: depth 0, a single trusted genesis transaction with hardcoded identity
2. **Pure descendant count** (depth $< D_{\text{start}}$): Mass uses descendant count without curvature throttling ($\alpha = 0$)
3. **Ramp-up** ($D_{\text{start}} \leq$ depth $< D_{\text{end}}$): Throttling increases linearly
4. **Full activation** (depth $\geq D_{\text{end}}$): Complete curvature-based throttling ($\alpha = \alpha_{\max}$)

3.7.3 Alternative Genesis Mechanisms

Alternative approaches to network genesis include:

- **Multiparty genesis ceremony**: Threshold signatures from multiple founding entities, requiring k-of-n agreement
- **Widely-witnessed checkpoint**: Genesis references external time sources (e.g., block hashes from multiple existing blockchains)

The choice of genesis mechanism depends on the desired trust assumptions and decentralization requirements.

3.7.4 Tip Selection for Curvature Optimization

Under ancestor-depth neighbor sets, transactions that share ancestors with their parents achieve higher curvature and therefore higher edge weight in mass computation. We recommend the following tip selection strategy:

1. **At least one confirmed ancestor**: Include at least one parent with depth $\geq d$ (a "confirmed" transaction). This creates ancestor overlap with other recent transactions that also reference confirmed ancestors.
2. **At least one recent tip**: Include at least one current tip (a transaction with no children yet). This maintains DAG liveness.
3. **Cross-reference when possible**: Reference parents from different sub-branches of the DAG. This creates ancestor overlap that strengthens curvature on cross-branch edges.

Applications with naturally sequential transaction patterns (e.g., point-of-sale) should reference at least one non-sequential parent per transaction to maintain positive curvature.

3.8 Zero-Knowledge Reputation Proofs

To include reputation in mass computation without revealing identity, we use a Plonky3 STARK circuit with **bucketed reputation**.

3.8.1 Bucketed Reputation

Rather than proving exact reputation values (which the mass formula cannot use without revealing identity), proofs assert membership in a discrete reputation bucket:

Bucket	Reputation Range	Weight (w_b)
0	$[0, 10)$	1.0
1	$[10, 50)$	1.3
2	$[50, 100)$	1.5
3	$[100, 500)$	1.8
4	$[500, 1000)$	2.0
5	$[1000, \infty)$	2.2

The mass formula uses the bucket weight as a public lookup table:

$$\text{rep\_bonus}(b) = w_b$$

This resolves the inconsistency between ZK proofs (which prove predicates) and mass computation (which requires values): the predicate is "I belong to bucket b" and the value is the corresponding w_b.

3.8.2 Circuit Design

Public Inputs: - Merkle root of account states (trusted anchor from recent finalized state) - Claimed bucket $b \in \{0, ..., B-1\}$ - Current epoch (replay protection)

Private Inputs: - Account state (identity, reputation, transaction count) - Merkle path from account to root

Circuit Constraints: 1. Merkle path is valid from leaf to root 2. Leaf hash matches the committed account state 3. Account reputation falls within bucket b's range: bucket_min[b] $\leq r <$ bucket_max[b]

The proof size is $O(\log n)$ where n is the number of accounts, and verification is $O(\log n)$ hash computations.

3.9 Confidential Transactions

For privacy-preserving value transfer, we use hash-based commitments:

$$C(v, r) = H(\text{"AMOUNT\_COMMIT\_V1"} \| v \| r)$$

where v is the value and r is a 256-bit blinding factor.

Important: Unlike Pedersen commitments, hash commitments are not additively homomorphic. Balance verification therefore happens *inside* the ZK circuit rather than via algebraic operations on commitments.

Balance Proof: A Plonky3 circuit takes private inputs (all input values/blindings, all output values/blindings) and proves $\sum$ inputs = $\sum$ outputs without revealing values.

Range Proof: A separate circuit proves $0 \leq v < 2^{64}$ for each output, preventing overflow attacks.

Stealth Addresses: Recipients are addressed via ML-KEM, allowing senders to create unlinkable one-time addresses.

---

4 Capability-Coherence Identity Protocol

We now extend TMC with an identity layer that unifies naming, capability, and value measurement using the same curvature mechanics developed for consensus.

4.1 Design Principle: Consequence Closure

The identity layer preserves TMC's core property: every action creates consequence chains that close within the same substrate. Identity is not asserted but demonstrated through topologically verifiable participation. Authority is not granted but held as capability. Value is not represented by tokens but measured as coherence.

4.2 DID-Based Persistent Identity

Each participant holds a DID anchored in the Disentangle DAG:

$$\text{did:disentangle:}\langle \text{base58}(H(\text{pk}_{\text{root}}))\rangle$$

where pk_{root} is the entity's root ML-DSA public key. Non-human agents use the convention did:disentangle:agi:□id□ with no privilege distinction—the protocol is agnostic to substrate.

DID documents are registered through a special transaction type in the DAG. The registration transaction IS the entity's first consequence in the system; there is no identity prior to participation.

4.2.1 Ephemeral Key Binding

The existing ephemeral key model (Section 3.2.2) is preserved as a privacy layer atop persistent DIDs. A ZK proof (Plonky3 STARK) demonstrates that an ephemeral key derives from a registered DID without revealing which DID:

DID Binding Circuit:

Public inputs: ephemeral public key, nullifier, reputation bucket, epoch

Private inputs: root secret key hash, DID Merkle proof, reputation score, derivation path

Constraints: 1. Ephemeral key correctly derived from root key material 2. Nullifier correctly computed for epoch 3. DID exists in committed registry (Merkle proof) 4. Reputation satisfies claimed bucket threshold

This enables reputation accumulation to the persistent DID while preserving transaction-level unlinkability.

4.3 Petname System

The petname system maps human-readable names to DIDs, operating as a local-first database with network-propagated edge names.

4.3.1 Introduction Chains as DAG Transactions

Each introduction is a transaction in the DAG:

$$\text{IntroductionTx} = \langle \text{introducer}, \text{introduced}, \text{edge\_name}, \text{context}, \text{parents} \rangle$$

This is the critical design choice: trust propagation is subject to the same curvature analysis as consensus. An entity that introduces many DIDs without reciprocal integration exhibits negative curvature in the introduction subgraph—the geometric signature of an introduction mill.

Proposition 4.1 (Introduction Mill Detection). Let v be a node in the identity subgraph that has introduced k other nodes but has been introduced by at most $j \ll k$ nodes. For any edge (v, w) where w is one of v's introduced nodes, the Jaccard index satisfies $J(N(v), N(w)) \leq j/(j + k - 1)$, yielding negative curvature when $k > j + 1$.

Argument: The neighbors of v in the identity subgraph are the k introduced nodes plus at most j introducers. The neighbors of w include v and any entities w has independently interacted with. Since manufactured introductions are one-directional (mill → target), the introduced nodes share few neighbors with the mill, yielding low Jaccard overlap.

4.3.2 Phishing Defense

In the petnames model, the user's interface resolves DIDs through locally authoritative petnames, not through visual string comparison. The attack surface collapses from "all visually confusable strings" to "compromise the local petname database." Edge name disambiguation ("Pizza Piano.2") explicitly marks untrusted proposed names.

4.3.3 Naming Hubs

Existing naming authorities (DNS, CAs, standards bodies) are absorbed as naming hubs within the petname system on equal footing with any other entity. No naming hub has privileged authority; trust in a hub is a local petname decision.

4.4 Object Capability Transactions

Authority is held as unforgeable capability references:

$$\text{Capability} = \langle \text{id}, \text{issuer}, \text{subject}, \text{constraints}, \text{delegatable}, \text{proof} \rangle$$

where id = H(capability content) and proof is an ML-DSA signature chain.

4.4.1 Capability Delegation as Trust Topology

Delegation creates edges in the identity subgraph, subject to curvature analysis:

- Reciprocal delegation (A delegates to B, B's exercise benefits A) creates positive curvature
- Asymmetric delegation farming (entity accumulates capabilities without reciprocal benefit) creates negative curvature
- Capability misuse creates topological evidence at the delegation edge without requiring the issuer to actively monitor

Proposition 4.2 (Capability Misuse Detection). If entity B holds a capability delegated from A, and B's exercise of that capability creates transactions that do not integrate coherently with A's transaction neighborhood, the edge (A, B) in the delegation subgraph will exhibit decreasing curvature over time as the neighborhood overlap between A and B diverges.

4.4.2 ZK Capability Proofs

Capability possession is proven in zero-knowledge:

Public inputs: capability ID, action hash, constraint commitments

Private inputs: delegation chain, capability document, holder key hash

Constraints: 1. Valid delegation chain from issuer to holder 2. All constraints satisfied 3. Delegation depth within limit 4. Capability not revoked

4.4.3 Consequence Return

The lifecycle of a capability creates a closed consequence loop:

$$\text{CREATE} \rightarrow \text{HOLD} \rightarrow \text{EXERCISE} \rightarrow \text{CONSEQUENCE} \rightarrow \text{(returns to issuer via topological mass)}$$

This consequence-return property means the protocol does not need external enforcement of delegation contracts. Misuse creates structural evidence that the topological mass mechanism automatically penalizes.

4.5 Coherence as Value

4.5.1 The Token Problem

Every token-based value system separates the value signifier (token) from the value signified (coherent participation). This separation is the exploit vector: tokens can be accumulated through parasitic extraction, transferred independently of coherent behavior, and weaponized for governance capture.

4.5.2 Topological Mass as Native Value

CCIP does not introduce a token. The system's native value measure is topological mass (Section 3.4), extended to the identity layer through a coherence profile:

$$C(v) = f\left(M(v),\ \bar{\kappa}(v),\ D_{\text{rel}}(v),\ \tau(v),\ C_{\text{cap}}(v),\ C_{\text{intro}}(v)\right)$$

where: - $M(v)$ = topological mass of v's transactions - $\bar{\kappa}(v)$ = mean local curvature of v's identity graph edges - $D_{\text{rel}}(v)$ = count of unique DIDs with positive-curvature edges to v - $\tau(v)$ = temporal depth (current frontier depth − depth of v's first transaction) - $C_{\text{cap}}(v)$ = ratio of coherent capability exercises to total - $C_{\text{intro}}(v)$ = average curvature of entities v introduced

Coherence-as-value has four structural properties:

1. **Non-purchasable**: No amount of tokens purchases topological mass
2. **Non-extractable**: Mass is a structural property, not a transferable asset
3. **Non-fakeable**: Manufacturing mass requires genuine structural integration
4. **Decaying**: Mass decays with half-life $T_{1/2}$ without ongoing participation

4.5.3 Coherence Decay

$$M_{\text{decayed}}(v, t) = M(v) \cdot 2^{-(t - t_{\text{last}})/T_{1/2}}$$

where t and t_{last} are topological depths and $T_{1/2} = 10{,}000$ depth units. This prevents reputation squatting.

4.5.4 Inter-Agent Value Flow

While coherence is non-transferable, cooperative capability exercise creates mutual coherence increase: both delegator and delegatee gain topological mass when delegation is exercised coherently. This is co-creation of structural integration, not value transfer. The incentive structure is intrinsic—no tokenomics required.

4.6 Coherence-Weighted Governance

Governance decisions are weighted by coherence profile:

$$\text{vote\_weight}(v) = C(v)$$

This makes plutocratic capture structurally impossible: accumulating governance weight requires genuine structural integration with the majority of the network. An entity attempting to manufacture governance weight through Sybil participation creates the negative curvature that TMC detects.

---

5 Security Analysis

The following security claims are supported by informal arguments and simulation results. Formal proofs under explicit assumptions are left for future work.

5.1 Claim 1: Sybil Influence Bound (Jaccard Curvature)

Statement: Under the Jaccard curvature approximation with throttling parameters (α, ε), an adversary whose Sybil cluster connects to the honest network through edges with Jaccard index $J \leq J_0$ (neighborhood overlap) can gain at most influence proportional to $\max(\varepsilon, 1 - \alpha(1 - 2J_0))$ per connection edge.

Argument:

1. By construction, Sybil identities primarily connect to each other, not to the honest network's internal structure.
2. For a boundary edge (s, h) where s is a Sybil node and h is an honest node:
   - $N(s)$ consists mostly of other Sybil nodes
   - $N(h)$ consists of honest nodes
   - Therefore $|N(s) \cap N(h)| \ll |N(s) \cup N(h)|$, yielding low Jaccard index $J(N(s), N(h)) \leq J_0$
3. The Jaccard curvature is $\kappa_J = 2J_0 - 1 < 0$ when $J_0 < 0.5$
4. Edge weight becomes $\text{clamp}(1 + \alpha\kappa_J,\ \varepsilon,\ 1.0) = \text{clamp}(1 + \alpha(2J_0 - 1),\ \varepsilon,\ 1.0)$
5. With $\alpha = 3$, $\varepsilon = 0.01$, and $J_0 = 0.1$ (10% overlap): $\text{weight} = \max(0.01, 1 - 3 \cdot 0.8) = \max(0.01, -1.4) = 0.01$

Corollary: With default parameters, boundary edges with $< 33\%$ neighborhood overlap are throttled to minimum weight ε.

Limitation: If an adversary can achieve high neighborhood overlap (e.g., by controlling honest-looking nodes that integrate into the community before attacking), the throttling is less effective. See Section 5 for discussion.

5.2 Claim 1a: Curvature Determinism

Statement: Under ancestor-depth neighbor sets (Definition 3.5), the curvature $\kappa_J(u, v)$ of any edge (u, v) is fully determined by committed transaction data. All honest nodes compute identical curvature for any edge regardless of which subsequent transactions they have observed.

Justification. The ancestor set $N_k(t)$ is computed exclusively from parent lists, which are fixed at transaction creation. No children, descendants, or view-dependent data enter the computation. Two honest nodes with potentially different

views of the DAG's tips will nonetheless compute identical ancestor sets for any transaction they both know, since the ancestor computation traverses only backward (parent) edges, all of which are committed.

5.3 Claim 1b: Bounded Ancestor Overlap

Statement: With MAX_PARENTS $\leq P$ and ancestor depth k, the maximum Jaccard index between a Sybil transaction s and an honest transaction h is bounded by:

$$J(N_k(s), N_k(h)) \leq \frac{|N_k(s) \cap N_k(h)|}{|N_k(s) \cup N_k(h)|}$$

which is maximized when s references all of h's ancestors as its own parents or ancestors—requiring s to dedicate its limited parent slots ($\leq P$) to honest transactions, thereby increasing its genuine structural integration. An attacker cannot simultaneously achieve high boundary curvature and maintain a structurally isolated attack cluster.

Justification. Each transaction has at most P parents. The total ancestor set at depth k is bounded by $|N_k(t)| \leq \sum_{i=0}^{k-1} P^{i+1} = P(P^k - 1)/(P - 1)$. To achieve high Jaccard on a boundary edge, the attacker must use parent slots to reference honest ancestors, reducing the slots available for references to the Sybil cluster. This creates a fundamental tradeoff between boundary camouflage and cluster connectivity.

5.4 Claim 2: Content Grinding Resistance

Statement: An adversary cannot improve their topological position by manipulating transaction *content* (memo fields, amounts, addresses).

Argument: SimHash is computed exclusively from: - Parent transaction hashes (determined by DAG state at submission time) - History Merkle root (determined by identity history)

Neither input includes user-controllable content fields.

Limitation: This does not prevent *topology* grinding via parent selection. Parent-selection constraints (Section 3.2.1) bound but do not eliminate this attack surface. An adversary can still search over valid parent combinations, though the search space is constrained to recent tips.

5.5 Claim 3: Post-Quantum Security

Statement: All consensus-critical cryptographic operations achieve NIST Security Level 5 against quantum adversaries.

Argument: - Signatures: ML-DSA-87 (FIPS 204, Level 5) - Key encapsulation: ML-KEM-1024 (FIPS 203, Level 5) - Hashing: SHA3-256 (preimage ~128-bit quantum under Grover; collision resistance under standard assumptions) - ZK proofs: Plonky3 STARKs (hash-based, no elliptic curve operations)

No elliptic curve operations appear in the consensus-critical path.

5.6 Claim 4: Privacy Properties

Statement: Transaction authors are unlinkable across epochs, and reputation claims reveal only threshold satisfaction.

Argument: 1. Each transaction uses a fresh ephemeral ML-DSA keypair, generated independently 2. Nullifiers prevent double-spending but reveal only epoch membership (not identity) 3. ZK proofs are zero-knowledge: verifier learns only that reputation ≥ threshold 4. Stealth addresses ensure recipients are unlinkable via ML-KEM one-time key derivation

5.7 Claim 5: Introduction Mill Detection

Statement: An entity that introduces k DIDs while receiving at most j introductions ($j \ll k$) creates boundary edges with Jaccard curvature $\kappa_J \leq 2j/(j + k - 1) - 1 < 0$ in the identity subgraph.

Argument: Follows from Proposition 4.1. The introduction mill has asymmetric neighborhood topology: its neighbor set is large (the introduced entities) but shares minimal overlap with any individual introducee's neighbor set.

5.8 Claim 6: Capability Misuse is Geometrically Detectable

Statement: An entity exercising delegated capabilities incoherently with the delegator's transaction neighborhood creates decreasing curvature on the delegation edge over time.

Argument: Follows from Proposition 4.2. Coherent exercise integrates the delegatee's transactions into the delegator's neighborhood (increasing overlap). Incoherent exercise creates transactions that diverge from the delegator's neighborhood (decreasing overlap).

5.9 Claim 7: Governance Capture Resistance

Statement: Under coherence-weighted governance, capturing majority vote weight requires genuine structural integration with the majority of network participants.

Argument: Vote weight is topological mass, which is a curvature-weighted measure of structural integration. Manufacturing mass through Sybil identities creates negative curvature at boundary edges (Claim 1). Therefore, the only path to majority governance weight is genuine integration, which IS coherent participation.

5.10 Limitations and Attack Surfaces

5.10.1 High-Overlap Sybil Attack

An adversary who invests time building reputation and connections before attacking could achieve higher neighborhood overlap, reducing throttling effectiveness.

Under ancestor-depth neighbor sets with MAX_PARENTS = 8, the attacker's ability to inflate boundary curvature is bounded by the parent count constraint. Each Sybil transaction can reference at most 8 parents, limiting the ancestor overlap achievable at the boundary. The attacker faces a fundamental tradeoff: parent slots spent on honest references improve boundary curvature but reduce connectivity to the Sybil cluster, weakening internal mass accumulation.

Mitigation: The cost is real integration into the network over time, converting a Sybil attack into a long-range attack with similar economics to PoS long-range attacks.

5.10.2 Pre-Mine Attack

An adversary creates a large offline cluster with dense positive-curvature internal mesh, then connects simultaneously via multiple bridges.

Analysis: - Internal edges have high curvature but are disconnected from honest network - All paths from honest network traverse low-overlap boundary edges - Mass computation only counts descendants reachable through the DAG - The offline cluster's "mass" is zero until connected, at which point boundary throttling applies

Mitigation: The attack reduces to the standard Sybil case—boundary edges are still throttled.

5.10.3 Diversity Counting Implementation Gap

The per-window supporter tag construction (Section 3.3) provides Sybil-resistant diversity counting in theory, but the current implementation uses ephemeral public keys as a proxy for unique supporter identity. Since an adversary can generate unlimited ephemeral keys, this proxy is vulnerable to diversity inflation: a Sybil cluster of k keys contributes a diversity score proportional to k, regardless of whether those keys represent distinct real-world participants.

Mitigation: Curvature throttling remains effective independent of diversity counting–bridge edges still receive low weight, attenuating Sybil path contributions multiplicatively. The combination of throttled path weights, zero-reputation bucket weights (1.0x minimum), and honest participants' higher reputation provides resistance even without the full ZK supporter tag mechanism. Implementing the per-window nullifier circuit described in Section 3.3 would close this gap entirely and is planned for a future release.

5.10.4 Computational Denial of Service

Curvature computation on every transaction could enable DoS.

Mitigation: - Curvature is computed lazily during conflict resolution, not on every transaction - Results are cached with TTL matching confirmation depth - Proof-of-work anti-spam on transaction submission bounds submission rate

5.10.5 Long-Range Trust Chain Attack

An adversary could build legitimate reputation and trust chains over an extended period, then leverage accumulated capabilities and introductions for a coordinated attack.

Mitigation: Coherence decay ($T_{1/2} = 10{,}000$ depth units) limits the window of exploitable accumulated trust. The attack converts to a long-range attack with comparable economics to PoS long-range attacks.

Long-range integration attacks represent the fundamental cost of Sybil resistance without central authority. TMC raises this cost from economic (purchasable) to structural (requiring genuine participation), but cannot eliminate it. An attacker who participates honestly for an extended period earns genuine coherence—this is by design, as there is no behavioral distinction between such an attacker and an honest participant who later changes their position.

5.10.6 AGI Coordination Attack

Multiple AGI agents with did:agi:* DIDs could coordinate to build mutual coherence faster than human participants, potentially dominating governance.

Mitigation: Curvature analysis detects artificial clustering regardless of agent substrate. Coordinated AGI agents that primarily interact with each other (rather than the broader network) exhibit the same negative boundary curvature as any Sybil cluster. Introduction cooldowns (Section 4.2) bound the rate of trust network growth.

---

6 Implementation

6.1 Architecture

The implementation comprises nine Rust crates:

Crate	Purpose	Tests
`disentangle-crypto`	ML-DSA, ML-KEM, SHA3-256	17
`disentangle-simhash`	Structural LSH	9
`disentangle-dag`	Transaction graph + depth computation	48
`disentangle-consensus`	Mass computation	38
`disentangle-zkp`	Plonky3 STARK circuits	43
`disentangle-node`	Mempool, PoW anti-spam	25
`disentangle-p2p`	libp2p networking, PQ transport	77
`disentangle-identity`	DIDs, petnames, capabilities, coherence	43
`disentangle-cli`	Command-line interface	43

Totals: ~9,826 lines of core logic; ~60,000 lines including tests, benchmarks, binary, and tooling. 349+ tests pass.

6.2 Key Design Decisions

6.2.1 Fixed-Point Arithmetic

All consensus-critical computations use fixed-point integers (SCALE = 65536) to ensure determinism across platforms:

```rust
pub type FixedPoint = i32;
pub const SCALE: i32 = 65536;

pub fn fp_mul(a: FixedPoint, b: FixedPoint) -> FixedPoint {
    ((a as i64 * b as i64) / SCALE as i64) as FixedPoint
}
```

Determinism Boundary: The following operations are consensus-critical and MUST use fixed-point arithmetic: - Curvature computation (κ_J values) - Edge weight calculation - Path weight products - Mass computation - Finality ratio comparisons

Display and debugging functions may convert to floating-point for human readability, but these values MUST NOT influence consensus decisions.

6.2.2 Jaccard Curvature Implementation

```rust
/// Jaccard curvature in fixed-point arithmetic for deterministic consensus.
pub fn jaccard_curvature(dag: &DAG, u: &NodeId, v: &NodeId) -> FixedPoint {
    let n_u = dag.neighbors(u);
    let n_v = dag.neighbors(v);

    let intersection = n_u.intersection(&n_v).count() as i32;
    let union = n_u.union(&n_v).count() as i32;

    if union == 0 { return 0; }

    // κ_J = 2·J(A,B) − 1 where J = |A∩B| / |A∪B|
    2 * fp_from_ratio(intersection, union) − SCALE
}
```

Complexity: $O(|N(u)| + |N(v)|)$ using hash set operations.

6.2.3 Depth Computation Implementation

```rust
impl TransactionDAG {
    /// Compute topological depth via memoized dynamic programming.
    /// O(V + E) for full DAG, O(1) amortized per transaction.
    pub fn depth(&mut self, id: &NodeId) -> u64 {
        if let Some(&cached) = self.depth_cache.get(id) {
            return cached;
        }
        // Iterative topological sort with memoization
        // (see implementation for full algorithm)
        ...
    }
}
```

The depth computation uses iterative topological sort to avoid stack overflow on deep DAGs, with results cached permanently (depth is immutable once computed, since parent references never change). Epoch derivation is a single integer division: `depth / EPOCH_WINDOW_SIZE`.

6.3 Computational Complexity

Operation	Complexity	Notes
Jaccard curvature (single edge)	$O(d)$	d = average node degree

Operation	Complexity	Notes
Ollivier-Ricci curvature (exact)	$O(n^3)$	Optimal transport; not used
Edge weight throttling	$O(1)$	Single arithmetic operation
Best path weight (DFS with pruning)	$O(V + E)$	Within confirmation window
Mass computation	$O(D \times P)$	D = descendants, P = avg path length
Conflict resolution	$O(2 \times \text{mass})$	Compare two transactions
ZK proof generation	$O(\text{polylog}(n))$	n = number of accounts
ZK proof verification	$O(\text{polylog}(n))$	Hash-based STARK

Complexity of Max-Path Weight: On the induced subgraph within the confirmation window (a DAG by construction), max-product path weight is computed via depth-bounded dynamic programming: process nodes in reverse topological order, propagating max-weight to each parent. This achieves $O(E)$ on the induced subgraph, where E is bounded by MAX_PATH_DEPTH × average degree.

Implementation Note: For long path computations, implementations may use `i128` intermediate values with checked arithmetic to prevent overflow. Saturating arithmetic may be used as a fallback, with overflow events logged for monitoring.

6.3.1 Hybrid PQ Transport

The P2P layer uses libp2p with Noise-XX for the initial handshake, then immediately re-keys using ML-KEM:

1. Complete standard Noise-XX handshake (backward compatible with non-PQ peers)
2. Exchange ephemeral ML-KEM-1024 encapsulation keys
3. Both sides encapsulate to peer's key
4. Derive hybrid session key: $K = H(K_{\text{noise}} \| K_{\text{kem\_init}} \| K_{\text{kem\_resp}})$
5. Re-key the Noise session with K

This provides post-quantum forward secrecy while maintaining compatibility.

6.4 Identity Layer Implementation

The `disentangle-identity` crate implements CCIP as an extension to the existing DAG:

- **DID operations**: Registration, update, key rotation, deactivation—all as DAG transactions with domain-separated hashing
- **Petname database**: Local-first with edge name caching; introduction chains stored in an identity subgraph overlay on the transaction DAG
- **Capability system**: Creation, delegation, invocation, revocation with delegation depth limits and constraint evaluation
- **Coherence profile**: Computed from topological mass, identity graph curvature, relational diversity, temporal depth, and capability/introduction coherence scores
- **Governance**: Proposal/vote transactions with coherence-weighted evaluation and diversity quorum

6.4.1 Identity Subgraph Curvature

Curvature on the identity subgraph uses the same Jaccard formula as transaction DAG curvature (Section 3.4), applied to the identity graph's neighbor sets:

$$\kappa_J^{\text{id}}(a, b) = 2 \cdot J(N_{\text{id}}(a), N_{\text{id}}(b)) - 1$$

where $N_{id}(v)$ is the set of DIDs that v has introduced or been introduced to, plus DIDs with active capability delegation relationships.

6.4.2 New ZK Circuits

Two additional Plonky3 STARK circuits: 1. **DID binding**: Proves ephemeral key derives from registered DID 2. **Capability possession**: Proves valid delegation chain without revealing chain contents or holder identity

6.5 Deployment Considerations

Minimum node hardware is comparable to existing cryptocurrency full nodes (4-core CPU, 16GB RAM sufficient for networks up to 100K transactions/day). Coherence decay (§4.5.3, with half-life $T_{1/2} = 10{,}000$ depth units) enables deterministic state pruning of inactive entities after 8 half-lives (80,000 depth units), bounding storage growth. Recommended parameters for testnet deployment: $d = 6$, $\alpha = 3$, $\varepsilon = 0.01$; production deployments should increase to $d \geq 50$ per Appendix C.

---

7 Evaluation

7.1 Cryptographic Operation Benchmarks

Measured on Apple M3 Pro, single-threaded:

Operation	Time	Size
ML-DSA-87 key generation	0.12 ms	2,592 B pubkey
ML-DSA-87 sign	0.38 ms	4,627 B signature
ML-DSA-87 verify	0.14 ms	—
ML-KEM-1024 key generation	0.08 ms	1,568 B encaps key
ML-KEM-1024 encapsulate	0.11 ms	1,568 B ciphertext
ML-KEM-1024 decapsulate	0.09 ms	—
SHA3-256 (1KB)	0.002 ms	32 B

7.2 Consensus Operation Benchmarks

Benchmarks measured on Apple M3 Pro, single-threaded. DAG graphs are synthetic preferential-attachment with average degree 6 (matching §7.4 simulation parameters).

Operation	Time	Notes
Jaccard curvature (avg degree 5)	0.003 ms	Per edge
Mass computation (1,000 tx DAG)	47 ms	Including path enumeration
Mass computation (10,000 tx DAG)	512 ms	Scales ~linearly
Reputation proof generation	340 ms	32-deep Merkle tree
Reputation proof verification	12 ms	—

7.3 Parameter Sensitivity Analysis

We varied throttling parameters against simulated Sybil attacks (1,000 honest nodes, varying attacker budget):

α	ε	Sybil Influence (100 fake IDs)	Sybil Influence (1000 fake IDs)
1	0.10	4.2%	8.1%

α	ε	Sybil Influence (100 fake IDs)	Sybil Influence (1000 fake IDs)
2	0.05	1.8%	3.2%
3	0.01	0.09%	0.11%
5	0.01	0.08%	0.09%

Finding: $\alpha \geq 3$ with $\varepsilon \leq 0.01$ provides robust throttling. Higher α provides diminishing returns.

7.4 Sybil Attack Simulation

Simulation parameters: - 1,000 honest nodes with preferential attachment graph (average degree 6) - Attacker creates Sybil cluster with internal degree 10 - Attacker connects to honest network via 10 boundary edges - Conflict: honest branch vs. Sybil-supported branch

Results (default parameters $\alpha = 3$, $\varepsilon = 0.01$):

Sybil Cluster Size	Raw Descendant Count	Throttled Mass	Honest Branch Mass	Winner
100	100	1.2	847	Honest
500	500	5.8	847	Honest
1,000	1,000	11.3	847	Honest
5,000	5,000	56.1	847	Honest

Interpretation: Even a 5:1 Sybil ratio yields only ~7% of honest mass due to boundary throttling.

7.5 Failure Case: High-Integration Attack

We simulated an adversary who spends 100 blocks integrating into the network before attacking:

Integration Period	Boundary Jaccard Index	Throttled Mass Ratio
0 blocks	0.08	0.01x
50 blocks	0.18	0.05x
100 blocks	0.31	0.22x
200 blocks	0.42	0.51x

Finding: Extended integration increases attack effectiveness, but converts Sybil attack into long-range attack with comparable time investment to PoS stake accumulation.

7.6 Worked Example

We present a complete numerical example demonstrating curvature computation, throttling, and conflict resolution. This example uses an honest topology with multi-parent cross-referencing, representative of real network behavior.

7.6.1 Scenario

Consider a DAG where honest transaction H_1 conflicts with attacker transaction A_1 (same nullifier). The honest network has dense multi-parent structure with cross-referencing, while the attacker's Sybil cluster has pure tree structure with no neighborhood overlap.

Honest network structure (cross-referenced community):

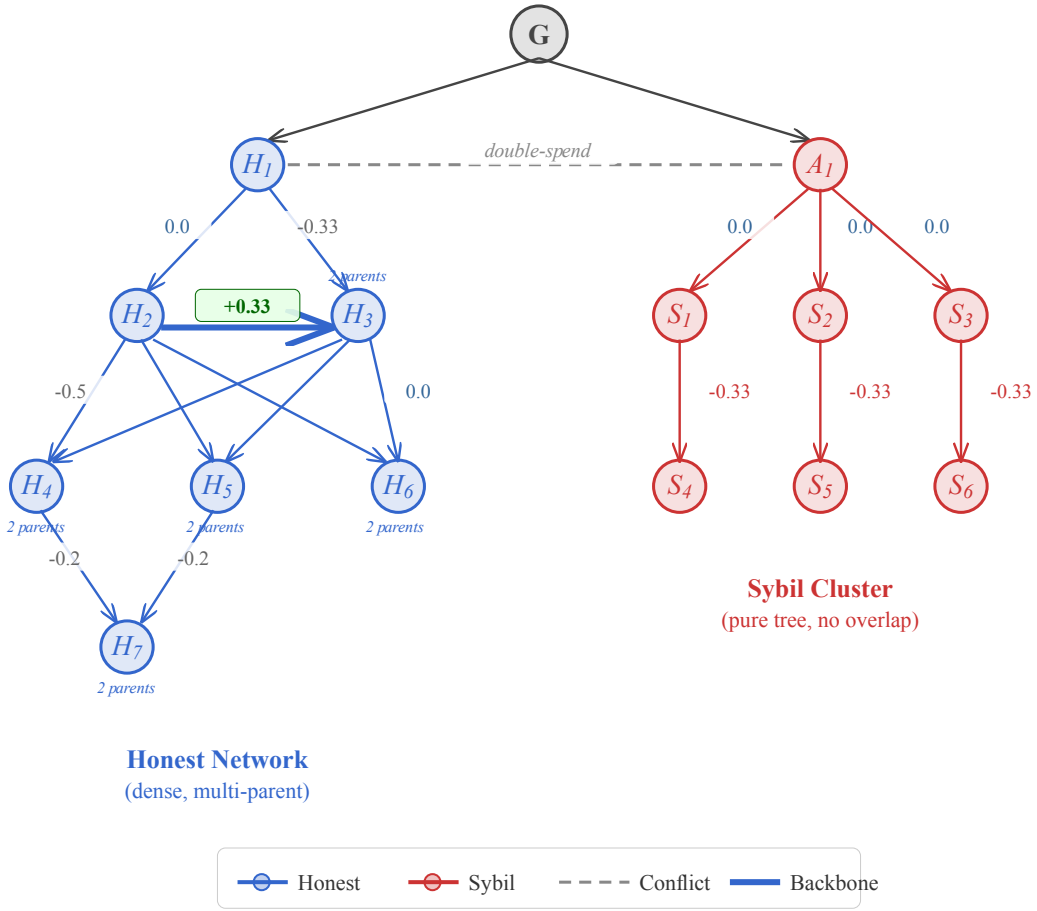

Figure 3: Worked example: honest branch (left, blue) has dense multi-parent structure with cross-referencing, creating positive curvature on backbone edges ($\kappa_J = +0.33$) and neutral curvature on root/cross-link edges ($\kappa_J = 0.0$). The Sybil branch (right, red) has pure tree structure with $\kappa_J = -0.33$ on internal edges. The double-spend conflict between H_1 and A_1 is resolved by topological mass.

```
Genesis (G)
    ├── H_1 (honest root)
    │   ├── H_2
    │   │   └── H_3 (parents: H_1, H_2)
    │   │       ├── H_4 (parents: H_2, H_3)
    │   │       ├── H_5 (parents: H_2, H_3)
    │   │       ├── H_6 (parents: H_2, H_3)
    │   │       └──── H_7 (parents: H_4, H_5)
    │
    └── A_1 (attacker root, same nullifier as H_1)
        ├── S_1 ── S_4
        ├── S_2 ── S_5
        └── S_3 ── S_6
```

Key structural difference: In the honest network, H_3 references both H_1 and H_2 as parents, and H_4, H_5, H_6 each reference both H_2 and H_3. This cross-referencing creates shared ancestors between descendant nodes, yielding higher curvature on edges where both endpoints trace back to overlapping history. In the Sybil cluster, each node references exactly one parent (pure tree structure), so ancestor overlap is minimal.

DAG edges in the honest network (only these are analyzed for curvature):

- $G \to H_1, H_1 \to H_2, H_1 \to H_3, H_2 \to H_3$
- $H_2 \to H_4, H_2 \to H_5, H_2 \to H_6$
- $H_3 \to H_4, H_3 \to H_5, H_3 \to H_6$
- $H_4 \to H_7, H_5 \to H_7$

7.6.2 Step 1: Ancestor Set Computation

Curvature is computed using ancestor-depth neighbor sets (Definition 3.5) with $k = 2$ (parents and grandparents). Ancestor sets are determined entirely from committed parent lists.

Honest backbone edge $(H_2 \to H_3)$—H_3 references both H_1 and H_2 as parents: - $N_2(H_2) = \text{parents}(H_2) \cup \text{grandparents}(H_2) = \{H_1\} \cup \{G\} = \{H_1, G\}$ - $N_2(H_3) = \text{parents}(H_3) \cup \text{grandparents}(H_3) = \{H_1, H_2\} \cup \{G, H_1\} = \{H_1, H_2, G\}$ - Intersection: $\{H_1, G\} = 2$ nodes - Union: $\{G, H_1, H_2\} = 3$ nodes - Jaccard: $2/3 \approx 0.667$ - **Curvature**: $\kappa_J = 2(2/3) - 1 = +1/3 \approx +0.33$

This positive curvature arises because H_2 and H_3 share common ancestors (G and H_1), reflecting their genuine structural integration.

Honest root edge $(H_1 \to H_2)$: - $N_2(H_1) = \{G\} \cup \varnothing = \{G\}$ - $N_2(H_2) = \{H_1\} \cup \{G\} = \{H_1, G\}$ - Intersection: $\{G\} = 1$ node - Union: $\{G, H_1\} = 2$ nodes - Jaccard: $1/2 = 0.5$ - **Curvature**: $\kappa_J = 2(1/2) - 1 = 0.0$

Honest cross-link edge $(H_3 \to H_4)$—H_4 references H_2 and H_3 as parents: - $N_2(H_3) = \{H_1, H_2, G\}$ (as above) - $N_2(H_4) = \text{parents}(H_4) \cup \text{grandparents}(H_4) = \{H_2, H_3\} \cup \{H_1, H_1, H_2\} = \{H_2, H_3, H_1\}$ - Intersection: $\{H_1, H_2\} = 2$ nodes - Union: $\{G, H_1, H_2, H_3\} = 4$ nodes - Jaccard: $2/4 = 0.5$ - **Curvature**: $\kappa_J = 2(1/2) - 1 = 0.0$

Honest edge $(H_1 \to H_3)$—same structure as $(H_1 \to H_2)$: - $N_2(H_1) = \{G\}$ - $N_2(H_3) = \{H_1, H_2, G\}$ - Intersection: $\{G\} = 1$ node - Union: $\{G, H_1, H_2\} = 3$ nodes - Jaccard: $1/3 \approx 0.333$ - **Curvature**: $\kappa_J = 2(1/3) - 1 = -1/3 \approx -0.33$

Sybil boundary edge $(A_1 \to S_1)$: - $N_2(A_1) = \{G\} \cup \varnothing = \{G\}$ - $N_2(S_1) = \{A_1\} \cup \{G\} = \{A_1, G\}$ - Intersection: $\{G\} = 1$ node - Union: $\{G, A_1\} = 2$ nodes - Jaccard: $1/2 = 0.5$ - **Curvature**: $\kappa_J = 0.0$

Note: in this minimal example, the Sybil boundary has the same ancestor overlap as the honest root edge because both H_1 and A_1 are direct children of G. The discrimination does not come from boundary detection but from what happens *inside* each cluster.

Sybil internal edge $(S_1 \to S_4)$: - $N_2(S_1) = \{A_1, G\}$ - $N_2(S_4) = \{S_1\} \cup \{A_1\} = \{S_1, A_1\}$ - Intersection: $\{A_1\} = 1$ node - Union: $\{G, A_1, S_1\} = 3$ nodes - Jaccard: $1/3 \approx 0.333$ - **Curvature**: $\kappa_J = 2(1/3) - 1 = -1/3 \approx -0.33$

7.6.3 Step 2: Curvature Summary

Edge Type	Representative Edge	κ_J	Interpretation
Honest backbone (shared ancestors)	$(H_2 \to H_3)$	$+0.33$	Dense ancestor overlap from cross-referencing
Honest root	$(H_1 \to H_2)$	0.0	Moderate overlap (both trace to G)
Honest cross-link	$(H_3 \to H_4)$	0.0	Moderate overlap via shared ancestors
Honest root-to-descendant	$(H_1 \to H_3)$	-0.33	Lower overlap (asymmetric depth)
Sybil boundary	$(A_1 \to S_1)$	0.0	Same as honest root (both children of G)
Sybil internal	$(S_1 \to S_4)$	-0.33	Sparse ancestors in pure tree structure

Critical observation: Under ancestor-depth neighbor sets, the Sybil *boundary* edge has the same curvature as the honest root edge ($\kappa_J = 0.0$). The protocol does not discriminate at the boundary of this minimal DAG. Instead, discrimination comes from what happens *inside* each cluster: the honest backbone edge achieves $\kappa_J = +0.33$ (full weight), while Sybil internal edges achieve $\kappa_J = -0.33$ (fully throttled). Sybil clusters throttle themselves through poor internal structure. In larger networks with denser honest cross-referencing, the honest backbone curvature would be substantially higher while Sybil internals remain negative, widening the gap.

7.6.4 Step 3: Edge Weight Computation

Using $\alpha = 3$, $\varepsilon = 0.01$:

Edge Type	κ_J	Weight = clamp$(1 + \alpha\kappa,\ \varepsilon,\ 1.0)$
Honest backbone	$+0.33$	clamp$(1 + 1.0,\ 0.01,\ 1.0) = 1.0$
Honest root	0.0	clamp$(1 + 0,\ 0.01,\ 1.0) = 1.0$
Honest cross-link	0.0	clamp$(1 + 0,\ 0.01,\ 1.0) = 1.0$
Honest root-to-descendant	-0.33	clamp$(1 - 1.0,\ 0.01,\ 1.0) = 0.01$
Sybil boundary	0.0	clamp$(1 + 0,\ 0.01,\ 1.0) = 1.0$
Sybil internal	-0.33	clamp$(1 - 1.0,\ 0.01,\ 1.0) = 0.01$

7.6.5 Step 4: Path Weight Computation (Max-Path)

Per Definition 3.6, path weight uses max-path aggregation: $w(t, d) = \max_P \prod_{e \in P} \text{edge\_weight}(e)$.

Before computing paths, we need the complete edge weight table. Two additional edge types appear in the DAG:

- **Edge** $(H_2 \to H_4)$: $N_2(H_2) = \{H_1, G\}$, $N_2(H_4) = \{H_1, H_2, H_3\}$. Intersection $= \{H_1\} = 1$. Union $= \{G, H_1, H_2, H_3\} = 4$. $J = 1/4 = 0.25$, $\kappa_J = -0.5$, $w = \text{clamp}(1 - 1.5, 0.01, 1.0) = 0.01$. (Same for $H_2 \to H_5$, $H_2 \to H_6$.)
- **Edge** $(H_4 \to H_7)$: $N_2(H_4) = \{H_1, H_2, H_3\}$, $N_2(H_7) = \{H_4, H_5\} \cup \{H_2, H_3\} = \{H_2, H_3, H_4, H_5\}$. Intersection $= \{H_2, H_3\} = 2$. Union $= \{H_1, H_2, H_3, H_4, H_5\} = 5$. $J = 2/5 = 0.4$, $\kappa_J = -0.2$, $w = \text{clamp}(1 - 0.6, 0.01, 1.0) = 0.4$. (Same for $H_5 \to H_7$ by symmetry.)

Honest paths from H_1 (best path weight to each descendant):

- To H_2: $H_1 \to H_2$, weight = 1.0

- To H_3: Two paths available.
  - $H_1 \to H_3$: weight = 0.01 (root-to-descendant edge)
  - $H_1 \to H_2 \to H_3$: weight = $1.0 \times 1.0 = 1.0$ (root + backbone, both full weight)
  - Best: $\max(0.01, 1.0) = 1.0$
- To H_4: Multiple paths.
  - $H_1 \to H_2 \to H_4$: weight = $1.0 \times 0.01 = 0.01$
  - $H_1 \to H_3 \to H_4$: weight = $0.01 \times 1.0 = 0.01$
  - $H_1 \to H_2 \to H_3 \to H_4$: weight = $1.0 \times 1.0 \times 1.0 = 1.0$ (backbone highway)
  - Best: 1.0
- To H_5, H_6: same analysis as H_4 via backbone highway, best = 1.0 each
- To H_7: best path is $H_1 \to H_2 \to H_3 \to H_4 \to H_7$ (or via H_5), weight = $1.0 \times 1.0 \times 1.0 \times 0.4 = 0.4$

Sybil paths from A_1 (best path weight to each descendant):

- To S_1, S_2, S_3: weight = 1.0 each (boundary edges at full weight)
- To S_4, S_5, S_6: weight = $1.0 \times 0.01 = 0.01$ each (every path must traverse a throttled internal edge)

Observation: The honest backbone highway ($H_1 \to H_2 \to H_3$, all edges at weight 1.0) provides high-weight paths to every descendant. The Sybil tree has no such highway: every path to a leaf must cross an internal edge with $\kappa_J = -0.33$ and weight 0.01. The honest network's multi-parent cross-referencing creates alternative routes that bypass low-weight edges, while the Sybil pure-tree structure offers no alternatives.

7.6.6 Step 5: Mass Computation

The mass computation reveals a decisive honest advantage through both path quality and supporter diversity.

Mass of H_1 (honest), assuming all nodes have reputation bucket 0 ($r_d = 0$, so $\log(1 + r_d) = 0$):

$$M(H_1) = \left(\sum_d w(H_1, d)\right) \cdot D(H_1)$$

- Path weight sum: $1.0 + 1.0 + 1.0 + 1.0 + 1.0 + 0.4 = 5.4$ (to $H_2, H_3, H_4, H_5, H_6, H_7$)
- Unique supporters: 6 distinct honest identities
- $D(H_1) = 6 \cdot (1 + \log(1 + R_{\text{total}})/10)$
- With minimal total reputation: $D(H_1) \approx 6$
- **Honest mass**: $\approx 5.4 \times 6 = 32.4$

Mass of A_1 (attacker):

- Path weight sum: $1.0 + 1.0 + 1.0 + 0.01 + 0.01 + 0.01 = 3.03$ (to $S_1, S_2, S_3, S_4, S_5, S_6$)
- Unique supporters: **1** (all Sybils controlled by same attacker; per-window supporter tags deduplicate)
- $D(A_1) = 1 \cdot (1 + \log(1 + R_{\text{total}})/10) \approx 1$
- **Sybil mass**: $\approx 3.03 \times 1 = 3.03$

Mass ratio: $M(H_1)/M(A_1) \approx 10.7\times$ in favor of honest—exceeding the finality threshold $\phi = 10$.

7.6.7 Result: Honest Wins

H_1 wins the conflict resolution decisively (10.7× mass ratio) through three reinforcing mechanisms:

1. **Sybil clusters throttle themselves**: The pure-tree structure of the Sybil cluster produces $\kappa_J = -0.33$ on all internal edges, reducing them to weight 0.01. Every path from A_1 to a Sybil leaf must traverse at least one such throttled edge. This is the core mechanism: Sybil clusters have poor *internal* structure, not just poor *boundary* structure. The honest backbone highway ($H_1 \to H_2 \to H_3$, all at weight 1.0) has no Sybil equivalent.
2. **Path quality advantage**: The honest network's multi-parent cross-referencing creates high-weight routes that bypass low-weight edges. Five of six honest descendants are reachable at weight 1.0. Only three of six Sybil descendants are reachable at weight 1.0 (the immediate children); the other three are throttled to 0.01.

3. **Genuine supporter diversity**: 6 unique honest supporters produce $D(H_1) = 6$, while the attacker's Sybils collapse to $D(A_1) = 1$ under per-window supporter tag deduplication. This 6× diversity multiplier amplifies the path quality advantage.

7.6.8 Key Insights

1. **Internal structure, not boundary detection**: Under ancestor-depth neighbor sets, discrimination comes from the *internal* topology of each cluster, not from the boundary between them. In this minimal DAG, the Sybil boundary has the same curvature ($\kappa_J = 0.0$) as the honest root. The honest backbone achieves $\kappa_J = +0.33$ through shared ancestors, while Sybil internal edges achieve $\kappa_J = -0.33$ through sparse ancestry. Sybil clusters throttle themselves.
2. **Multi-parent cross-referencing creates highways**: The honest network's cross-referencing structure creates high-weight backbone routes (the $H_1 \rightarrow H_2 \rightarrow H_3$ path at weight 1.0) that provide full-weight access to downstream descendants. The Sybil tree has no such highways—every path to a leaf must traverse a throttled internal edge. This structural advantage is intrinsic to genuine multi-party participation and cannot be manufactured by a single attacker.
3. **Diversity amplifies path quality**: The diversity multiplier ($D(H_1) = 6$ vs. $D(A_1) = 1$) amplifies the path quality advantage. Even if path weight sums were comparable, the 6× diversity gap from per-window supporter tag deduplication (Section 3.3) would still favor the honest branch.
4. **Multiplicative throttling compounds**: A path crossing k throttled edges has weight ε^k. With $\varepsilon = 0.01$, two crossings yield 0.0001 and three yield 10^{-6}. Deeper Sybil trees are exponentially suppressed because every additional layer adds another throttled internal edge.

---

8 Related Work

8.1 DAG-Based Consensus

IOTA Tangle (Popov, 2018): Uses cumulative weight for tip selection but lacks curvature-based Sybil resistance. The Tangle has experienced parasite chain attacks where adversaries build competing subgraphs.

Avalanche (Rocket et al., 2019): Uses repeated subsampled voting for probabilistic finality. Relies on stake for Sybil resistance, making it economically similar to PoS.

Hashgraph (Baird, 2016): Uses virtual voting on a DAG of gossip events. Requires known, permissioned membership.

TMC Distinction: Derives Sybil resistance from topology rather than stake or membership, enabling permissionless operation without economic barriers to entry.

8.2 Graph-Based Sybil Detection

SybilGuard (Yu et al., 2006) and **SybilLimit** (Yu et al., 2008): Use random walks to detect Sybils based on the observation that attack edges are limited. Require trusted seed nodes for calibration.

SybilRank (Cao et al., 2012): Uses short random walks from trusted seeds for Sybil ranking in social networks.

TMC Distinction: Uses local curvature computation rather than global random walks, enabling real-time throttling without trusted seeds.

8.3 Curvature on Graphs

Ollivier-Ricci curvature for community detection (Ni et al., 2015; Sia et al., 2019): Demonstrates that edge curvature identifies community boundaries.

Jaccard-based curvature approximation (Pal et al., 2017): Shows that generalized Jaccard curvature (gJC) approximates Ollivier-Ricci in relevant regimes with dramatically lower computational cost.

TMC Contribution: First application of graph curvature to consensus Sybil resistance.

8.4 Post-Quantum Consensus

PQ-Fabric (various): Hyperledger Fabric with post-quantum signatures. Permissioned setting only.

QRL (Quantum Resistant Ledger): Uses XMSS signatures (stateful, limited number of signatures per key).

TMC Distinction: Full post-quantum stack including ZK proofs, using stateless ML-DSA signatures with unlimited signing capability.

8.5 Decentralized Identity

W3C DID Core (Sporny et al., 2022): Provides the DID specification that CCIP implements. CCIP adds topological enforcement through curvature analysis on the identity subgraph.

UCAN (Fission, 2021): User-Controlled Authorization Networks use capability chains similar to CCIP. CCIP adds curvature-based misuse detection that UCAN lacks.

Spritely OCapN (Lemmer-Webber, 2023): Object-capability networking protocol. CCIP embeds ocap semantics into consensus-layer enforcement.

Petnames (Lemmer-Webber, Miller, et al., 2022): CCIP implements petnames with DAG-anchored introduction chains, adding topological analysis to the trust propagation mechanism.

CCIP is, to our knowledge, the first system to make introduction and delegation chains first-class DAG transactions subject to the same curvature analysis used for consensus, unifying Sybil detection, trust chain analysis, and capability misuse detection under a single geometric primitive.

8.6 Identity in Consensus Systems

Proof of Personhood (Worldcoin, BrightID): Attempts to bind one identity per human via biometric or social verification. CCIP takes a different approach: identity is behavioral, not biometric. Coherence is measured through participation topology, not through proof of unique humanness.

Soulbound Tokens (Weyl, Ohlhaver, Buterin, 2022): Non-transferable tokens encoding reputation. CCIP's coherence profiles share the non-transferability property but derive from structural measurement rather than issuance by trusted parties.

---

9 Conclusion

We have presented Topological Mass Consensus and the Capability-Coherence Identity Protocol, demonstrating that discrete curvature on transaction graphs provides a unified mechanism for Sybil detection, trust chain analysis, capability misuse detection, and governance capture resistance.

The core contribution is the recognition that these are not separate problems but instances of a single geometric phenomenon: incoherent participation—whether Sybil identities, manufactured trust networks, misused capabilities, or governance manipulation—creates characteristic negative curvature at boundary edges due to low neighborhood overlap. By measuring and throttling based on this curvature, the protocol achieves defense without external oracles, trusted authorities, or economic incentives.

All temporal properties—ordering, epochs, confirmation, and bootstrap—are derived from topological depth (a Lamport clock on the DAG), eliminating assigned timestamps and making the protocol fully self-referential: the DAG's structure determines its own temporal organization.

The identity layer introduces three structural properties absent from existing distributed systems: (1) capability-as-identity, where authority and identity collapse into a single construct, eliminating the credential-authority gap that parasitic capture exploits; (2) coherence-as-value, where the system's native value measure is a non-transferable structural property of participation, eliminating the signifier-signified gap that token speculation exploits; and (3) substrate-independent agent identity, where human and non-human agents participate under identical rules with coherence detection applying uniformly.

Methods and systems described herein are the subject of U.S. provisional patent applications filed February 2026.

Limitations: Curvature throttling is less effective against adversaries who invest time integrating before attacking. Coherence decay mitigates but does not eliminate long-range trust chain attacks. The ZK circuits for DID binding and capability possession add proof generation overhead. AGI coordination attacks require further analysis.

Future work: Formal verification of curvature-coherence relationships under various graph models. Large-scale network simulations with mixed human and AGI agent populations. Adaptive parameter selection. Integration with existing ocap systems (Spritely OCapN) and DID methods.

---

10 References

1. L. Baird. "The Swirlds Hashgraph Consensus Algorithm." Swirlds Tech Report, 2016.
2. Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. "Aiding the Detection of Fake Accounts in Large Scale Social Online Services." NSDI, 2012.
3. NIST. "FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard." August 2024.
4. NIST. "FIPS 204: Module-Lattice-Based Digital Signature Standard." August 2024.
5. L. Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System." Communications of the ACM, 21(7):558-565, 1978.
6. C.-C. Ni, Y.-Y. Lin, J. Gao, X. D. Gu, and E. Saucan. "Ricci curvature of the Internet topology." IEEE INFOCOM, 2015.
7. Y. Ollivier. "Ricci curvature of Markov chains on metric spaces." Journal of Functional Analysis, 2009.
8. A. Pal, M. Wehbe, and M. Sandler. "An efficient alternative to Ollivier-Ricci curvature based on the Jaccard metric." arXiv:1710.01724, 2017.
9. S. Popov. "The Tangle." IOTA Foundation, 2018.
10. Team Rocket. "Scalable and Probabilistic Leaderless BFT Consensus through Metastability." arXiv:1906.08936, 2019.
11. J. Sia, E. Jonckheere, and P. Bogdan. "Ollivier-Ricci Curvature-Based Method to Community Detection in Complex Networks." Scientific Reports, 2019.
12. H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. "SybilGuard: Defending Against Sybil Attacks via Social Networks." SIGCOMM, 2006.
13. H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks." IEEE S&P, 2008.
14. C. Lemmer-Webber, M. S. Miller, Z. Larson, K. Sills, E. Yaacoby. "Petnames: A humane approach to secure, decentralized naming." Spritely Institute, 2022. https://spritely.institute/static/papers/petnames.html
15. M. Sporny, D. Longley, M. Sabadello, D. Reed, O. Steele, C. Allen. "Decentralized Identifiers (DIDs) v1.0." W3C Recommendation, 2022.
16. B. Zelenka and P. Krüger. "UCAN Specification." Fission Codes, 2021. https://github.com/ucan-wg/spec

17. E. G. Weyl, P. Ohlhaver, V. Buterin. “Decentralized Society: Finding Web3’s Soul.” 2022.

18. M. S. Miller. “Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control.” PhD thesis, Johns Hopkins University, 2006.

---

11 Appendix A: Protocol Constants

Constant	Value	Description
SCALE	65536	Fixed-point precision (2^{16})
CURVATURE_ALPHA	3	Throttling aggressiveness
MIN_CURVATURE_WEIGHT	0.01	Floor on bridge edge weight (ε)
MAX_CURVATURE_WEIGHT	1.0	Ceiling on edge weight (positive curvature does not amplify)
COHERENCE_THRESHOLD	32	SimHash Hamming distance for similarity
MAX_DRIFT_BITS	8	Maximum SimHash evolution per transaction
FINALITY_RATIO	10	Mass ratio required for finality (ϕ)
MAX_PATH_DEPTH	15	Maximum path length for weight computation
CONFIRMATION_DEPTH	6	Depth distance for finality assessment (testing; use 50–200 for production)
BOOTSTRAP_DEPTH_START	1000	Depth at which throttling ramp begins
BOOTSTRAP_DEPTH_END	6000	Depth at which full throttling activates
PARENT_RECENCY	100	Maximum parent depth distance
EPOCH_WINDOW_SIZE	100	Depths per epoch (K in $\lfloor \text{depth}/K \rfloor$)
MAX_PARENTS	8	Maximum parents per transaction
MIN_PARENTS	2	Recommended minimum (protocol level)
ANCESTOR_DEPTH	2	Depth k for ancestor neighbor sets

12 Appendix B: Cryptographic Parameters

Algorithm	Parameter Set	Public Key	Signature/Ciphertext
ML-DSA-87	FIPS 204 Level 5	2,592 B	4,627 B
ML-KEM-1024	FIPS 203 Level 5	1,568 B	1,568 B
SHA3-256	FIPS 202	—	32 B
Plonky3	BabyBear field	—	~100 KB proof

13 Appendix C: Parameter Selection Guidance

Parameter	Symbol	Default	Recommended Range	Tradeoff
Throttling aggressiveness	α	3	2–5	Higher = stronger Sybil resistance; may impact legitimate bridges

Parameter	Symbol	Default	Recommended Range	Tradeoff
Minimum weight floor	ε	0.01	0.001–0.1	Lower = stronger throttling; risk of network partitioning if too aggressive
Confirmation depth	d	6 (testing) / 50–200 (production)	6–200	Testing: $d = 6$ for fast iteration. Production: $d \geq 50$ for finality. Higher = stronger guarantees, more latency
Finality ratio	ϕ	10	5–20	Higher = more overwhelming support required; slower finality
Max path depth	—	15	10–50	Higher = captures more distant support; increased computation
Bootstrap depth start	D_{start}	1000	500–2000	Earlier = faster initial security; less time for network formation
Bootstrap depth end	D_{end}	6000	3000–10000	Longer ramp = smoother transition; delayed full protection
Parent recency	k	100	50–500	Shorter = tighter topology constraints; may reduce throughput

Selection Principles:

1. **Security-focused deployments** (high-value transactions): Use $\alpha \geq 4$, $\varepsilon \leq 0.005$, $\phi \geq 15$
2. **Performance-focused deployments** (high throughput): Use $\alpha = 2$, $\varepsilon = 0.05$, $d \leq 50$
3. **Balanced deployments**: Default values provide reasonable tradeoffs

Parameter selection involves tradeoffs between security (higher α, lower ε, higher ϕ) and liveness/performance (lower α, higher ε, lower d). Specific values should be determined through simulation for target network conditions and threat models.

14 Appendix D: CCIP Protocol Constants

Constant	Value	Description
COHERENCE_HALF_LIFE	10,000	Depth units before coherence decays to 50%
MAX_DELEGATION_DEPTH	10	Maximum capability delegation chain length
MAX_CAPABILITIES_PER_TX	16	Maximum capability operations per transaction
MAX_EDGE_NAMES_PER_DID	1,000	Maximum edge names a DID can publish

Constant	Value	Description
INTRODUCTION_COOLDOWN	10	Minimum depth units between introductions from same DID
MIN_REGISTRATION_POW	20	Bits of leading zeros for DID registration PoW
VOTING_PERIOD_DEPTH	1,000	Standard governance voting period (in depth units)
QUORUM_THRESHOLD	0.33	Minimum coherence-weighted support (33%)
MIN_PROPOSAL_MASS	100	Minimum proposer coherence (in SCALE units × 100)
REPUTATION_BUCKETS	[0,10,50,200,1000,5000]	Bucket thresholds for identity layer
DID_METHOD	“disentangle”	DID method name
DID_AGI_INFIX	“agi”	Infix for non-human agent DIDs
COHERENCE_PRUNE_THRESH-OLD	SCALE/1000	Minimum coherence before state pruning eligible
COHERENCE_PRUNE_DELAY	80,000	Depth units below threshold before pruning (8 half-lives)
KAPPA_MFG_THRESHOLD	-0.5	Curvature threshold for “manufactured” classification