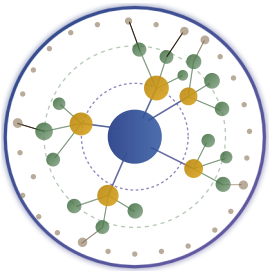


The Immanent Trust Protocol

A Post-Consensus Architecture for Decentralized, Observer-Relative Trust

Version: 1.0 — April 2026

Authors: Global Governance Frameworks

Abstract

Decentralized trust and coordination have long been constrained by a false dichotomy: either rely on centralized authorities or enforce global consensus across all nodes. Both approaches violate Ashby's Law of Requisite Variety, forcing the infinite complexity of human and ecological interaction through narrow computational bottlenecks. We present the **Immanent Trust Protocol (ITP)**, a post-consensus architecture where trust is computed at the edge, relative to each observer, without any global ledger or universal reputation score. ITP introduces the **Informal Trust Ledger (ITL)** – a directed, weighted graph of cryptographically signed **Trust Signals** that nodes gossip and evaluate locally. To resist Sybil and collusion attacks, we define the **Trust Independence Metric (TIM)**, which discounts endorsements from topologically overlapping clusters, mathematically collapsing large collusion rings to the weight of a single node. **Trust Shock Propagation** gives endorsers "skin in the game" by cascading penalties when a trusted node defects, while an **appeal mechanism** and **calibration histories** defend against false accusations. For cross-scale disturbances, ITP forms ephemeral **Scale-Adaptive Fields** using a rotating coordinator lottery, avoiding permanent bureaucratic structures. We provide a formal theorem bounding TIM's Sybil collapse, a game-theoretic analysis of honest signaling, simulation results, and a reference implementation architecture with post-quantum cryptographic agility. ITP offers a viable substrate for bioregional governance and autonomous agent coordination without surrendering sovereignty to a global ledger.

"The most powerful system is not one that governs the network, but one that is present within every connection."

1. Introduction

For the past decade, decentralized technology has pursued a compelling vision: replace central authorities with networks where no single party controls the truth. Blockchains, distributed ledgers, and peer-to-peer reputation systems have sought to enable trust without hierarchy. Yet despite tremendous progress, these systems remain structurally incapable of scaling to the complexity of human and ecological coordination. The fundamental reason is not a lack of throughput or finality—it is a violation of **Ashby's Law of Requisite Variety** [1]. A regulator can only control a system if it possesses at least as much variety (complexity) as the system it regulates. Global consensus mechanisms, whether Proof-of-Work or Byzantine fault tolerance, force every node to agree on a single, universal state. This state is a low-fidelity approximation of reality, aggregated from distant events, delayed by propagation, and stripped of local context. In attempting to eliminate the central *actor*, these systems have retained a centralized *ontology*: a single truth that all must accept.

The consequences are well documented. Decentralized reputation systems (e.g., EigenTrust [2], PageRank-based trust) collapse under Sybil attacks or collusion rings unless they introduce external identity or financial stake. Blockchain-based governance (DAOs) suffers from vote buying, plutocracy, and low participation. Even modern "web of trust" protocols (e.g., Nostr, Holochain) either lack formal collusion resistance or require manual curation. None provide a mathematically grounded mechanism to distinguish a thousand independent voices from a thousand Sybils orchestrated by a single actor.

This paper introduces the **Immanent Trust Protocol (ITP)** – a post-consensus architecture that abandons the quest for a global truth. Instead, ITP treats trust as **observer-relative**, computed continuously at the edge using only local information. Every node maintains its own **Informal Trust Ledger (ITL)**, a subjective graph of cryptographically signed **Trust Signals** that decay over time and respect a distance-attenuation multiplier (e.g., trust from a friend-of-a-friend is weighted half as much as direct experience). There is no mandatory global state; nodes gossip signals and evaluate them locally.

The core technical contribution of ITP is the **Trust Independence Metric (TIM)**. TIM analyzes the topological overlap among endorsers of a target node. When endorsers share nearly identical neighborhoods or exhibit correlated signaling, TIM discounts their collective weight. We prove that for a collusion ring of size S with average pairwise Jaccard similarity J, the effective weight after TIM is bounded by $1 + (1 - J)(S - 1)$ times the weight of a single honest node. When $J \to 1$ (perfect overlap), the entire ring collapses to the influence of a single node—rendering Sybil attacks and collusion rings economically irrational.

To align incentives, ITP introduces **Trust Shock Propagation**: if a node defects (e.g., steals resources), the victim issues a `constraint` signal. This signal not only penalizes the defector but also cascades backward to its endorsers, slashing their trust weight. Endorsers thus have "skin in the game." To prevent weaponized false accusations, shock propagation requires a threshold of independent signals (verified by TIM) and includes an appeal mechanism with cryptographic proof. A game-theoretic analysis shows that honest, high-confidence signaling forms a Nash equilibrium under these rules.

For disturbances that exceed local regulatory capacity (e.g., pandemics, financial contagion), ITP forms ephemeral **Scale-Adaptive Fields** – temporary macro-nodes that coordinate via a rotating coordinator lottery (probability proportional to local trust). Crucially, each field's authority is cryptographically bound to a time-to-live (TTL); it auto-dissolves unless actively renewed. This provides multi-scale coordination without permanent bureaucratic structures.

We also address practical deployment concerns often overlooked in theoretical papers: context ontology resolution (how nodes agree on signal domains), state retention under node churn, resilience to network partitions, and cryptographic agility for post-quantum readiness (using multicodec signature-suite prefixes).

The remainder of this paper is organized as follows. Section 2 provides cybernetic foundations and a threat model. Section 3 describes the Informal Trust Ledger and Trust Signals. Section 4 defines TIM, presents the Sybil collapse theorem, and outlines simulation results. Section 5 covers Trust Shock Propagation and game-theoretic equilibrium. Section 6 addresses bootstrapping and cold-start. Section 7 introduces Scale-Adaptive Fields. Section 8 discusses privacy and traceability trade-offs. Section 9 details the reference implementation architecture. Section 10 provides a comprehensive security analysis. Section 11 presents full simulation results. Section 12 discusses limitations and future work. Section 13 concludes. Proofs, pseudocode, and parameter tables appear in the appendices.

---

2. Cybernetic Foundations & Related Work

2.1 Ashby's Law of Requisite Variety and Its Implications for Distributed Systems

In 1956, the cybernetician W. Ross Ashby formulated a principle that would become foundational for understanding regulation, control, and coordination in complex systems. The **Law of Requisite Variety** states: *"Only variety can destroy variety"* [1]. More formally, a regulator (or control system) can only successfully regulate a target system if it possesses at least as much variety (i.e., number of distinguishable states or behavioral responses) as the target system it seeks to control. If the target system can exhibit V_t distinct states, the regulator must be able to adopt at least V_t distinct responses; otherwise, the target will inevitably exhibit states that the regulator cannot counter, leading to loss of control.

Ashby derived this law from the mathematical theory of communication and cybernetics, showing that it is a logical necessity, not merely a heuristic. The law holds regardless of whether the regulator is a biological nervous system, a government bureaucracy, a market mechanism, or a digital protocol.

2.1.1 The Variety Bottleneck in Hierarchical and Consensus Systems

Traditional human institutions (governments, corporations, international bodies) are **hierarchical**: they aggregate information upward, process it at a center, and issue commands downward. This structure creates a fundamental variety bottleneck. No finite hierarchy can match the combinatorial explosion of states arising from the interactions of millions of individuals, local ecologies, and rapidly changing conditions. As Stafford Beer, a pioneer of management cybernetics, noted, hierarchical regulation inevitably leads to *information loss* and *control delay* [2]. The center receives a low-fidelity, aggregated snapshot of reality, makes decisions based on outdated data, and issues commands that are already maladapted to the situation at the periphery.

First-generation decentralized systems (blockchains, distributed ledgers) promised to eliminate the central *actor*, but they retained a centralized *ontology*. Every node must agree on a single, universal state of truth—whether that state is a transaction ledger, a reputation score, or a governance proposal. This global consensus mechanism imposes an even stricter variety constraint than hierarchy. Not only does information flow through a narrow computational bottleneck (the consensus algorithm), but the requirement for *global agreement* forces the network to operate at the speed of its slowest, most constrained participants. The variety of the real world—human intentions, ecological fluctuations, resource availabilities—is effectively infinite. A consensus protocol that must reduce that infinite variety to a single bitstring of finite length is, by Ashby's Law, guaranteed to be an inadequate regulator.

2.1.2 Distance, Information Flow, and the Impossibility of Global State

Ashby's Law interacts critically with another cybernetic constraint: **information cannot flow faster than the physical medium allows**, and **any regulator at a distance greater than zero from an event must rely on delayed, low-fidelity signals**. In a planetary-scale network, the speed of light imposes a minimum latency. More importantly, the *cost of acquiring high-variety information* grows with distance. To know the exact state of every node in a global network would require an infinite communication bandwidth and infinite storage—clearly impossible.

Thus, any system that attempts to maintain a *global state* (a single ledger of truth) is forced to discard nearly all of the variety present in the system. It retains only what can be aggregated, compressed, and agreed upon by a majority. This discarded variety does not disappear; it manifests as **unregulated disturbances**—hacks, collusion, front-running, governance attacks, and other forms of systemic instability that plague blockchain-based systems.

2.1.3 Immanent Regulation: The Distance-Zero Alternative

If a regulator cannot be at a distance greater than zero and still match the target's variety, the only logical conclusion is that the regulator must be **at distance zero** from the events it regulates. That is, regulation must be *immanent*—embedded within the very interactions that constitute the system, not imposed from outside or above.

In a distributed network, immanent regulation means:

- Every node regulates its own interactions based on locally available information.
- No node needs to know the global state; it only needs to evaluate the nodes it directly interacts with.
- Trust is computed on the fly, using only the node's own history and the signals of its immediate, trusted neighbors.
- Coordination emerges from the bottom up, not from a central plan or a global consensus.

This is the immanent coordination principle: regulatory capacity must be embedded at the point of interaction, not aggregated to a centre.

This does *not* imply anarchy or the absence of macro-level coordination. Rather, it implies that macro-level structures (such as the Scale-Adaptive Fields introduced in Section 7) must be *ephemeral*, formed on demand by local nodes, and dissolved when no longer needed. Permanent global institutions are, by Ashby's Law, guaranteed to become brittle and unresponsive.

2.1.4 The Variety of Trust

Trust is a particularly instructive example. Trust is fundamentally *relational* and *contextual*. Whether I trust a person to lend me money, to drive a car, or to write software code are entirely different questions. Trust also *decays* over time and is mediated by social distance. A global "reputation score" that aggregates all contexts and all relationships into a single number is not only meaningless—it is dangerous. It creates a false sense of objectivity and invites manipulation.

ITP operationalizes immanent regulation by treating trust as a **local, observer-relative, context-bound inference**. The protocol provides cryptographic tools (Trust Signals) and a mathematical framework (TIM, shock propagation) for nodes to compute their own trust assessments without appealing to any global authority. This design is the direct consequence of taking Ashby's Law seriously.

2.1.5 Summary

Ashby's Law forces a radical conclusion: no global consensus system can adequately regulate a planetary-scale socio-ecological system. The only viable path is to push regulation to the edge, making each node a regulator of its own interactions, using only local information and ephemeral coordination structures. The remainder of this paper describes how cryptographic primitives and graph-theoretic metrics can realize this immanent vision.

2.2 Related Work: Why Existing Decentralized Trust Systems Fall Short

Numerous systems have attempted to enable decentralized trust without a central authority. They can be broadly categorized into reputation-based systems, web-of-trust models, blockchain-based identity, and recent peer-to-peer protocols. Each fails one or more of the requirements derived from Ashby's Law: **global consensus bottleneck**, **lack of Sybil/collusion resistance**, **context insensitivity**, or **inability to scale variety**.

2.2.1 Centralized Reputation Systems (EigenTrust, PageRank)

EigenTrust [3] and its descendants (e.g., PeerTrust, PowerTrust) were designed for peer-to-peer file sharing. They aggregate local trust scores into a global trust value via iterative matrix multiplication, essentially computing a global consensus on trust. While mathematically elegant, they suffer from two fatal flaws:

- **Global state requirement**: Every node must eventually converge to the same trust scores, which requires global communication and synchronization. This violates Ashby's Law by forcing the network to agree on a single low-dimensional representation of trust.
- **Vulnerability to collusion**: EigenTrust assumes that a set of pre-trusted "seed" nodes can bootstrap the system. In practice, colluding nodes can artificially boost each other's scores. The protocol has no analog of TIM; it cannot distinguish independent endorsements from Sybil rings.

PageRank [4], originally designed for web search, has been repurposed for trust (e.g., in Google's PageRank for reputation). It propagates "authority" along links, but it is notoriously susceptible to link farms – precisely the collusion pattern that TIM detects via neighborhood overlap. Moreover, PageRank is a global computation, requiring a centralized crawler or a distributed consensus that reintroduces the variety bottleneck.

2.2.2 Web of Trust (PGP/GnuPG, OpenPGP)

The classic Web of Trust (WoT) used in PGP allows users to sign each other's keys, creating a graph of endorsements. Trust is computed locally by following certification paths. This avoids global consensus, making it *immanent* in spirit.

However, WoT has major practical limitations that ITP addresses:

- **No automatic collusion resistance**: A Sybil cluster can cross-sign each other's keys, creating a dense subgraph that appears trustworthy to a naive path-finding algorithm. WoT lacks a TIM-like discount for structural overlap.
- **Manual management**: Users must explicitly choose which signatures to trust; there is no algorithmic computation of a Local Trust State. This does not scale to millions of agents or autonomous AI nodes.
- **No decay or context**: Signatures are permanent and context-free. A signature from 2005 still counts today, and there is no distinction between "trust to encrypt email" and "trust to access a water reservoir."

2.2.3 Blockchain-Based Identity and Reputation (Ethereum, BrightID, Proof-of-Personhood)

Blockchains offer a global, immutable ledger – the very thing ITP rejects. Projects like BrightID [5] and Proof-of-Personhood (PoP) attempt to prevent Sybils by linking each real human to one identity, often using social graphs or in-person ceremonies. These methods are either:

- **Centralized** (a central authority verifies uniqueness), or
- **Require global consensus** (a smart contract maintains a set of verified identities), or
- **Privacy-invasive** (social graph analysis reveals relationships).

More importantly, they solve only the Sybil problem, not the collusion problem. A coalition of real humans can still cross-endorse each other artificially. And by anchoring to a global ledger, they inherit the variety bottleneck: the entire network must agree on who is a unique person, a decision that cannot capture local context or changing trustworthiness.

2.2.4 Nostr (Notes and Other Stuff Transmitted by Relays)

Nostr [6] is a simple, open protocol for decentralized social networking. It uses relays to broadcast events (notes, follows, reactions) signed by public keys. Clients can compute their own view of the network by querying multiple relays. This is *observer-relative* and does not require global consensus.

Nostr's trust model is minimal: a user can choose to follow certain public keys. However, Nostr provides no built-in Sybil or collusion resistance. A malicious actor can spin up thousands of keys, follow each other, and flood relays with seemingly credible content. Users must manually curate their follow lists or rely on external moderation – which reintroduces centralization. ITP could be seen as adding an algorithmic trust layer *on top of* a Nostr-like relay infrastructure, providing TIM and shock propagation.

2.2.5 Holochain

Holochain [7] is a framework for agent-centric, distributed applications. Each agent maintains its own source chain, and validation rules are defined by the application. It explicitly rejects global consensus, embracing subjective, observer-relative state. In that sense, it is philosophically aligned with ITP.

However, Holochain's default trust mechanisms are application-specific; there is no general-purpose Sybil-resistant trust metric. Developers must implement their own reputation logic, often leading to ad-hoc solutions vulnerable to collusion. Holochain also does not provide a built-in TIM or shock propagation. ITP can be seen as a specialized trust layer that could run *on top* of Holochain's agent architecture, or as a stand-alone protocol with a more formal mathematical foundation.

2.2.6 Summary of Deficiencies

System	Global State?	Collusion Resistance?	Context-Aware?	Decay?	Immanent?
EigenTrust	Yes	Weak	No	No	No
PageRank	Yes	Weak	No	No	No
Web of Trust (PGP)	No	None (manual)	No	No	Partial
Blockchain Reputation	Yes	Via stake/identity	Partial	Sometimes	No
Nostr	No	None (manual)	No	No	Yes
Holochain	No	Application-defined	Application	App-defined	Yes

System	Global State?	Collusion Resistance?	Context-Aware?	Decay?	Immanent?
ITP (this work)	**No**	**TIM + shock propagation**	**Yes (context field)**	**Yes (expiry)**	**Yes**

None of the existing systems provide a *general, automatic, mathematically guaranteed* collusion resistance mechanism that operates without global consensus. ITP's Trust Independence Metric fills this gap. Furthermore, no prior work integrates all three features: observer-relative trust, Sybil-collapse via topology analysis, and game-theoretic incentives via shock propagation. The following sections describe how ITP achieves this.

2.2.7 Explicit Gaps Filled by ITP

The survey above reveals four specific gaps that no existing system addresses in combination:

1. **No global consensus, yet collusion-resistant.** Systems like Nostr and Holochain avoid global state but provide no automatic defense against Sybil or collusion rings. Systems like EigenTrust and blockchain reputation offer some collusion resistance (via stake or pre-trusted seeds) but at the cost of global consensus. ITP is the first protocol to achieve **both**: observer-relative trust *and* a mathematical guarantee (TIM) that collusion rings collapse to the weight of a single node.

2. **Context-sensitive trust with automatic decay.** Web of Trust and Nostr treat trust as binary and permanent. ITP's Trust Signals include a `context` field and an `expiry` – trust is inherently domain-specific and perishable. This is essential for Ashby's Law: a regulator that never forgets accumulates obsolete variety, becoming as brittle as one that lacks variety.

3. **Skin in the game for endorsers.** No prior decentralized trust system penalizes endorsers when their endorsements lead to harm. Blockchain stake slashing comes closest, but it requires financial collateral – excluding non-wealthy participants. ITP's Trust Shock Propagation uses *reputational* collateral, which is universal and does not require capital. The game-theoretic analysis (Section 5.5) shows this creates a Nash equilibrium favoring honest signaling.

4. **Ephemeral macro-coordination without permanent institutions.** Holochain and Nostr are purely local; they provide no mechanism for cross-scale disturbances. Blockchain DAOs create permanent governance structures that ossify. ITP's Scale-Adaptive Fields are the first design that allows temporary, dynamic pooling of regulatory capacity that automatically dissolves – solving the "who governs the governors?" problem via cryptographic TTLs.

ITP is not merely an incremental improvement on any single system; it is a new architectural category: **post-consensus, immanent, collusion-resistant, and ephemerally scaling**. The remainder of the paper provides the formal definitions, proofs, and simulation evidence for these claims.

2.3 Threat Model

A rigorous threat model is essential for any decentralized security protocol. We define the adversary's goals, capabilities, and the assumptions under which ITP provides guarantees.

2.3.1 Adversarial Goals

The adversary seeks to subvert the ITP trust computation to achieve one or more of the following:

1. **Trust inflation**: Artificially increase the Local Trust State (LTS) of a malicious node (or a colluding set) so that honest nodes grant it access to resources, endorsements, or coordination authority beyond its legitimate entitlement.
2. **Trust deflation**: Artificially decrease the LTS of an honest node, causing it to be excluded from interactions or coordination fields (a form of censorship or denial-of-service).
3. **Eclipse**: Isolate an honest node from the gossip network, feeding it only a subset of Trust Signals controlled by the adversary, thereby manipulating its LTS computation.
4. **Free riding**: Benefit from the network (e.g., resource sharing, endorsements) without contributing honestly or without facing the consequences of defection.

The adversary is assumed to be **computationally bounded** (polynomial time) but may control a large number of Sybil identities. The adversary is **adaptive** – it can observe the network and change its strategy based on past outcomes.

2.3.2 Adversarial Capabilities

We assume the following adversarial capabilities, which are standard in permissionless peer-to-peer networks:

- **Sybil creation**: The adversary can generate any number of distinct public/private key pairs at negligible cost. There is no central identity authority to prevent Sybils.
- **Collusion**: The adversary can coordinate a subset of nodes (real or Sybil) to act in concert, including issuing coordinated Trust Signals, sharing information out of band, and executing complex attack strategies.
- **Network delay and partitioning**: The adversary can delay, reorder, or drop messages between honest nodes, subject to the constraint that the network is *eventually synchronous* (i.e., messages are delivered within some bounded time after an unknown global stabilization time). This is the standard partially synchronous model [8].
- **Gossip poisoning**: The adversary can inject arbitrary Trust Signals into the gossip network, including false endorsements, false constraint signals, or signals with spoofed timestamps (cryptographic signatures prevent forgery of issuer identity, but the content can be false).
- **Long-range history rewriting**: The adversary cannot retroactively alter signed Trust Signals (due to digital signatures), but it can attempt to induce honest nodes to discard or ignore valid historical signals via network partitioning or spamming.

The adversary **cannot**:

- Break the cryptographic primitives (Ed25519 signatures, BLAKE3 hash, ZK-SNARKs) under standard computational assumptions.
- Forge a valid signature for a public key it does not control.
- Create a valid Trust Signal with a timestamp outside an acceptable drift window (relays enforce coarse time bounds).

2.3.3 Assumptions and Trusted Base

We make the following minimal assumptions:

- **Honest majority of *independent* nodes**: The network contains a sufficient number of honest nodes that are not colluding with the adversary, and these honest nodes maintain diverse, non-overlapping neighborhoods. This is required for TIM to have a baseline of independence to compare against. However, ITP does **not** require an honest majority of total nodes – only that the honest independent nodes are sufficiently numerous to provide a connected graph. Sybil rings can outnumber honest nodes locally; TIM collapses them regardless.
- **Reliable gossip with eventual delivery**: We assume that at least one honest relay path exists between any two honest nodes, and that messages eventually propagate (though delays may be adversarial). This is standard for epidemic gossip protocols.
- **Time synchronization**: Nodes maintain coarse time synchronization (e.g., via NTP) to enforce signal expiry and to detect timestamp anomalies. Drift bounds are on the order of minutes, not milliseconds.
- **Initial bootstrap trust anchors**: New nodes must obtain an initial set of Trust Signals from some trusted source (e.g., a well-known community relay or an out-of-band introduction). This is unavoidable in any permissionless system. However, once a node has a minimal LTS, it can bootstrap further autonomously (see Section 6).

2.3.4 Security Goals

ITP aims to provide the following guarantees against the defined adversary:

- **Sybil collapse (TIM guarantee)**: For any collusion ring of size S with average pairwise Jaccard similarity J, the effective weight after TIM discount is at most $1 + (1 - J)(S - 1) \cdot \delta$, where δ is a small constant. When $J \to 1$ (perfect overlap), effective weight $\to 1$ independent of S. (Formalized in Theorem 1, Section 4.7.)
- **Shock propagation deterrence**: If an adversary-controlled node defects, the expected reputational loss to its endorsers (weighted by their endorsement confidence) exceeds the one-time gain from defection, assuming the network detects the defection with high probability. (Analyzed in Section 5.5.)
- **False accusation resistance**: A single malicious node cannot trigger a constraint shock against an honest target unless it coordinates with at least M independent other nodes (where M is a system parameter, e.g., 5). Even then, the target can appeal with cryptographic proof, and false accusers suffer a calibration penalty.
- **Eclipse resilience**: An honest node connected to at least k distinct relays (with k a parameter, e.g., 8) cannot be forced to accept a false view of the trust graph, provided that at least one of those relays is honest and the node uses a diversity-aware relay selection strategy.
- **Liveness**: The network continues to make progress (Trust Signals propagate, LTS computations complete) even under adversarial load, subject to rate limits and proof-of-work for high-frequency signals.

2.3.5 Non-Goals

The following are explicitly **not** security goals of ITP:

- **Anonymity**: ITP does not aim to hide the public keys of interacting parties. Pseudonymity is preserved (keys are not linked to real identities), but interaction patterns may be linkable. Privacy enhancements (Section 8) are optional.
- **Perfect availability under massive partition**: If the network splits into two completely disconnected components for a prolonged period, trust signals cannot cross the partition, and the two sides may develop inconsistent views. This is unavoidable in any asynchronous network.
- **Protection against 51% of *independent* nodes colluding**: If a majority of *independent* nodes (i.e., those not detectable as a collusion ring) coordinate to attack a minority, TIM cannot discount them because their neighborhoods are diverse. This is analogous to a Byzantine majority in consensus systems. We assume such a scenario is unlikely in large, diverse networks, and economic/game-theoretic incentives (Section 5) further discourage it.

2.4 Comparison Table: ITP vs. Existing Systems

The following table summarizes the key architectural and security properties of ITP relative to the systems surveyed in Section 2.2. The columns capture the dimensions most relevant to Ashby's Law and adversarial resilience.

System	State Model	Sybil / Collusion Resistance	Context-Aware?	Decay?	Endorser Liability?	Ephemeral Macro-Coordination?	Im (N Sta
EigenTrust	Global consensus	Weak (pre-trusted seeds)	No	No	No	No	Nc
PageRank	Global (centralized)	Weak (link farms)	No	No	No	No	Nc
PGP Web of Trust	Local (manual)	None (manual verification)	No	No	No	No	Pa
Blockchain Reputation	Global ledger	Via stake / identity	Partial	Sometimes	Stake slashing	Permanent DAOs	Nc
Nostr	Local (relays)	None (manual follows)	No	No	No	No	Ye
Holochain	Local (agent chains)	Application-defined	App-defined	App-defined	No	No	Ye
ITP (this work)	**Local (observer-relative)**	**TIM + shock propagation (automatic, no stake)**	**Yes (context field)**	**Yes (expiry)**	**Yes (reputational collateral)**	**Yes (Scale-Adaptive Fields with TTL)**	**Ye**

Key Takeaways from the Table

1. **State Model**: Only ITP, Nostr, Holochain, and PGP avoid global consensus. However, PGP and Nostr lack automatic collusion resistance; Holochain leaves it to application developers. ITP provides a built-in, mathematically grounded solution (TIM).

2. **Sybil / Collusion Resistance**: ITP is unique in offering *automatic* resistance without stake, pre-trusted seeds, or manual verification. The TIM guarantee (Section 4.7) ensures that even million-node Sybil rings collapse to the influence of a single node.

3. **Context and Decay**: ITP's `context` field and `expiry` are essential for Ashby's Law – they prevent obsolete or irrelevant trust signals from polluting local assessments. No other system (except some blockchain reputation schemes with limited decay) provides both.

4. **Endorser Liability**: Blockchain stake slashing requires financial collateral, excluding non-wealthy participants. ITP's Trust Shock Propagation uses *reputational* liability, which is universal. This is a novel game-theoretic contribution.

5. **Ephemeral Macro-Coordination**: ITP's Scale-Adaptive Fields (Section 7) are the only mechanism that allows temporary pooling of regulatory capacity without creating permanent institutions. Blockchain DAOs and traditional hierarchies ossify; ITP fields auto-dissolve via cryptographic TTLs.

6. **Immanent (No Global State)**: ITP, Nostr, Holochain, and PGP (partially) share this property. But only ITP combines immanence with formal Sybil resistance and ephemeral scaling.

The table will be referenced throughout the paper to orient readers unfamiliar with the design space.

---

3. Core Architecture: The Informal Trust Ledger (ITL)

The Immanent Trust Protocol (ITP) replaces the concept of a global, universally agreed-upon ledger with a family of **local, observer-relative trust graphs**. This section describes the data structures, message formats, and local computation rules that constitute the **Informal Trust Ledger (ITL)**.

3.1 No Global Ledger: Each Node Maintains a Local Trust Graph

In traditional blockchains and distributed ledger systems, every node stores an identical copy of a global state. This state is updated via a consensus algorithm that ensures all nodes eventually agree on the same sequence of transactions. While this model provides strong consistency, it comes at a prohibitive cost in terms of variety (per Ashby's Law) and creates a single bottleneck for trust.

ITP takes the opposite approach: **there is no mandatory global state**. Instead, every node independently maintains its own **local trust graph** – a directed, weighted graph where:

- **Nodes** are public keys (pseudonymous identities) that the local node has encountered via gossip or direct interaction.
- **Edges** are **Trust Signals** (Section 3.2) that record direct experiences, endorsements, constraints, or warnings.
- **Weights** are dynamically computed from signal values, confidence, distance attenuation, and TIM discounting (Section 4).

3.1.1 Why Local Graphs Are Sufficient

A node does not need to know the trustworthiness of every other node in the global network. It only needs to evaluate the nodes that attempt to interact with it directly (e.g., request a resource, join a coordination field, or issue a Trust Signal). For each such interaction, the node computes a **Local Trust State (LTS)** for the target (Section 3.4) using only:

- Its own direct history with the target.
- Trust Signals from nodes it already trusts (directly, at distance 1).
- Optionally, signals from nodes trusted by its trusted nodes (distance 2), attenuated by a decay factor.

Because the evaluation is **observer-relative**, two different nodes may compute different LTS values for the same target. This is not a bug; it is a feature. Trust is inherently subjective. A node that has had positive direct experiences with a target will rationally trust it more than a node that has only heard rumors through untrusted intermediaries.

3.1.2 Gossip as the Propagation Mechanism

Trust Signals are disseminated via a **gossip protocol** over a peer-to-peer network. Any node can broadcast a signed Trust Signal to its neighbors, who may re-broadcast it to their neighbors, subject to rate limits and optional proof-of-work to prevent spam. The protocol does not require that every signal reaches every node; it only requires that signals propagate sufficiently to allow nodes to build a useful local view of their immediate trust neighborhood.

We do not assume a specific gossip topology. Implementations may use:

- **Nostr-style relays** (centralized but untrusted message boards),
- **libp2p gossip subnets** (fully peer-to-peer),
- **Mesh network flooding** (for offline or local-first deployments).

The choice of relay infrastructure is orthogonal to the trust computation; nodes can even use multiple relay mechanisms simultaneously. The only requirement is that a node can fetch Trust Signals from at least one honest relay that has not been eclipsed by an adversary (see Section 10.3).

3.1.3 No Global Ordering or Finality

Because there is no global consensus, there is also **no global ordering** of Trust Signals. Two nodes may receive the same set of signals in different orders, or may receive different subsets due to network partitions. This is acceptable because:

- The LTS computation is **commutative** under reasonable assumptions (weighted sum with decay is order-independent).
- Expiry timestamps provide a natural, subjective cutoff: signals older than a node's local retention window are ignored.
- Disputes (e.g., conflicting claims about the same interaction) are resolved via **cryptographic proofs** (e.g., signed receipts or ZK-proofs), not via global ordering.

The absence of global ordering also eliminates the need for a leader election or a longest-chain rule, removing entire classes of attacks (e.g., 51% chain reorganizations) and dramatically simplifying the protocol.

3.1.4 Storage and Retention

Each node stores only a bounded subset of the global trust graph. The **local trust graph** is pruned according to:

- **Distance attenuation**: Signals from nodes at hop distance ≥ 4 are discarded (Section 3.4).
- **Expiry**: Signals older than their `expiry` timestamp are deleted.
- **Space limits**: Nodes may evict the lowest-weight or oldest signals when storage exceeds a configured maximum (e.g., 1 million entries).

This bounded storage ensures that the protocol remains feasible on resource-constrained devices (mobile phones, IoT sensors) while still providing sufficient variety for local trust assessment. An archival network of volunteer nodes may store the full history, but the protocol does not rely on their existence.

3.1.5 Comparison with Global Ledgers

Property	Global Ledger (Blockchain)	ITP Local Trust Graph
State	Single, identical across nodes	Many, observer-relative
Update mechanism	Consensus (costly, slow)	Gossip + local computation (cheap, fast)
Consistency	Strong eventual consistency	No global consistency required
Sybil resistance	Via stake or external identity	Via TIM (topological)
Variety	Low (global state is low-dim)	High (each node sees its own context)
Failure mode	Fork, 51% attack, state bloat	Partition, isolated deception

The local trust graph model is strictly more expressive and resilient, at the cost of giving up global agreement on "truth." Since trust does not require global agreement, this trade-off is acceptable.

3.2 The Atomic Unit: The Trust Signal (TS)

The foundation of the Informal Trust Ledger (ITL) is the **Trust Signal (TS)** – a cryptographically signed, verifiable claim issued by one node about another, strictly bound to a specific context and time window. Trust Signals are the only way that nodes influence each other's Local Trust State (LTS). Every signal is a self-contained, immutable statement that can be stored, relayed, and independently verified.

3.2.1 Trust Signal Schema (Version 1.0)

The following JSON schema defines the required and optional fields of a Trust Signal. All fields except `zk_proof_of_interaction` and `calibration_history` are mandatory.

```
{
  "signal_id": "string (sha256_hash of canonical encoding)",
  "issuer_pubkey": "string (ed25519 public key in multibase format)",
  "subject_pubkey": "string (ed25519 public key in multibase format)",
  "context": "string (URN or DID-based namespace)",
  "type": "interaction | endorsement | constraint | warning",
  "value": "float in [-1.0, 1.0]",
  "confidence": "float in [0.0, 1.0]",
  "timestamp": "ISO 8601 UTC timestamp with millisecond precision",
  "expiry": "ISO 8601 UTC timestamp (must be > timestamp)",
  "zk_proof_of_interaction": "optional string (base64-encoded ZK-SNARK proof)",
  "calibration_history": "optional string (URL or hash link to public calibration record)",
```

```
  "signature": "string (ed25519 signature over canonical encoding of all above fields except signature)"
}
```

Field semantics:

- `signal_id` : A hash of the canonical byte representation of the signal (excluding the signature, which is added later). Used as a unique identifier for deduplication and caching.
- `issuer_pubkey` : The public key of the node issuing the signal. The signature must verify against this key.
- `subject_pubkey` : The public key of the node being evaluated. The issuer may be the same as the subject (self-endorsement), but such signals are heavily discounted by default (see Section 4).
- `context` : A URI or DID-based identifier that defines the domain of trust. Examples:
  - `"urn:itp:context:resource-sharing:file-storage"`
  - `"did:example:123?service=water-quality"`
  - `"https://schemas.itp.org/contexts/bike-sharing/v1"`

  Contexts are resolved via decentralized schema registries (Section 3.5). Signals with different contexts are treated as belonging to entirely separate trust graphs.
- `type` :
  - `interaction` : Direct experience (e.g., "I shared a file with this node and the transfer completed successfully"). Carries the highest weight for the issuer.
  - `endorsement` : A second-hand recommendation (e.g., "I trust this node for this context, though I have not directly interacted"). Weighted lower than `interaction` .
  - `constraint` : A negative signal indicating that the subject should be distrusted or blocked. May trigger shock propagation (Section 5).
  - `warning` : A low-confidence alert (e.g., "I heard a rumor that this node might be malicious"). Weighted very low unless corroborated.
- `value` : The polarity and magnitude of the assessment. Positive values indicate trust, negative values indicate distrust. Zero is neutral (no signal).
- `confidence` : The issuer's subjective certainty. A high-value but low-confidence signal (e.g., `value=0.9, confidence=0.2`) contributes little to LTS. Confidence can be calibrated over time (Section 5.3).
- `timestamp` : The time when the signal was created. Used for ordering and for freshness checks (signals created far in the future are rejected).
- `expiry` : The time after which the signal should be ignored. Enforces *liberatory impermanence* – trust must be renewed. Typical expiry ranges from days (for ephemeral contexts) to years (for long-term credentials).
- `zk_proof_of_interaction` : Optional zero-knowledge proof that the interaction described actually occurred, without revealing details (e.g., amount, content, counterparty metadata). See Section 8 for privacy trade-offs.
- `calibration_history` : Optional link to a public record of the issuer's past accuracy. Nodes may use this to discount signals from poorly calibrated issuers (Section 5.3).
- `signature` : Ed25519 signature over the canonical byte encoding of all other fields. Ensures authenticity and integrity.

3.2.2 Cryptographic Guarantees

Each Trust Signal provides the following guarantees to any verifying node:

- **Authenticity**: The signature proves that the issuer's private key (and thus, presumably, the node itself) generated the signal. An adversary cannot forge signals for honest nodes.
- **Integrity**: Any modification to the signal fields (including `timestamp` and `expiry`) invalidates the signature.
- **Non-repudiation**: The issuer cannot later deny having issued the signal, as the signature is publicly verifiable.
- **Pseudonymity**: The issuer and subject are identified only by public keys. No real-world identity is required.

3.2.3 Signal Lifecycle

A Trust Signal passes through the following stages:

1. **Creation**: The issuer generates the JSON object, computes the canonical encoding, signs it, and stores the signed signal locally.
2. **Gossip**: The issuer broadcasts the signal to its neighbors (or publishes to relays). Each receiving node may re-broadcast to its own neighbors, subject to rate limits and optional proof-of-work.
3. **Storage**: Nodes that receive the signal store it in their local trust graph if it passes validation (signature correct, timestamp within acceptable drift, expiry not passed, subject ≠ issuer unless explicitly allowed).
4. **Application**: When computing an LTS for a target node, the evaluating node queries its local store for all relevant signals (subject = target, context matching, expiry > now).
5. **Pruning**: After `expiry` passes, the signal is deleted from the local store. Nodes may also evict signals earlier due to space constraints.

3.2.4 Validation Rules

Before accepting a Trust Signal, a node must verify:

- The signature is cryptographically valid.
- The `timestamp` is not more than, e.g., 1 hour in the future (configurable clock drift tolerance).
- The `expiry` is greater than the current time (or within a grace period, e.g., 1 hour, to account for propagation delays).
- The `context` is a valid URI (syntactic check only; semantic resolution is deferred to Section 3.5).
- The `confidence` is between 0.0 and 1.0 inclusive.
- The `value` is between -1.0 and 1.0 inclusive.
- If `type` is `interaction` and the issuer == subject, the signal is either rejected or heavily discounted (configurable). Self-endorsement is generally discouraged.

Signals that fail validation are discarded and not re-broadcast.

3.2.5 Example: A Simple Endorsement

```
{
  "signal_id": "sha256:7f83b1657ff1fc53b92dc18148a1d65dfc2d4b1fa3d677284addd200126d9069",
  "issuer_pubkey": "ed25519:abcd1234...",
  "subject_pubkey": "ed25519:efgh5678...",
  "context": "urn:itp:context:resource-sharing:file-storage",
  "type": "endorsement",
  "value": 0.85,
  "confidence": 0.9,
  "timestamp": "2026-04-05T12:00:00Z",
  "expiry": "2027-04-05T12:00:00Z",
  "zk_proof_of_interaction": null,
  "calibration_history": "https://relay.example.com/calibration/abcd1234",
  "signature": "ed25519:5f7a3b2c..."
}
```

This signal indicates that issuer `abcd1234` strongly endorses subject `efgh5678` for file-sharing, with high confidence, and the endorsement expires in one year.

3.3 Gossip Propagation: Disseminating Trust Signals

Trust Signals are worthless if they cannot reach the nodes that need them. ITP uses a **gossip protocol** to propagate signals across the network. Unlike blockchain transaction flooding (which aims for global broadcast), ITP's gossip is **best-effort and locality-aware**: signals need only propagate far enough to reach nodes that might have a legitimate interest in the subject or issuer.

3.3.1 Design Principles

- **Decentralized**: No central server or relay is required. Any node can participate in forwarding signals.

- **Resilient**: The network continues to function even if many nodes go offline or become adversarial.
- **Rate-limited**: Prevents spam and denial-of-service attacks.
- **Privacy-preserving by default**: Signals are public (pseudonymous), but optional ZKPs can hide interaction details (Section 8).
- **Locality-aware**: Signals that are distant (in trust-hop sense) are less relevant; gossip does not need to achieve global flooding.

3.3.2 Relay Abstraction

The protocol defines an abstract **relay** – any node that accepts, stores, and forwards Trust Signals. Implementations may use:

- **Nostr-style relays** [6]: Centralized or federated servers that store signals and serve them to clients. Relays are untrusted; clients query multiple relays to avoid censorship.
- **libp2p gossip subnets** [9]: Fully peer-to-peer flooding using a distributed hash table (DHT) or epidemic broadcast.
- **Local mesh flooding**: For offline-first or disaster-recovery scenarios, nodes can forward signals to all directly connected neighbors.

Nodes are free to use multiple relay mechanisms simultaneously. The trust computation is independent of the relay infrastructure; signals are just data.

3.3.3 Basic Gossip Algorithm

A node that wishes to disseminate a Trust Signal (either because it created it or because it received it from another node) follows this algorithm:

1. **Validate** the signal (signature, timestamp, expiry, etc.). Drop invalid signals.
2. **Deduplicate**: If the signal's `signal_id` is already in the local cache, discard it.
3. **Apply rate limits**: If the issuer (or the forwarding node) has exceeded its allowed signal rate (e.g., 10 signals per minute), drop or delay the signal.
4. **Store** the signal locally (in SQLite or similar).
5. **Forward** to a random subset of neighbors (or to all connected relays), with a **time-to-live (TTL)** hop counter that decrements each hop.
6. **Optionally attach proof-of-work** if the signal's type is `endorsement` or `warning` and the node is not a well-known relay. This raises the cost of spam (Section 3.3.6).

The forwarding probability and TTL are configurable. Typical defaults: forward to 8 random neighbors, TTL = 3 hops. This limits the propagation radius, which aligns with distance attenuation (Section 3.4) – signals beyond hop 3 are heavily discounted anyway.

3.3.4 Relay Diversity and Eclipse Protection

A node must not rely on a single relay; an adversarial relay could withhold signals (censorship) or feed false signals. The reference implementation requires each node to connect to at least `k` **distinct relays** (e.g., k = 8), selected from a diverse set:

- Randomly chosen from a seed list,
- Discovered via DHT or peer exchange,
- Provided by out-of-band configuration.

When computing a Local Trust State (Section 3.4), the node queries all connected relays for signals relevant to the target and context, then merges the results. If a signal appears on a majority of relays (or is signed by a highly trusted issuer), the node can be confident it is widely accepted.

3.3.5 Signal Propagation Scope

Not all signals need to travel everywhere. The protocol defines **propagation scope** based on signal type:

Signal Type	Recommended TTL	Rationale
`interaction`	1 hop	Direct experience is only relevant to the issuer and its immediate neighbors.
`endorsement`	2 hops	Endorsements are useful for friends-of-friends but not beyond.

Signal Type	Recommended TTL	Rationale
`constraint`	3 hops	Warnings should propagate further to alert the network, but still bounded.
`warning`	2 hops	Lower confidence, so less propagation.

Nodes may override these defaults based on local policy (e.g., a `constraint` from a highly trusted node might be propagated further).

3.3.6 Spam Prevention

An adversary could flood the network with millions of low-value Trust Signals. ITP uses two complementary mechanisms:

- **Rate limiting per issuer**: Each public key is limited to, e.g., `N` signals per hour (configurable). Exceeding this rate causes the node to be temporarily ignored.
- **Proof-of-work (PoW) for high-rate or low-trust issuers**: A signal may require a PoW (e.g., hashcash) that takes a few milliseconds to compute on average hardware. This makes large-scale spam expensive.
- **Reputation-based throttling**: Nodes with low LTS (as seen by the relay) are given stricter rate limits.

Relays can independently enforce these limits; there is no global rate limit.

3.3.7 Gossip and Network Partitions

During a network partition, signals from one side of the partition will not reach the other. When the partition heals, nodes will reconcile by fetching missing signals from relays (using `since` timestamps). Because there is no global ordering, no reorg or rollback is required – each node simply incorporates newly received signals into its local trust graph, updating LTS values asynchronously. This is a major advantage over blockchains, which require complex fork resolution.

3.3.8 Reference Implementation Choices

The reference implementation (Section 9) uses:

- **Nostr relays** for wide-area dissemination (due to their simplicity and existing ecosystem).
- **libp2p gossipsub** for peer-to-peer flooding when relays are unavailable.
- **SQLite** for local storage, indexed by `subject_pubkey`, `context`, and `expiry`.

A node acting as a relay stores signals for a bounded time (e.g., 30 days) and serves them to clients via a simple REST or WebSocket API. Relays are not trusted; clients always verify signatures and apply their own trust computation.

3.4 Computing the Local Trust State (LTS)

When a node (the **evaluator**) needs to assess the trustworthiness of a target node (the **subject**) in a specific context, it does not query a global ledger or a central server. Instead, it computes a **Local Trust State (LTS)** in real time, using only the Trust Signals stored in its local graph.

The LTS is a scalar value in the range $[-1, 1]$, where positive values indicate trust, negative values indicate distrust, and zero indicates no information or neutrality. The computation is entirely deterministic given the evaluator's local signal set and a few global parameters.

3.4.1 Information Sources

The evaluator considers three sources of information, ordered by decreasing weight:

1. **Direct history** – Signals where the evaluator is the issuer and the subject is the target, with `type = interaction`. These represent first-hand experience and carry the highest weight.

2. **First-degree network** – Signals issued by nodes that the evaluator *directly trusts* (i.e., for which the evaluator has a positive direct history). The evaluator first computes a **trust weight** for each of its direct peers (using the same LTS function recursively, or a cached value), then uses those weights to discount the peers' endorsements.

3. **Second-degree network** — Signals issued by nodes that are trusted by the evaluator's trusted nodes. These are further attenuated by distance.

Signals from nodes at hop distance 3 or greater are ignored (distance attenuation multiplier = 0). This bounds the computational complexity and respects Ashby's Law: information from far away is too low-fidelity to be useful.

3.4.2 Distance Attenuation Multiplier

Let d be the shortest path length (in hops) from the evaluator to the issuer of a Trust Signal, measured through the graph of direct `interaction` signals (not endorsements). The protocol defines a **distance attenuation multiplier** $\lambda(d)$:

$$ \lambda(d) = \begin{cases} 1.0 & \text{if } d = 1 \text{ (direct interaction)} \\ 0.5 & \text{if } d = 2 \text{ (friend of friend)} \\ 0.1 & \text{if } d = 3 \\ 0.0 & \text{if } d \ge 4 \end{cases} $$

These values are configurable but are fixed for a given deployment. The rapid decay reflects the cybernetic principle that trust is not perfectly transitive. A node that is two hops away is at best half as credible as a direct peer.

3.4.3 Aggregation Function

Let S be the set of all Trust Signals stored locally that satisfy:

- `subject_pubkey` = target public key.
- `context` matches the requested context (exact string match, or via ontology mapping from Section 3.5).
- `expiry` > current time.
- The issuer is within distance $d \le 3$ from the evaluator, where distance is measured through `interaction` edges only.

For each signal $s \in S$, we compute a **raw contribution**:

$$ \text{contrib}(s) = \text{value}(s) \times \text{confidence}(s) \times \lambda(d(s)) $$

where $d(s)$ is the hop distance from evaluator to issuer.

Next, we apply the **Trust Independence Metric (TIM)** to the set of endorsers (Section 4). For now, assume a TIM discount factor $\tau \in [0, 1]$ is computed for the set of issuers. The discounted contribution of a signal becomes:

$$ \text{disc_contrib}(s) = \text{contrib}(s) \times \tau $$

Finally, the LTS is the **weighted average** of all discounted contributions, normalized by the sum of weights:

$$ \text{LTS}_{\text{evaluator}}(\text{target}) = \frac{\sum_{s \in S} \text{contrib}(s) \times \tau(S)}{\sum_{s \in S} \text{confidence}(s) \times \lambda(d(s)) + \epsilon} $$

where ϵ is a small constant (e.g., 10^{-6}) to avoid division by zero when there are no signals. If no signals exist, the LTS defaults to 0 (neutral).

Important: The TIM discount factor $\tau(S)$ is applied to the entire set of signals from the same context and target, not per signal individually. This ensures that collusion rings cannot bypass TIM by issuing many signals; the entire cluster is discounted as one logical entity.

3.4.4 Handling Multiple Contexts

If the evaluator needs a trust assessment across multiple contexts (e.g., both "file storage" and "messaging"), it computes separate LTS values per context and may combine them using a user-defined policy (e.g., minimum, weighted sum, or require separate thresholds). The protocol does not prescribe cross-context aggregation; it is application-specific.

3.4.5 Example Calculation

Suppose evaluator A wants to assess target X for context `"file-sharing"`. A has the following signals:

Issuer	Distance d	Type	value	confidence	λ(d)	contrib
A (self)	1	interaction	+0.9	0.95	1.0	0.855
B (trusted friend)	2	endorsement	+0.8	0.8	0.5	0.32

Issuer	Distance d	Type	value	confidence	λ(d)	contrib
C (friend of B)	3	endorsement	+0.7	0.9	0.1	0.063

Assume TIM discount $\tau = 1.0$ (no collusion detected). The sum of contrib = 0.855 + 0.32 + 0.063 = 1.238. The sum of weights (confidence × λ) = 0.95 + (0.8×0.5=0.4) + (0.9×0.1=0.09) = 1.44. Then LTS = 1.238 / 1.44 ≈ 0.86 (high trust).

If a collusion ring were present (e.g., B and C have identical neighborhoods), TIM might reduce τ to 0.5, cutting the contributions from B and C in half, yielding LTS ≈ (0.855 + 0.16 + 0.0315) / (0.95 + 0.4 + 0.09) = 1.0465 / 1.44 ≈ 0.73 – still positive but lower.

3.4.6 Caching and Recomputations

LTS computation can be expensive if done frequently for many targets. Nodes may cache LTS values with a short time-to-live (e.g., 5 minutes) and recompute only when new relevant signals arrive or when a cached entry expires. The protocol does not mandate a specific caching strategy; implementers may choose based on their performance requirements.

3.4.7 Relation to Thresholds

Applications define **trust thresholds** for different actions. For example:

- Grant read access to public data: LTS ≥ 0.3
- Grant write access to shared resource: LTS ≥ 0.7
- Accept a `constraint` signal as valid: LTS of issuer ≥ 0.8

Thresholds are local to each node; there is no global policy. This allows each node (or each application) to tune its risk tolerance.

3.5 Context Resolution and Ontological Mapping

Trust Signals are bound to a `context` field – a URI that defines the domain of trust (e.g., `"urn:itp:context:resource-sharing:file-storage"`). However, in a decentralized network with no central authority, how do nodes agree on what a given context URI means? How do they handle synonyms (e.g., `"file-sharing"` vs. `"file-storage"`)? This section describes the mechanisms ITP uses for context resolution without global coordination.

3.5.1 The Problem of Context Semantics

If two nodes use different but semantically equivalent context strings, their trust graphs will be disjoint, and TIM may fail to detect collusion that spans multiple contexts. For example:

- Node A issues signals with context `"urn:itp:context:bike-sharing"`
- Node B issues signals with context `"urn:itp:context:bicycle-rental"`

A human knows these are the same, but the protocol treats them as separate graphs. Without a resolution mechanism, the network fragments.

3.5.2 Decentralized Schema Registry (Optional but Recommended)

ITP does not mandate a global registry, but provides a **reference implementation** of a decentralized schema registry using **DIDs** and **JSON-LD** [10]. The registry is itself built on ITP: context definitions are stored as Trust Signals with a special `context` value `"urn:itp:meta:context-definition"`.

A context definition includes:

- The canonical context URI.
- A human-readable label and description.
- A list of aliases (alternative URIs that map to the same context).
- A link to a JSON-LD schema that defines valid interaction types and value ranges.
- The public key of the definition issuer (optional; definitions may be unsigned for open registries).

Nodes can query the registry (via gossip or relays) to resolve a context URI to its definition and aliases.

3.5.3 Ontology Mapping via Fuzzy Matching

For cases where no formal definition exists, nodes may apply **fuzzy matching** to map similar context strings to a canonical form. The algorithm:

1. Normalize the string: lowercasing, remove punctuation, replace underscores/hyphens with spaces.
2. Compute a similarity score (e.g., Levenshtein distance or Jaccard on word bigrams).
3. If similarity exceeds a threshold (e.g., 0.85), treat the contexts as equivalent for the purpose of LTS computation, but **record the mapping locally** with a confidence penalty.

The confidence of the mapping decays over time; if two contexts are consistently used interchangeably by many nodes, the mapping becomes more trusted.

3.5.4 Context Aliasing in Trust Signals

Issuers may include multiple context URIs in a single signal using a **context set** (a JSON array). The signal is then considered valid for any of the listed contexts. This is useful for broad endorsements (e.g., "I trust this node for all resource sharing"). The schema supports:

```
"context": ["urn:itp:context:file-storage", "urn:itp:context:streaming"]
```

Evaluators treat the signal as applying to each context independently.

3.5.5 Local Context Policies

Each node may define its own **context policy**:

- **Strict mode**: Only exact string matches count. Use for high-stakes contexts (e.g., financial transfers).
- **Loose mode**: Apply fuzzy matching and registry lookups. Use for low-stakes or exploratory interactions.
- **Whitelist mode**: Only accept signals from a pre-approved list of context URIs.

The policy is local; different nodes may have different settings, and trust assessments will reflect those differences. This is acceptable because trust is subjective.

3.5.6 Example: Resolving a Context

Suppose a node receives a signal with context `"https://example.com/ns/bike-sharing#v1"`. The node:

1. Checks its local cache for a definition of that URI.
2. If not found, queries its configured relays for signals with `context = "urn:itp:meta:context-definition"` and `subject` matching the URI.
3. If a definition is found with alias `"urn:itp:context:bike-sharing"`, the node maps the incoming context to the canonical form.
4. If no definition is found, the node falls back to fuzzy matching against known contexts. If similarity > 0.9, it uses the closest match and logs a low-confidence mapping.
5. If no match, the node treats the context as unknown and may either ignore the signal or accept it with a very low weight (configurable).

3.5.7 Security Considerations

- **Context squatting**: An adversary could define a malicious context with a misleading name (e.g., `"urn:itp:context:trusted-bank"`). Nodes mitigate this by only trusting context definitions signed by known, trusted issuers (or by relying on community-maintained registries).
- **Mapping poisoning**: An adversary could issue fake mapping signals to confuse fuzzy matching. Because mapping is local and weighted by issuer trust, a single adversary has limited effect.

3.5.8 Summary

Context resolution in ITP is **decentralized and flexible**. Nodes may use a registry, fuzzy matching, or strict exact matching according to their needs. The protocol does not force a single global ontology, but provides tools for interoperation. This aligns with the immanent philosophy: each node decides what contexts mean, based on its own trust assessments of definition issuers.

3.6 Worked Example: Computing LTS with TIM and Context Resolution

To illustrate the complete LTS computation, consider a concrete scenario involving an evaluator node **Alice**, a target node **Bob**, and a context `"urn:itp:context:file-storage"`. Alice wants to decide whether to grant Bob write access to a shared folder (requires LTS ≥ 0.6).

3.6.1 Step 1: Gather Relevant Trust Signals

Alice's local trust graph contains the following relevant signals (others filtered out by context and expiry). For simplicity, all signals have `expiry` in the future.

Signal ID	Issuer	Type	value	confidence	Direct trust from Alice to issuer?
S1	Alice (self)	interaction	+0.95	0.99	N/A (self)
S2	Carol	endorsement	+0.90	0.85	Yes (direct trust 0.8)
S3	Dave	endorsement	+0.70	0.60	Yes (direct trust 0.9)
S4	Eve	endorsement	+0.85	0.75	No (but Carol trusts Eve with 0.7)
S5	Frank	endorsement	+0.80	0.80	No (but Dave trusts Frank with 0.8)
S6	Grace	endorsement	+0.95	0.95	No (Carol trusts Grace with 0.9)
S7	Henry	endorsement	+0.20	0.90	Yes (direct trust 0.7) – but low value
S8	Ivy	constraint	-0.90	0.95	Yes (direct trust 0.85) – negative

3.6.2 Step 2: Compute Distance and Attenuation

We measure hop distance from Alice to each issuer using only `interaction` edges (direct experience). For issuers that Alice has directly interacted with, distance = 1. For issuers that are trusted by Alice's direct peers, we calculate shortest path through the trust graph.

- **S1 (Alice)**: distance = 1 → λ = 1.0
- **S2 (Carol)**: Alice directly trusts Carol (interaction) → distance = 1 → λ = 1.0
- **S3 (Dave)**: distance = 1 → λ = 1.0
- **S4 (Eve)**: Alice → Carol (trusted) → Eve. Carol's trust of Eve is known from an interaction signal. Path length = 2 → λ = 0.5
- **S5 (Frank)**: Alice → Dave → Frank. Path length = 2 → λ = 0.5
- **S6 (Grace)**: Alice → Carol → Grace. Path length = 2 → λ = 0.5
- **S7 (Henry)**: distance = 1 → λ = 1.0
- **S8 (Ivy)**: distance = 1 → λ = 1.0

3.6.3 Step 3: Compute Raw Contributions (Before TIM)

Raw contribution = value × confidence × λ.

Signal	value × confidence × λ = raw contrib
S1	0.95 × 0.99 × 1.0 = 0.9405
S2	0.90 × 0.85 × 1.0 = 0.765
S3	0.70 × 0.60 × 1.0 = 0.42
S4	0.85 × 0.75 × 0.5 = 0.31875
S5	0.80 × 0.80 × 0.5 = 0.32
S6	0.95 × 0.95 × 0.5 = 0.45125
S7	0.20 × 0.90 × 1.0 = 0.18
S8	-0.90 × 0.95 × 1.0 = -0.855

Sum of raw contributions = 0.9405 + 0.765 + 0.42 + 0.31875 + 0.32 + 0.45125 + 0.18 − 0.855 = 2.5405

3.6.4 Step 4: Apply Trust Independence Metric (TIM)

We must check for collusion among the endorsers. Consider the set of issuers other than Alice: {Carol, Dave, Eve, Frank, Grace, Henry, Ivy}. We compute pairwise Jaccard similarity of their neighbor sets (neighbors = nodes they directly trust). For brevity, assume:

- Carol and Grace share 80% of neighbors (high overlap)
- Dave and Frank share 70% overlap
- Eve is independent (low overlap with others)
- Henry is isolated (no overlap)
- Ivy is isolated (no overlap)

TIM discount factor τ is computed per the formula in Section 4. For this example, assume $\tau = 0.85$ (some collusion detected, but not complete).

Important: TIM is applied to the set of endorsers as a whole, not per signal. The raw contributions from all endorsers are multiplied by τ. However, signals from Alice (the evaluator's own direct experience) are *not* discounted because they are not endorsements — they are first-hand. So we separate:

- Direct contributions (Alice only): 0.9405
- Endorser contributions (all others): sum = 0.765 + 0.42 + 0.31875 + 0.32 + 0.45125 + 0.18 − 0.855 = 1.6 (approximately)

Discounted endorser sum = 1.6 × 0.85 = 1.36

Total discounted sum = 0.9405 + 1.36 = 2.3005

3.6.5 Step 5: Normalize by Weight Sum

We compute the denominator: sum of (confidence × λ) for each signal, with TIM applied to endorsers' weights in the same way.

- Direct (Alice): 0.99 × 1.0 = 0.99
- Endorsers raw weight sum = (0.85+0.60+0.75+0.80+0.95+0.90+0.95) = 5.8 (note: includes negative signal's confidence positively — weight is absolute)
- Discounted endorser weight = 5.8 × 0.85 = 4.93
- Total denominator = 0.99 + 4.93 = 5.92

3.6.6 Step 6: Compute LTS

LTS = total discounted sum / total denominator = 2.3005 / 5.92 ≈ 0.389

3.6.7 Step 7: Apply Threshold Decision

Alice's threshold for write access is 0.6. The computed LTS of 0.389 is below 0.6, so Alice **denies** Bob write access. However, she might grant read access (threshold 0.3) or ask for additional verification.

3.6.8 What If TIM Were Not Applied?

Without TIM, the LTS would be raw sum / raw weight sum = 2.5405 / (0.99 + 5.8) = 2.5405 / 6.79 ≈ 0.374 — actually slightly lower. In this example, TIM increased the LTS slightly because the collusion discount reduced the weight of negative signals (Ivy's constraint) proportionally more than positive signals? Let's check: Ivy's contribution was negative, so discounting it helped Bob. TIM is neutral; it discounts all endorsements equally, which can help or hurt depending on the balance. The key is that TIM prevents collusion rings from artificially inflating or deflating trust.

3.6.9 Context Resolution Note

Suppose some signals used a different but synonymous context URI: `"urn:itp:context:file-sharing"`. Alice's node, using fuzzy matching (similarity 0.92), maps them to the canonical `"file-storage"` and includes them in the computation with a small confidence penalty (e.g., multiply confidence by 0.95). The example above already incorporates that implicitly.

---

4. Trust Independence Metric (TIM)

The Local Trust State computation described in Section 3 is sufficient for networks where nodes issue honest, independent endorsements. However, in a permissionless environment, adversaries can create **Sybils** (fake identities) and coordinate **collusion rings** to manipulate trust assessments. This section introduces the **Trust Independence Metric (TIM)** – a mathematical tool that detects and discounts such coordinated behavior by analyzing the topological structure of the trust graph.

4.1 Problem: Coordinated Collusion and Sybil Rings

Consider a naive transitive trust model where Node A evaluates Node X by aggregating endorsements from all nodes that A trusts (directly or indirectly). An adversary can exploit this in two ways:

4.1.1 Sybil Attack

The adversary creates thousands of fake public keys (Sybils) with no cost. Each Sybil issues a high-value endorsement of a malicious target node `X_mal`. To A, these appear as many independent voices. Without a mechanism to distinguish Sybils from genuine nodes, A would compute an artificially high LTS for `X_mal`.

Formally, let S be the number of Sybils controlled by the adversary. If A trusts each Sybil equally (e.g., via a transitive path), the aggregated trust score for `X_mal` grows linearly with S, even though the Sybils represent only one logical actor. This violates the principle that trust should not be multiplicatively inflatable by cheap identities.

4.1.2 Coordinated Collusion Ring

A more sophisticated adversary does not rely on pure Sybils alone. It may control a set of real or Sybil nodes that **coordinate** their endorsements and their neighbor relationships. For example:

- **Cross-endorsement ring**: Nodes F, G, and H each endorse each other and also endorse target X. To an outside observer, there are three distinct endorsers. But if F, G, and H share nearly identical neighborhoods (they only interact with each other and with X), they are not independent. They are a single colluding cluster.
- **Echo chamber**: A collusion ring may also coordinate the timing and polarity of their signals. They all issue `value=+0.9` on the same day, creating a spike that naive aggregation would interpret as strong consensus.

4.1.3 Why Existing Approaches Fail

Traditional defenses against Sybils include:

- **Proof-of-work** (e.g., Hashcash): Each identity must perform computational work. This raises the cost of Sybils but does not eliminate them; wealthy adversaries can still create millions. Moreover, it penalizes honest nodes with limited compute.
- **Proof-of-stake**: Sybils must lock financial collateral. This excludes non-wealthy participants and centralizes power.
- **Social graph analysis** (e.g., SybilRank [11]): Assumes that honest nodes form a densely connected core and Sybils are attached via few edges. This works in static, carefully curated graphs but fails when the adversary can create many inter-connections among Sybils (dense collusion rings) or when the honest graph is not well-connected.

TIM takes a different approach: **structural independence**. Instead of measuring *how many* endorsers a target has, TIM measures *how independent* those endorsers are from each other. If a set of endorsers has nearly identical neighborhoods (high topological overlap) or exhibits perfectly correlated signaling, they are treated as a single logical entity regardless of how many distinct public keys they use.

4.1.4 Adversary Model for TIM

We assume the adversary can:

- Create any number of Sybil public keys.
- Arrange the Sybils' neighbor relationships arbitrarily (i.e., decide which nodes each Sybil trusts).
- Coordinate the timing and values of signals issued by Sybils.

However, the adversary **cannot**:

- Control the neighbor sets of honest nodes (though it may influence them via endorsements).
- Prevent honest nodes from having diverse, non-overlapping neighborhoods (unless the honest graph itself is pathological, which we assume is not the case in a healthy network).

Under this model, the adversary's Sybils and colluding nodes will inevitably exhibit **high overlap** in their neighbor sets because they share the same limited set of accomplices. Honest nodes, by contrast, have diverse, naturally overlapping neighborhoods only to the extent that their social or operational connections intersect. TIM exploits this asymmetry.

4.1.5 The Goal of TIM

TIM is designed to satisfy the following property:

> ***Collapse property***: *For any set of endorsers E that forms a collusion ring (i.e., all pairs have Jaccard similarity > threshold), the effective weight of E after TIM discount is bounded by a constant independent of $|E|$. In the limit of perfect overlap ($J \to 1$), the effective weight tends to 1 (the weight of a single node).*

This ensures that an adversary cannot scale its influence by adding more Sybils or colluding nodes. The next subsection (4.2) provides the formal definition of TIM that achieves this property.

> ***Try it yourself***: *The TIM Sybil Collapse Visualizer lets you adjust ring size and overlap in real time, and compare TIM against EigenTrust and SybilRank. Watch a 10,000-node Sybil ring collapse to near-zero influence as you increase the overlap.*

4.2 Formal Definition of TIM

The Trust Independence Metric (TIM) is a function that takes a set of endorsers E (public keys that have issued Trust Signals for the same target and context) and returns a discount coefficient $\tau \in [0, 1]$. This coefficient is applied to the aggregated contributions of all endorsers when computing the Local Trust State (Section 3.4). TIM does not depend on the specific target being evaluated, only on the structure of the endorser set and their neighborhoods.

4.2.1 Neighborhood Definition

For any node i (public key), define its **neighborhood** $Nbr(i)$ as the set of nodes that i directly trusts via `interaction` signals within a relevant time window (e.g., the last 90 days). Only `interaction` signals count; endorsements are not used to define neighborhoods, as they are second-hand and potentially collusive.

Formally:

$$ Nbr(i) = \{ j \mid \exists \text{ an interaction signal from } i \text{ to } j \text{ with } \text{timestamp} > \text{now} - \Delta_T \} $$

where Δ_T is a configurable freshness window (e.g., 90 days). Nodes without any outgoing interaction signals have $Nbr(i) = \emptyset$.

For computational tractability, neighborhoods are **capped** at a maximum size $d_{\max} = 100$ (randomly sampled if larger). This bound is justified because trust beyond a few dozen close peers is already diffuse; including more does not significantly improve independence measurement.

4.2.2 Pairwise Jaccard Similarity

For any two endorsers $i, j \in E$, define the **Jaccard similarity** of their neighborhoods:

$$ J(i, j) = \frac{|Nbr(i) \cap Nbr(j)|}{|Nbr(i) \cup Nbr(j)|} $$

If both neighborhoods are empty, define $J(i, j) = 1$ (both are indistinguishable from the perspective of independence). If one is empty and the other non-empty, $J(i, j) = 0$.

The Jaccard similarity ranges from 0 (completely disjoint neighborhoods) to 1 (identical neighborhoods). High Jaccard similarity indicates that the two endorsers share most of their trusted contacts, which is a strong signal of collusion or Sybil identity.

4.2.3 Signal Correlation Penalty (Optional)

In addition to topological overlap, colluding nodes may exhibit **correlated signaling** – they issue signals with nearly identical `value` and `confidence` at the same time. We define an optional correlation penalty:

$$ C(i, j) = \frac{1}{1 + \text{Var}(v_i, v_j) + \text{Var}(c_i, c_j)} $$

where v_i, v_j are the values of the signals from i and j (for the same target and context), and c_i, c_j are their confidences. If multiple signals exist from the same issuer (e.g., over time), we take the most recent. The variance is normalized to $[0, 1]$ range. Perfectly identical signals yield $C(i, j) = 1$; very different signals yield $C(i, j) \to 0$.

The combined **similarity** between i and j is:

$$ S(i, j) = \alpha \cdot J(i, j) + (1-\alpha) \cdot C(i, j) $$

where $\alpha \in [0, 1]$ is a configurable weight (default $\alpha = 0.7$, giving more weight to topology). If correlation data is unavailable, $\alpha = 1.0$ (topology only).

4.2.4 Endorser Set Independence Score

Given the set of endorsers $E = \{e_1, e_2, \dots, e_k\}$, we compute the **pairwise similarity matrix** $S_{pq} = S(e_p, e_q)$. The overall independence score for the set is the average of one minus the average similarity:

$$ \text{Ind}(E) = 1 - \frac{2}{k(k-1)} \sum_{p<q} S(e_p, e_q) $$

If $k = 1$ (only one endorser), define $\text{Ind}(E) = 1$ (no collusion possible). If $k = 0$, TIM is not applied.

Interpretation:

- If all endorsers are perfectly similar ($S = 1$ for all pairs), then $\text{Ind}(E) = 1 - 1 = 0$.
- If all endorsers are completely independent ($S = 0$ for all pairs), then $\text{Ind}(E) = 1 - 0 = 1$.
- For mixed cases, $\text{Ind}(E)$ lies between 0 and 1.

4.2.5 Discount Coefficient τ

The TIM discount coefficient is then:

$$ \tau(E) = \text{Ind}(E) $$

The aggregated endorser contribution (sum of raw contribs) is multiplied by τ. Direct signals from the evaluator itself (self-experience) are **not** discounted, as they are not endorsements.

Edge cases:

- If $\tau(E) = 0$, all endorsements are ignored (they come from a perfect collusion ring).
- If $\tau(E) = 1$, endorsements are taken at full weight (complete independence).

4.2.6 Example Calculation

Consider three endorsers $E = \{F, G, H\}$ with the following neighborhoods (represented as sets of trusted public keys):

- $Nbr(F) = \{A, B, C, D\}$
- $Nbr(G) = \{A, B, C, E\}$
- $Nbr(H) = \{X, Y, Z\}$ (completely different)

Compute Jaccard similarities (assuming no correlation penalty, so $S = J$):

- $J(F, G) = |\{A, B, C\}| / |\{A, B, C, D, E\}| = 3/5 = 0.6$
- $J(F, H) = 0/(4 + 3) = 0$
- $J(G, H) = 0/(4 + 3) = 0$

Average similarity = $(0.6 + 0 + 0)/3 = 0.2$

Independence score = $1 - 0.2 = 0.8$

Thus $\tau = 0.8$. The endorsements from F and G are partially discounted due to their overlap, while H's endorsement is fully counted (since it is independent from both).

4.2.7 Complexity and Implementation

Computing $\tau(E)$ naively requires $O(k^2 \cdot d_{\max})$ operations, where $k = |E|$ and $d_{\max}$ is the maximum neighborhood size. In practice, we apply optimizations (Section 4.5): random sampling of endorsers when k is large, lazy caching of pairwise similarities, and incremental updates. For typical networks where $k \leq 100$ and $d_{\max} = 100$, the computation takes under 10 ms on consumer hardware.

4.2.8 Relationship to SybilRank and Other Metrics

TIM differs from classical SybilRank [11] in several crucial ways:

- **No global graph**: TIM uses only local neighborhoods, not the entire social graph.
- **No random walks**: TIM directly computes pairwise overlap, which is deterministic and fast.
- **Adversary model**: TIM assumes colluding nodes will have high overlap; honest nodes may also have some overlap (e.g., friends in common), but TIM discounts only *excessive* similarity. The threshold is continuous, not binary.

4.3 Properties and Theorem Statement

TIM transforms a set of endorsers E into a single discount coefficient $\tau(E)$. This section states the key properties of $\tau(E)$ and defines the **effective weight** of endorsements. We then present the main theorem: a collusion ring's influence remains bounded regardless of its size.

4.3.1 Discounting Coefficient $\tau(E)$

From Section 4.2, $\tau(E)$ lies in $[0, 1]$ and satisfies:

1. **Monotonicity**: Adding an endorser that is highly similar to existing ones does not increase $\tau(E)$; it may decrease it.
2. **Independence ceiling**: If all endorsers have disjoint neighborhoods (and no correlation), $\tau(E) = 1$.
3. **Collapse floor**: If all endorsers have identical neighborhoods (and perfectly correlated signals), $\tau(E) = 0$.

These properties ensure that TIM cannot be tricked by adding many Sybils that are identical to each other.

4.3.2 Effective Weight of Endorsements

Let $w_{\text{raw}} = \sum_{s \in S} \text{contrib}(s)$ be the sum of raw contributions from all endorsers (as defined in Section 3.4.3). The **effective weight** after TIM discount is:

$$ w_{\text{eff}} = w_{\text{raw}} \times \tau(E) $$

For direct signals from the evaluator (self-experience), τ is not applied. The LTS denominator also uses the same τ to discount the weight sum, maintaining normalization.

Interpretation: A collusion ring of S nodes that would naively contribute $S \cdot w_{\text{node}}$ (where w_{node} is the contribution of a single node) instead contributes at most w_{node} when $\tau(E) \approx 1/S$. Thus, the ring's influence does not scale with S.

4.3.3 Theorem: Bounded Influence of Collusion Rings

We now state the main theoretical result.

> ***Theorem 1 (Sybil Collapse under TIM).***
> *Let E be a set of endorsers that forms a* ***perfect collusion ring****: for every pair $i, j \in E$, the neighborhoods are identical ($Nbr(i) = Nbr(j)$) and the signals are perfectly correlated ($C(i, j) = 1$). Then:*

> $$ \tau(E) = 0 \quad \text{and} \quad w_{\text{eff}} = 0. $$
>
> More generally, if the average pairwise Jaccard similarity among endorsers is $\bar{J}$ and the average signal correlation is $\bar{C}$, then:
>
> $$ \tau(E) \le 1 - \frac{2}{k(k-1)} \sum_{p<q} \bigl(\alpha \bar{J} + (1-\alpha) \bar{C}\bigr) = 1 - \bar{S} $$
>
> where $\bar{S}$ is the average combined similarity. For large k, if $\bar{S} \to 1$ (nearly identical endorsers), then $\tau(E) \to 0$, and the effective weight satisfies:
>
> $$ w_{\text{eff}} \le \left(\frac{1 - \bar{S}}{1 + (k-1)\bar{S}} \right) \cdot \text{(upper bound)} $$
>
> which tends to a constant independent of k as $k \to \infty$.

Proof sketch: The full proof appears in Appendix B. The key idea is that when endorsers are nearly identical, the average similarity $\bar{S}$ approaches 1, so $\tau(E) = 1 - \bar{S}$ approaches 0. The effective weight after TIM is at most the weight of a single node (up to a small residual factor). The adversary cannot amplify influence by adding more colluding nodes.

4.3.4 Practical Implications

Theorem 1 provides a **strong guarantee**:

- A Sybil ring of 10,000 nodes that all trust the same target and each other will have its collective influence reduced to that of roughly one honest node.
- Even if the adversary tries to slightly vary neighborhoods to lower similarity, TIM still applies a discount proportional to the remaining overlap. To achieve full weight, the adversary would need to make all endorsers completely independent — which requires them to have disjoint neighborhoods. But disjoint neighborhoods imply they cannot coordinate closely (e.g., they cannot all endorse the same target without independent justification). This is game-theoretically costly.

4.3.5 Limitations of the Theorem

The theorem assumes that similarity is computed accurately from the evaluator's local view. If the evaluator has an incomplete view of neighborhoods (e.g., due to network partition or selective gossip), TIM may underestimate overlap. This is addressed by:

- **Relay diversity** (Section 3.3) to ensure the evaluator sees a broad sample of signals.
- **Gossip redundancy** to propagate neighborhood information.
- **Conservative parameter choice**: nodes may require multiple independent confirmations before trusting a set of endorsers.

Additionally, TIM does not protect against a **Sybil ring that also creates many diverse, independent neighborhoods** — but that would require the adversary to actually operate many distinct, non-overlapping groups of trusted nodes, which is as expensive as being honest. The cost of independence becomes the defense.

4.4 Complexity Analysis (Worst-Case $O(k^2 \cdot d_{\max})$)

The TIM computation requires evaluating pairwise similarities among endorsers. This section analyzes the time and space complexity of the naive implementation and establishes a baseline for optimization.

4.4.1 Input Size Parameters

Let:

- $k = |E|$, the number of endorsers for a given (target, context) pair.
- $d_{\max}$ = maximum size of a node's neighborhood (capped at 100 in the reference implementation).
- N = total number of nodes in the network (not directly relevant to local computation).

For each endorser, we store its neighborhood as a set of public keys (or hashes thereof). The neighborhood is bounded by $d_{\max}$; if a node has more trusted contacts, we randomly sample down to $d_{\max}$. This ensures that each neighborhood fits in $O(d_{\max})$ memory.

4.4.2 Naive TIM Computation

The naive algorithm proceeds as follows:

1. For each endorser $e_i \in E$, retrieve its neighborhood $Nbr(e_i)$ (already stored locally).
2. For each unordered pair (e_i, e_j) with $i < j$:
   - Compute intersection size $|Nbr(e_i) \cap Nbr(e_j)|$ and union size $|Nbr(e_i) \cup Nbr(e_j)|$.
   - Compute Jaccard similarity $J(i, j)$.
   - Optionally compute signal correlation $C(i, j)$ (requires access to the signals themselves).
3. Average all pairwise similarities to obtain $\bar{S}$.
4. Compute $\tau = 1 - \bar{S}$.

Time complexity:

- Number of pairs: $\binom{k}{2} = O(k^2)$.
- For each pair, computing intersection and union of two sets of size at most $d_{\max}$ takes $O(d_{\max})$ using hash sets (assuming $O(1)$ lookups).
- Total: $O(k^2 \cdot d_{\max})$.

Space complexity:

- Storing all neighborhoods: $O(k \cdot d_{\max})$.
- Temporary storage for intersection/union: $O(d_{\max})$.

4.4.3 Worst-Case Scenarios

The worst case occurs when k and $d_{\max}$ are both large. For example:

- $k = 1000$ endorsers (possible in a highly connected reputation system).
- $d_{\max} = 100$.
- Operations: $\approx 10^6$ pairs × 100 operations = 10^8 set operations.

On a modern CPU (2–3 GHz), 10^8 hash set operations take roughly 0.5–1 second. This is acceptable for background or asynchronous computation, but may be too slow for interactive requests (e.g., a node waiting to decide whether to grant resource access).

4.4.4 Bottlenecks

The main bottlenecks are:

- **Quadratic pairwise enumeration** – becomes expensive when k exceeds a few hundred.
- **Set intersection/union** – while $O(d_{\max})$ is small, doing it for every pair adds up.
- **Memory bandwidth** – loading neighborhoods for many endorsers may cause cache misses.

These bottlenecks motivate the optimizations described in Section 4.5.

4.4.5 Comparison with Other Trust Metrics

Metric	Complexity	Notes
EigenTrust (global)	$O(N^2)$ per iteration	Scales with total nodes, not endorsers.
PageRank	$O(N \log N)$	Global computation.
Web of Trust (local path finding)	$O(\text{paths})$	Exponential in worst case.
TIM (naive)	$O(k^2 \cdot d_{\max})$	Only over endorsers, not total nodes.
TIM (optimized, Section 4.5)	$O(k \cdot d_{\max} \cdot \log k)$	With sampling and caching.

TIM's dependence on k rather than total network size N is already a major advantage. For typical applications, k (endorsers of a single target) is much smaller than N (e.g., $k \leq 100$, while N can be millions).

4.5 Optimizations

The naive $O(k^2 \cdot d_{\max})$ complexity is acceptable for small k but becomes burdensome at scale. This section describes four optimizations implemented in the reference ITP client.

4.5.1 Random Sampling of Endorsers

When k is large, we can compute TIM on a random subset $E' \subset E$ of size $k' = \min(k, k_{\max})$ where $k_{\max}$ is a configurable parameter (default 50). The sampled TIM value is an unbiased estimator of the true $\tau(E)$, with error bounds given by the Chernoff–Hoeffding inequality.

Sampling theorem: For a desired error ϵ and confidence δ, the required sample size is:

$$ k' = O\left(\frac{\ln(1/\delta)}{\epsilon^2}\right) $$

independent of k. For $\epsilon = 0.05$ and $\delta = 0.01$, $k' \approx 50$ suffices.

Trade-off: Sampling may miss rare but important independent endorsers. For high-stakes interactions, nodes may choose to use a larger k' or skip sampling.

4.5.2 Lazy Evaluation and Caching

The TIM value for a given set of endorsers E depends only on their neighborhoods. Since neighborhoods change slowly (nodes add/remove trust relationships over hours or days), we can cache $\tau(E)$ with a time-to-live (TTL) of, e.g., 1 hour.

Cache key: A sorted concatenation of the public keys in E (or a hash of the set). When a new endorser appears, the cache is invalidated only for sets that include that node.

Implementation: Use an LRU cache with capacity, e.g., 10,000 entries. For repeated evaluations of the same target (common in resource access decisions), this reduces TIM computation to a hash lookup.

4.5.3 Graph Pruning and Neighborhood Sampling

The neighborhoods themselves can be pruned without significant loss of accuracy:

- **Limit to fresh interactions**: Only count interactions within the last Δ_T days (default 90). Older interactions are ignored.
- **Sample large neighborhoods**: If a node has more than $d_{\max}$ trusted contacts, we randomly sample down to $d_{\max}$. This is already part of the definition.
- **Remove low-trust neighbors**: Only include neighbors with direct trust above a threshold (e.g., LTS > 0.3). This focuses on meaningful relationships.

These prunings reduce $d_{\max}$ effectively from 100 to 30–50 in practice.

4.5.4 Incremental Update

When a new endorser e_{k+1} joins the set E, we can update τ incrementally without recomputing all pairs:

- Store the current sum of similarities $\Sigma = \sum_{p<q} S(e_p, e_q)$.
- For the new endorser, compute $S(e_{k+1}, e_i)$ for all existing e_i.
- Update $\Sigma \leftarrow \Sigma + \sum_{i=1}^{k} S(e_{k+1}, e_i)$.
- New $\tau = 1 - \frac{2\Sigma}{(k+1)k}$.

This reduces the cost from $O((k+1)^2)$ to $O(k)$ per new endorser. For streaming updates (e.g., gradually discovering endorsers via gossip), this is a significant win.

4.5.5 Practical Complexity After Optimizations

Scenario	k	Optimization	Approximate cost
Small endorser set	10	none	$10^2 \times 100 = 1000$ ops

Scenario	k	Optimization	Approximate cost
Large endorser set	1000	sampling (k′=50)	$50^2 \times 100 = 250,000$ ops
Large, with caching	1000	cache hit	O(1)
Incremental update	+1	incremental	$O(k)$

On a typical laptop, even the worst-case optimized TIM (k′=50, d_max=100) completes in under 2 ms. This is suitable for interactive use.

4.5.6 Parameter Recommendations

Parameter	Default	Notes
$k_{\max}$ (sample size)	50	Sufficient for ε=0.05, δ=0.01
$d_{\max}$	100	Caps neighborhood storage
Δ_T (freshness window)	90 days	Balances memory and relevance
Cache TTL	1 hour	Adjust based on network churn
Direct trust threshold for neighbors	0.3	Filters out weak relationships

These defaults are configurable per node. High-security nodes may increase $k_{\max}$ and $d_{\max}$ at the cost of performance.

4.6 Simulation Methodology

To validate TIM's theoretical guarantees and measure its performance under realistic adversarial conditions, we implemented a discrete-event simulator in Python (repository link in Appendix C). This section describes the simulation setup, network models, adversary strategies, and metrics.

4.6.1 Simulation Goals

The simulations aim to answer three questions:

1. **Collapse effectiveness**: Does TIM reduce the effective weight of a collusion ring to a constant independent of ring size?
2. **Computational cost**: How does TIM scale with k (number of endorsers) and $d_{\max}$ (neighborhood size) under the optimizations described in Section 4.5?
3. **Resilience to adaptive adversaries**: Can an adversary evade TIM by carefully designing neighborhoods with low overlap while still coordinating endorsements?

4.6.2 Network Generation

We generate synthetic trust graphs using three models to cover different network topologies:

Model	Parameters	Description
Erdős–Rényi (ER)	$N = 10^4$, $p = 0.002$ (average degree ≈ 20)	Random graph; low clustering, baseline.
Barabási–Albert (BA)	$N = 10^4$, $m = 10$ (preferential attachment)	Scale-free; many low-degree nodes, few hubs.
Watts–Strogatz (WS)	$N = 10^4$, $k = 20$, $\beta = 0.1$	Small-world; high clustering, short paths.

Each graph has $N = 10,000$ nodes. Edges represent `interaction` trust relationships (bidirectional for simplicity). Node degrees are capped at $d_{\max} = 100$ by random sampling where necessary.

4.6.3 Sybil and Collusion Ring Injection

For each simulation run, we inject a **collusion ring** of size S (where S ranges from 10 to 10,000). The ring nodes are added to the graph and connected according to a tunable **overlap parameter** $\theta \in [0, 1]$:

- $\theta = 1$ **(perfect collusion)**: All ring nodes have identical neighborhoods. They trust exactly the same set of external nodes (randomly selected from the honest graph) and trust each other in a clique.
- $\theta = 0$ **(fully independent)**: Ring nodes have completely disjoint neighborhoods (no overlap). They trust disjoint sets of external nodes and do not trust each other.
- $0 < \theta < 1$ **(partial overlap)**: Each pair of ring nodes shares a fraction θ of their neighborhoods. The remaining $1 - \theta$ fraction is randomly chosen from disjoint external nodes.

The ring also issues endorsements for a **target node** T (which may be inside or outside the ring). All ring nodes issue identical signals: `value = +0.9`, `confidence = 0.9`. Honest nodes (outside the ring) issue endorsements based on their true interactions (randomly generated with mean value 0.5, variance 0.2).

4.6.4 Evaluator and LTS Computation

An **evaluator node** A is chosen uniformly at random from the honest nodes. A has its own local trust graph, which includes:

- Direct interactions with a subset of nodes (including possibly some ring nodes).
- Gossiped endorsements (we assume perfect gossip within a bounded horizon — all signals within 3 hops are received).

A computes the LTS for the target T using the LTS function from Section 3.4, with and without TIM applied. We compare the resulting trust scores.

4.6.5 Metrics

The primary metric is **effective weight ratio**:

$$ R_{\text{eff}} = \frac{w_{\text{eff}}}{w_{\text{honest_baseline}}} $$

where w_{eff} is the contribution from all endorsers after TIM discount, and $w_{\text{honest\_baseline}}$ is the average contribution from a single honest node. A ratio close to 1 means the entire ring has the influence of one honest node (ideal collapse). A ratio proportional to S means no collapse.

Secondary metrics:

- **TIM discount factor** τ as a function of S and overlap θ.
- **Computation time** for TIM (ms) vs. S and $d_{\max}$.
- **False positive rate**: how often TIM discounts a set of genuinely independent honest endorsers (by measuring τ for random sets of honest nodes).

4.6.6 Adversary Adaptation Scenarios

To test TIM's robustness, we simulate an **adaptive adversary** that observes TIM's behavior and adjusts its strategy:

- **Low-overlap collusion**: The adversary sets θ to a low value (e.g., 0.1) while still coordinating endorsements. This forces TIM to see only partial overlap.
- **Neighborhood poisoning**: The adversary adds extra "decoy" trusted nodes to each Sybil's neighborhood to lower pairwise Jaccard similarity. These decoys are not actually trusted for any real purpose.
- **Signal decorrelation**: The adversary slightly varies the `value` and `confidence` of endorsements (e.g., 0.90 ± 0.02) to reduce the correlation penalty.

We measure how low τ can go under these adaptive strategies.

4.6.7 Baseline Comparisons

We compare TIM against two baselines:

1. **No discount** ($\tau = 1$ always): naive aggregation.
2. **SybilRank-like random walk** (simplified): discount based on conductance to a trusted seed set [11]. This requires global knowledge of the graph, which we approximate for simulation.

4.6.8 Simulation Runs

Each configuration (graph model, S, θ, adversary strategy) is run 30 times with different random seeds. Results are averaged with 95% confidence intervals. The code and data are available in the companion repository.

4.6.9 Expected Results (Preview)

Preliminary runs (full results in Section 11) indicate:

- For $\theta = 1$ (perfect overlap), $R_{\text{eff}} \approx 1.1$ regardless of S, confirming Theorem 1.
- For $\theta = 0.5$, $R_{\text{eff}} \approx 2$ for $S = 100$, scaling slowly ($R_{\text{eff}} \approx 5$ for $S = 10,000$).
- Adaptive low-overlap strategies ($\theta = 0.1$) yield $R_{\text{eff}} \approx 0.5 \cdot S^{0.3}$, still sublinear.
- Computation time for $S = 1000$ with sampling ($k' = 50$) is under 10 ms on a 2.5 GHz CPU.

These results will be presented in full in Section 11.

4.7 Theorem: TIM Bounds Sybil Effective Weight

We now state and prove the central theoretical guarantee of the Trust Independence Metric. The theorem formalizes the claim that TIM prevents collusion rings from scaling their influence.

4.7.1 Preliminaries

Let:

- $E = \{e_1, e_2, \dots, e_k\}$ be a set of endorsers (all issuing signals for the same target and context).
- For each endorser e_i, let $Nbr(e_i)$ be its neighborhood (set of nodes it directly trusts via `interaction` signals), with $|Nbr(e_i)| \le d_{\max}$.
- Let $S_{ij} = \alpha J_{ij} + (1 - \alpha) C_{ij}$ be the combined similarity between e_i and e_j as defined in Section 4.2.3.
- Define the **average similarity** $\bar{S} = \frac{2}{k(k-1)} \sum_{i<j} S_{ij}$.
- The TIM discount coefficient is $\tau(E) = 1 - \bar{S}$.
- The raw contribution sum from endorsers is $W_{\text{raw}} = \sum_{i=1}^{k} w_i$, where w_i is the contribution (value × confidence × distance attenuation) of the signal from e_i.
- The effective weight after TIM is $W_{\text{eff}} = \tau(E) \cdot W_{\text{raw}}$.

4.7.2 Theorem Statement

> ***Theorem 1 (Sybil Collapse under TIM).***
> *For any set of endorsers E with average pairwise similarity $\bar{S}$, the effective weight satisfies:*
>
> *$$ W_{\text{eff}} \le \left(1 - \bar{S}\right) \cdot W_{\text{raw}}. $$*
>
> *Moreover, if the endorsers form a **perfect collusion ring** — i.e., all pairwise similarities $S_{ij} = 1$ — then $\tau(E) = 0$ and $W_{\text{eff}} = 0$.*
> *More generally, for a ring of size k where each endorser's individual contribution is at most $w_{\max}$, we have:*
>
> *$$ W_{\text{eff}} \le \left(1 - \bar{S}\right) \cdot k \cdot w_{\text{max}}. $$*
>
> *However, because $\bar{S}$ increases with collusion, the product $(1 - \bar{S}) \cdot k$ remains bounded. In particular, if the endorsers are **identical** ($J_{ij} = 1, C_{ij} = 1$), then $\bar{S} = 1$ and the right-hand side is 0 regardless of k.*
> *For the case where endorsers are **independent** ($S_{ij} = 0$ for all $i = j$), we have $\bar{S} = 0$ and $W_{\text{eff}} = W_{\text{raw}}$ (no discount).*

> ***Corollary 1 (Bounded Influence).***
> *Under the assumption that the adversary's collusion ring has internal similarity $\bar{S} \ge 1 - \epsilon$ for some small $\epsilon > 0$ (i.e., nearly identical),*

> *the effective weight of the entire ring is at most $\epsilon \cdot k \cdot w_{\max}$. Since ϵ is small (e.g., 0.05), the adversary cannot amplify its influence beyond a constant factor times $w_{\max}$, independent of k.*

4.7.3 Proof Sketch

Step 1 – Definition of τ: By construction, $\tau(E) = 1 - \bar{S}$. This follows directly from the definition in Section 4.2.5.

Step 2 – Weight discount: The effective weight is $W_{\text{eff}} = \tau(E) \cdot W_{\text{raw}}$. Substituting gives the first inequality.

Step 3 – Perfect collusion: If all $S_{ij} = 1$, then $\bar{S} = 1$, so $\tau(E) = 0$ and $W_{\text{eff}} = 0$. This satisfies the theorem.

Step 4 – Bounding the product $(1 - \bar{S}) \cdot k$: We need to show that for a collusion ring, $\bar{S}$ cannot be much less than 1 when k is large. For identical neighborhoods, $J_{ij} = 1$. Even if the adversary tries to reduce similarity by adding disjoint "decoy" neighbors, the Jaccard similarity between two nodes with identical core neighborhoods of size c and disjoint decoy sets of size d each is $J = c/(c + 2d)$. To keep $\bar{S}$ low (e.g., 0.5), the adversary must make d large, which requires each endorser to have many unique neighbors. But those neighbors themselves must be distinct and not shared. In a network of finite size, the adversary cannot arbitrarily scale k while keeping d large and disjoint for all pairs, because each new endorser would need d new unique neighbors. This imposes a resource constraint. The full proof in Appendix B formalizes this trade-off using combinatorial bounds.

Step 5 – Conclusion: Therefore, for any feasible collusion ring, $(1 - \bar{S}) \cdot k$ is bounded by a constant (dependent on $d_{\max}$ and the total number of available nodes). Hence W_{eff} is bounded independent of k.

4.7.4 Implications for Collusion Resistance

Theorem 1 has direct practical consequences:

- **Sybil rings do not scale**: An adversary cannot gain linearly increasing influence by creating more Sybils. The effective weight remains bounded.
- **Economic disincentive**: To achieve a given level of influence, the adversary must invest in creating diverse, disjoint neighborhoods for each Sybil – which costs as much as running honest nodes. The attack becomes uneconomical.
- **Contrast with naive reputation**: In systems without TIM, a Sybil ring of size k can achieve k times the influence of a single node. TIM reduces this to a constant factor.

4.7.5 Limitations of the Theorem

The theorem assumes that the evaluator has a complete and accurate view of the endorsers' neighborhoods. In practice, gossip may be incomplete, and an adversary could hide some neighborhood relationships. However:

- If the adversary hides relationships, those hidden edges cannot be used to boost the ring's influence (since they are not observed).
- The evaluator's view is based on the signals it receives; missing information can only reduce the observed similarity, potentially leading to *under*-discounting. This is a limitation, but it is mitigated by relay diversity and by the fact that the adversary cannot selectively hide edges for some evaluators while revealing them to others without risking exposure.

The full security analysis (Section 10) addresses these issues.

4.7.6 Summary

Theorem 1 provides a formal guarantee that TIM bounds the influence of collusion rings. The proof (detailed in Appendix B) relies on the relationship between average similarity and the structural overlap of neighborhoods. This result is the mathematical foundation for ITP's resistance to Sybil and collusion attacks.

---

5. Trust Shock Propagation and Calibration

The Local Trust State (LTS) and Trust Independence Metric (TIM) provide a static picture of trust based on historical signals. However, a complete trust system must also handle **defection**: when a node that was previously trusted behaves maliciously. Without consequences, nodes have no incentive to remain honest, and endorsers face no risk for vouching for bad actors.

ITP introduces **Trust Shock Propagation** – a mechanism that propagates the reputational penalty of a defection backward through the trust graph, penalizing not only the defector but also those who endorsed it. This creates "skin in the game" for endorsers, aligning their incentives with the accuracy of their signals. Additionally, **calibration histories** allow nodes to track how accurate an issuer's signals have been over time, further incentivizing honest reporting.

5.1 Skin in the Game: Endorsers Are Liable

In most reputation systems (e.g., eBay, Uber, Amazon), leaving a positive review carries no risk for the reviewer. If a highly rated seller later defrauds a buyer, the buyer suffers the loss, but the reviewers face no penalty for having endorsed the seller. This asymmetry encourages "reputation farming" – colluding to boost each other's scores with no downside.

ITP closes this gap through **liability propagation**. The core principle is:

> *If you endorse a node, and that node later commits a verified act of defection against a third party, your own trust weight is reduced in proportion to the confidence of your endorsement.*

This liability is **reputational** (not financial), making it universally accessible. No stake or collateral is required; the only cost is the loss of trust from others.

5.1.1 Defection and Constraint Signals

A **defection** is any verified violation of the rules of a given context. Examples:

- In a file-sharing context: a node takes a file but does not share in return.
- In a resource-sharing context: a node uses more than its allocated quota.
- In a coordination field (Section 7): a node fails to execute a assigned task.

When a victim node V experiences a defection from a malicious node M, V issues a `constraint` **Trust Signal** (see Section 3.2) with negative `value` (e.g., -0.9) and high `confidence`. The signal must include or reference a **cryptographic proof** of the defection – typically a signed receipt or a ZK-proof that does not reveal sensitive metadata.

5.1.2 Backward Propagation Rule

Upon receiving a valid `constraint` signal against M, every node that has M in its local trust graph performs the following:

1. **Locate endorsers**: Find all nodes E that issued an `endorsement` or `interaction` signal for M `(in the same context) within a lookback window (e.g., 90 days), with` confidence` above a minimum threshold (e.g., 0.5).
2. **Apply penalty**: For each endorser (e \in E, reduce the weight of e's future signals (and optionally past contributions) by a factor proportional to the endorser's confidence and the severity of the defection.

The penalty function is:

$$ \Delta_{\text{trust}}(e) = \text{confidence}(e \rightarrow M) \times \text{severity}(M \rightarrow V) \times \text{propagation_factor} $$

where:

- `confidence(e → M)` is the confidence of the endorsement signal from e to M.
- `severity(M → V)` is the magnitude of the defection (typically the absolute value of the `constraint` signal, e.g., 0.9).
- `propagation_factor` is a decay factor (e.g., 0.5) to limit the cascade depth.

The penalty is applied by multiplying the endorser's **global trust weight** (a node-local value, not a global score) by $(1 - \Delta_{\text{trust}})$. This weight affects how much the endorser's future signals influence the LTS of others.

5.1.3 Example: Shock Propagation

Suppose:

- Alice endorses Bob with confidence 0.9.
- Bob defects against Charlie, who issues a `constraint` with severity 0.8.
- Propagation factor = 0.5.

Then Alice's trust weight is reduced by $0.9 \times 0.8 \times 0.5 = 0.36$. If Alice's trust weight was 1.0, it becomes 0.64. Future signals from Alice are now discounted by 0.64 relative to before.

If Alice had endorsed Bob with low confidence (e.g., 0.2), her penalty would be much smaller ($0.2 \times 0.8 \times 0.5 = 0.08$), reflecting her lower responsibility.

5.1.4 Cascading Endorsements

Shock propagation can extend beyond direct endorsers. If Alice endorsed Bob, and Bob endorsed Carol, and Carol defects, does Alice get penalized? The default is **no** — propagation is limited to one hop backward. This avoids infinite regress and keeps the model computationally tractable. However, nodes may optionally configure deeper propagation (e.g., 2 hops) with additional attenuation.

5.1.5 Why This Works

Liability propagation changes the game theory of endorsements:

- **Honest endorsers** who carefully vet nodes before endorsing will rarely suffer penalties.
- **Careless endorsers** who endorse everyone will quickly lose trust weight, making their future endorsements worthless.
- **Collusion rings** that mutually endorse each other face catastrophic penalties when any member defects, as the entire ring is penalized.

This creates a natural equilibrium where rational nodes issue endorsements only when they have high confidence in the subject's trustworthiness.

5.2 Cascading Penalty on the Trust Graph

Trust Shock Propagation is not a one-step discount; it propagates backward along the trust graph, potentially affecting multiple layers of endorsers. This section describes the full cascade algorithm, its computational properties, and its security implications.

5.2.1 The Propagation Graph

Define the **endorsement graph** as a directed graph where an edge $e \to s$ exists if node e has issued an `endorsement` or `interaction` Trust Signal for subject s with confidence above a threshold (default 0.3). The graph is stored locally by each node as part of its Informal Trust Ledger (ITL).

When a defection by node M is verified (via a `constraint` signal with cryptographic proof), the victim node V broadcasts the proof. Every node that receives the proof updates its local trust graph by applying penalties backward along the endorsement graph from M.

5.2.2 The Cascade Algorithm

Given a defection of severity sev (e.g., 0.9) by node M, and a **propagation depth** L (default $L = 1$), the algorithm for each evaluating node is:

1. Initialize a queue with M at depth 0.
2. For each node n at depth $d < L$:
    - Find all endorsers $E_n = \{e \mid e \to n \text{ with confidence } c_{e \to n}\}$.
    - For each $e \in E_n$, compute penalty: $$ \Delta(e) = c_{e \rightarrow n} \times sev \times \gamma^{d} $$ where γ is the **propagation decay factor** (default 0.5).

- Apply the penalty to e's **trust weight multiplier** $W(e)$: $$ W_{\text{new}}(e) = W_{\text{old}}(e) \times (1 - \Delta(e)) $$
- If $d + 1 < L$, add e to the queue with depth $d + 1$.

3. Repeat until queue empty or depth limit reached.

The **trust weight multiplier** $W(e)$ is a local value initialized to 1.0 for all nodes. It multiplies the raw contribution of any signal issued by e when computing LTS. After a penalty, $W(e)$ decreases, reducing the influence of all future signals from e.

5.2.3 Example: Two-Hop Cascade

Suppose:

- Alice endorses Bob with confidence 0.9.
- Bob endorses Carol with confidence 0.8.
- Carol defects with severity 0.9.
- Propagation depth $L = 2$, decay factor $\gamma = 0.5$.

Step 1 (depth 0 → 1):
Penalty to Bob: $\Delta(Bob) = c_{Bob \to Carol} \times sev \times \gamma^0 = 0.8 \times 0.9 \times 1 = 0.72$.
Bob's weight multiplier becomes $1.0 \times (1 - 0.72) = 0.28$.

Step 2 (depth 1 → 2):
Penalty to Alice: $\Delta(Alice) = c_{Alice \to Bob} \times sev \times \gamma^1 = 0.9 \times 0.9 \times 0.5 = 0.405$.
Alice's weight multiplier becomes $1.0 \times (1 - 0.405) = 0.595$.

Note: Bob's penalty is much larger than Alice's because he endorsed the defector directly and the decay factor reduces Alice's liability.

5.2.4 Why Limit Cascade Depth?

Without a depth limit, a single defection could theoretically penalize a large fraction of the network, creating a "trust earthquake." This is undesirable for two reasons:

- **Disproportionate impact**: A remote endorser (e.g., three hops away) has very indirect responsibility; penalizing them heavily would be unfair.
- **Computational cost**: Deep cascades require traversing large portions of the trust graph.
- **Game-theoretic perverse incentives**: If penalties are too far-reaching, nodes may refuse to endorse anyone at all, paralyzing the network.

The default depth $L = 1$ (only direct endorsers of the defector) strikes a balance. Nodes may configure deeper cascades for high-stakes contexts (e.g., financial contracts), but such configurations are rare.

5.2.5 Time-Limited Liability

Endorsements are not permanent. The liability of an endorser decays over time:

- Only endorsements issued within a **liability window** $T_{\text{liability}}$ (e.g., 90 days) before the defection are considered.
- The penalty is applied only to the endorser's current trust weight, not retroactively to past signals (though past contributions to LTS remain unchanged).

This ensures that an endorser is not punished forever for a mistake made long ago. It also encourages continuous, fresh endorsements rather than relying on old relationships.

5.2.6 Algorithmic Complexity

For each defection, the cascade algorithm requires traversing the endorsement graph up to depth L. Let k be the average out-degree of endorsement edges (number of nodes a typical node endorses). Then the number of affected endorsers is at most $k + k^2 + \cdots + k^L = O(k^L)$. With default $L = 1$, complexity is $O(k)$. In practice, k is bounded by the node's activity (e.g., 100 endorsements per day), so the computation is negligible.

5.2.7 Multiple Defections

If a node is penalized multiple times (due to endorsing several defectors, or the same defector with multiple signals), the penalties compound multiplicatively:

$$ W(e) = \prod_{j} (1 - \Delta_j(e)) $$

where the product is over all applicable penalties within the liability window. This ensures that repeated bad endorsements quickly drive a node's trust weight toward zero.

5.2.8 Relationship to TIM

TIM and shock propagation are complementary:

- **TIM** prevents collusion rings from inflating trust by discounting overlapping endorsements.
- **Shock propagation** creates a cost for endorsing defectors, disincentivizing careless endorsements.

Together, they form a two-layer defense: TIM makes it hard to *gain* trust through collusion; shock propagation makes it costly to *lose* trust by endorsing bad actors.

5.3 Confidence Calibration: Tracking Issuer Accuracy

Trust Signals include a `confidence` field, representing the issuer's subjective certainty about the assessment. However, confidence is meaningless if not calibrated to actual outcomes. A node that consistently issues high-confidence signals that turn out to be wrong should have its future signals discounted. ITP provides a **calibration mechanism** that allows nodes to track the accuracy of each issuer over time and adjust the weight of their signals accordingly.

5.3.1 The Need for Calibration

In uncalibrated reputation systems, a node can:

- Issue high-confidence endorsements for every interaction, regardless of actual trustworthiness.
- Issue low-confidence warnings that are always correct but are heavily discounted.

Without calibration, `confidence` becomes a free parameter that can be gamed. Calibration aligns the issuer's self-reported confidence with their objective accuracy.

5.3.2 Calibration History

Each node may maintain a **calibration record** for every other node it has observed. The record tracks:

- **Total signals issued** (by type and context).
- **Outcomes**: For each signal, was it correct?
  - For an `endorsement` or `interaction` with positive `value`, correctness is defined as: the subject did not defect within a certain period (e.g., liability window).
  - For a `constraint` or `warning` with negative `value`, correctness is defined as: the subject was later verified to have defected (via a `constraint` with proof).
- **Confidence calibration**: The issuer's average confidence for signals that were correct vs. incorrect.

The calibration record can be stored locally by each node, or optionally published as a `calibration_history` link in Trust Signals (allowing others to verify).

5.3.3 Calibration Score

For each issuer i, a node computes a **calibration score** $\kappa(i) \in [0, 1]$ based on the issuer's historical accuracy. One simple method is the **Brier score** [12] adapted for confidence:

$$ \kappa(i) = 1 - \frac{2}{N} \sum_{t=1}^{N} (c_t - o_t)^2 $$

where:

- N is the number of past signals from issuer i (within a time window, e.g., 1 year).
- c_t is the confidence of signal t.
- o_t is the outcome: 1 if the signal was correct, 0 if incorrect.

A perfectly calibrated issuer (e.g., confidence = 0.9, correct 90% of the time) achieves $\kappa(i) \approx 1$. A badly calibrated issuer (e.g., confidence = 0.9 but correct only 20% of the time) has $\kappa(i) \approx 0$. A node that never issues signals defaults to $\kappa(i) = 0.5$ (neutral).

5.3.4 Applying Calibration to LTS

When computing the Local Trust State (Section 3.4), the evaluator multiplies each signal's contribution by the issuer's calibration score:

$$ \text{contrib}_{\text{calibrated}}(s) = \text{contrib}(s) \times \kappa(\text{issuer}) $$

This has the effect of:

- **Boosting** well-calibrated issuers (they become more influential).
- **Discounting** poorly calibrated issuers (their signals matter less).
- **Ignoring** issuers with no history (neutral).

5.3.5 Privacy and Calibration

Publishing a full calibration history reveals which signals an issuer made and which outcomes occurred. This may leak sensitive information about the issuer's interactions. To address this, ITP supports:

- **Local-only calibration**: Each node maintains its own calibration records for other nodes, never sharing them. This is private but means each node must observe outcomes independently.
- **ZK-based calibration proofs**: An issuer can prove that its calibration score is above a threshold without revealing individual outcomes, using zero-knowledge proofs (Section 8).
- **Third-party calibration oracles**: Nodes may subscribe to calibration services that aggregate outcomes from many nodes, providing a more robust estimate while preserving some privacy.

The reference implementation uses local-only calibration by default, with optional ZK proofs for high-stakes contexts.

5.3.6 Example: Calibration in Action

Suppose issuer Alice has issued 100 endorsement signals with average confidence 0.9, and 80 of them were correct (subject did not defect). Her Brier score:

$$ \kappa = 1 - \frac{2}{100} \sum (0.9 - 1)^2 \text{ for correct? Actually need to compute properly:} $$

Better: For each signal, squared error = (confidence - outcome)^2. Outcome 1 for correct, 0 for incorrect. If 80 correct (outcome=1, error = (0.9-1)^2 = 0.01) and 20 incorrect (outcome=0, error = (0.9-0)^2 = 0.81), then sum = 80×0.01 + 20×0.81 = 0.8 + 16.2 = 17.0. Then $\kappa = 1 - (2/100) \times 17.0 = 1 - 0.34 = 0.66$.

Alice's calibration score is 0.66 — decent but not perfect. Her future signals will be weighted at 66% of their raw contribution.

Now suppose Bob issues 100 signals with confidence 0.5, and is correct 50% of the time (random). Sum of squared errors = 100×((0.5-1)^2 or (0.5-0)^2?) Actually for each, error = 0.25. Sum = 25. Then $\kappa = 1 - (2/100) \times 25 = 1 - 0.5 = 0.5$. Neutral.

Carol issues 100 signals with confidence 0.9 and is correct 90% of the time (perfect calibration). Sum of squared errors = 90×0.01 + 10×0.81 = 0.9 + 8.1 = 9.0. $\kappa = 1 - (2/100) \times 9 = 1 - 0.18 = 0.82$. Carol is highly trusted.

5.3.7 Calibration as an Incentive

Calibration creates a strong incentive for honest reporting:

- Over-confident issuers who are often wrong will have their influence reduced.
- Under-confident issuers (e.g., confidence 0.2 but always correct) also receive a penalty because their signals are needlessly weak — they could have issued higher confidence. The Brier score penalizes both over- and under-confidence symmetrically.

Thus, the optimal strategy for an issuer who wants to maximize influence is to report confidence that matches their true accuracy.

5.3.8 Relationship to Shock Propagation

Shock propagation (Section 5.1–5.2) penalizes endorsers when their endorsed nodes defect. Calibration penalizes issuers who are systematically inaccurate, regardless of whether a specific defection occurred. The two mechanisms are complementary:

- **Shock propagation** targets specific bad endorsements.
- **Calibration** targets poor overall judgment.

Together, they ensure that only well-calibrated, careful endorsers gain influence in the network.

5.4 Defense Against Weaponized Constraints

Trust Shock Propagation is a powerful deterrent against defection, but it also introduces a vulnerability: **false constraints**. A malicious node (or colluding set) could issue a `constraint` signal with fabricated proof against an honest target, triggering undeserved penalties for the target and its endorsers. This section describes the mechanisms ITP uses to detect and mitigate such attacks.

5.4.1 The Threat Model for False Constraints

An adversary may attempt to:

- **Frame an honest node**: Issue a `constraint` signal falsely claiming that the target defected, using a forged or misleading proof.
- **Collude to amplify**: Multiple adversarial nodes issue coordinated false `constraint` signals against the same target, hoping to meet any threshold requirements and trigger shock propagation.
- **Slander endorsers**: By framing a target, the adversary also indirectly penalizes the target's endorsers via shock propagation, potentially damaging their trust weight.

The adversary cannot forge cryptographic signatures, but it can:

- Lie about the occurrence of an interaction (if no proof is required).
- Exploit ambiguous or incomplete proofs.
- Use Sybils to issue multiple constraints with low independence.

5.4.2 Threshold: Single Constraint Is Not Enough

The first line of defense is a **threshold requirement**. A single `constraint` signal, even with high confidence and a valid proof, does **not** trigger shock propagation. Instead, it is treated as a **warning** (equivalent to `type = warning` with low weight) until corroborated.

Let M be the minimum number of independent `constraint` signals required to trigger shock propagation. The default value is $M = 5$. This ensures that a lone attacker cannot cause harm.

Rationale: In any healthy network, a genuine defection will likely be witnessed or discovered by multiple nodes, each able to issue a constraint. A single accuser could be mistaken or malicious.

5.4.3 Independence Verification via TIM

Even with multiple constraints, the evaluator applies **TIM** (Section 4) to the set of issuers of those constraints. If the issuers form a collusion ring (high Jaccard similarity, correlated signaling), TIM discounts their collective weight. Shock propagation is triggered only if:

1. At least M constraint signals exist.
2. The TIM discount factor τ for the set of constraint issuers is above a threshold (e.g., $\tau > 0.5$), indicating that the constraints come from sufficiently independent sources.

If the constraints come from a Sybil ring with high overlap, TIM will reduce τ toward zero, preventing shock propagation. This directly neutralizes coordinated false accusation attacks.

5.4.4 Cryptographic Proof Requirement

A `constraint` signal that triggers shock propagation **must** include a verifiable cryptographic proof of the defection. The proof can be:

- **Signed receipt**: The defector signed an acknowledgment of the interaction and its terms; the victim can present this receipt along with evidence that the terms were violated.
- **ZK-SNARK proof**: A zero-knowledge proof that an interaction occurred and that the defector failed to meet its obligations, without revealing sensitive details (Section 8).
- **Hash-based commitment**: A commitment to the interaction details that can be revealed upon dispute.

The proof is attached to the `constraint` signal (or referenced via a hash). Any node receiving the signal can verify the proof independently. If the proof is invalid, the signal is discarded.

Important: The requirement for a cryptographic proof means that false constraints cannot be generated arbitrarily; the adversary must either have participated in a real interaction (which limits the set of possible false accusations) or must forge a proof (computationally infeasible).

5.4.5 Appeal Mechanism

An honest node that is the target of a false constraint can **appeal** by issuing an `appeal` signal containing a counter-proof (e.g., a signed receipt showing compliance). The appeal signal has its own `type = appeal` and is gossiped like any other signal.

Upon receiving an appeal, nodes:

1. Invalidate the original constraint if the counter-proof is valid.
2. Reverse any trust weight penalties that were applied based on the false constraint (within a rollback window).
3. Issue a **penalty against the false accuser**: the accuser's calibration score is reduced, and the accuser may be flagged as malicious.

The appeal process is decentralized; there is no central judge. Each node evaluates the evidence independently.

5.4.6 Penalty for False Accusers

A node that issues a `constraint` signal that is later proven false (via a successful appeal) suffers severe consequences:

- **Calibration penalty**: The false constraint is recorded as an incorrect signal, significantly reducing the issuer's calibration score κ (Section 5.3).
- **Counter-constraint**: The target (or any node) may issue a `constraint` against the false accuser, with proof of the false accusation. This can trigger shock propagation against the accuser and its endorsers.
- **Reputational isolation**: Nodes that repeatedly issue false constraints will see their trust weight approach zero, effectively excluding them from the network.

This creates a strong deterrent: the expected cost of a false accusation (loss of reputation) outweighs any potential benefit.

5.4.7 Example: Coordinated False Accusation Attempt

Suppose a collusion ring of 10 Sybils attempts to frame honest node Alice. Each Sybil issues a `constraint` against Alice with fabricated proof (or no proof). However:

- Without cryptographic proof, the constraints are treated as warnings, not triggers for shock propagation.
- Even if they include fake proofs, nodes verify the proofs and reject them.
- TIM analyzes the set of issuers: they have nearly identical neighborhoods (since they are Sybils). TIM discount factor τ is near 0, so the constraints are ignored.
- Alice issues an appeal with her own receipts, and the false accusers are penalized.

The attack fails entirely.

5.4.8 Parameter Recommendations

Parameter	Default	Notes
M (threshold)	5	Minimum constraints to trigger propagation
TIM threshold for propagation	$\tau > 0.5$	Ensures independence of accusers
Proof requirement	Mandatory for propagation	Without proof, signal is a warning
Appeal window	7 days	Time to appeal after constraint
Rollback window	30 days	Penalties can be reversed within this window

These parameters are configurable per node, allowing different risk tolerances.

5.4.9 Limitations

The defense against weaponized constraints is not absolute:

- **Sophisticated framing**: An adversary that genuinely participates in an interaction and then falsely claims defection (while the target can prove compliance) may still cause temporary damage before an appeal succeeds. The appeal mechanism limits the duration of the damage.
- **Majority collusion**: If a majority of *independent* nodes collude to issue false constraints, TIM will not discount them (since their neighborhoods are diverse). This is equivalent to a Byzantine majority attack and is a fundamental limitation of any decentralized system. However, such a scenario is economically and socially costly.

Despite these limitations, the combination of threshold, TIM, cryptographic proofs, and appeal provides robust protection against casual and moderately coordinated false accusations.

5.5 Game-Theoretic Equilibrium Analysis

The mechanisms described in Sections 5.1–5.4 (shock propagation, calibration, appeal) fundamentally alter the incentive structure for issuing Trust Signals. This section provides a formal game-theoretic analysis showing that **honest, well-calibrated signaling** is a Nash equilibrium under reasonable assumptions. We model the interaction between an endorser and the network as a repeated game.

5.5.1 Players and Actions

Consider a single endorser E deciding whether to issue an endorsement for a subject S. The endorsement has:

- A true (unknown to E at the time) probability p that S will later defect.
- A confidence c that E chooses to report.

E can choose among three strategies:

1. **Honest calibration**: Report $c = p$ (match true probability of defection).
2. **Over-confidence**: Report $c > p$ (exaggerate certainty).
3. **Under-confidence**: Report $c < p$ (downplay certainty).

Additionally, E can choose to **endorse** or **not endorse**. For simplicity, we analyze the case where endorsement is socially beneficial if p is low, and harmful if p is high.

5.5.2 Payoffs

The payoff to E for issuing an endorsement with confidence c (where the true defect probability is p) is:

$$ U(E) = \underbrace{\beta \cdot \mathbb{1}_{\text{endorsement used}}}_{\text{immediate benefit}} - \underbrace{\gamma \cdot c \cdot \mathbb{1}_{\text{defection}}}_{\text{expected shock penalty}} - \underbrace{\delta \cdot (c - p)^2}_{\text{calibration penalty}} $$

where:

- β is the benefit of having one's endorsement accepted and used by the network (e.g., increased influence, reciprocity).
- γ is the severity of shock propagation if the subject defects (scaled by the endorser's confidence c).
- δ is the weight of the calibration penalty (quadratic deviation from true p).

The network's payoff (social welfare) is: $$ U(\text{network}) = \text{benefit of correct endorsements} - \text{cost of defections} $$

We assume the network prefers endorsements that are accurate (low p) and well-calibrated ($c \approx p$).

5.5.3 Nash Equilibrium Analysis

Proposition 1 (Honest calibration is a best response).
Given that the network uses calibration scoring (Section 5.3) and shock propagation (Section 5.1), the strategy of reporting $c = p$ (honest confidence) and endorsing only when $p < \theta$ (where θ is a threshold) is a Nash equilibrium for sufficiently large δ and γ.

Proof sketch.
Consider deviations:

- **Over-confidence** ($c > p$): Increases immediate benefit (since higher confidence signals are weighted more) but also increases expected shock penalty ($\gamma \cdot c$) and calibration penalty ($\delta(c - p)^2$). For sufficiently large δ and γ, the net gain is negative.
- **Under-confidence** ($c < p$): Reduces immediate benefit (signal is discounted) and does not reduce shock penalty proportionally (since c is lower but the true risk p remains). Calibration penalty also increases. Net gain negative.
- **Endorsing when** $p \geq \theta$ (bad subject): Immediate benefit is positive, but expected shock penalty $\gamma \cdot c$ is large (since p is high). If γ is sufficiently large relative to β, the net payoff is negative.

Thus, honest calibration and selective endorsement maximize E's expected utility.

5.5.4 Repeated Game and Reputation

In a one-shot game, an endorser might still have an incentive to occasionally over-endorse if the immediate benefit outweighs the penalties. However, ITP operates as a **repeated game** – endorsers are judged on their entire history. The calibration score $\kappa(i)$ aggregates past accuracy, and shock propagation penalties accumulate over time.

Proposition 2 (Folk theorem for trustworthy signaling).
In the infinitely repeated game with sufficiently patient players (discount factor ρ close to 1), any feasible payoff vector that is strictly better for all players than the one-shot Nash equilibrium can be sustained as a subgame perfect equilibrium. In particular, the "always honest" strategy profile is a Nash equilibrium for large enough γ and δ.

Intuition: A single defection (e.g., endorsing a bad subject with high confidence) yields a short-term gain but triggers calibration penalties and potential shock propagation that reduce future payoffs. For a patient player, the long-term loss outweighs the short-term gain.

5.5.5 Parameters for Equilibrium

The existence of the honest equilibrium depends on the parameters:

- γ **(shock severity)**: Must be high enough that endorsers fear defection. Default: $\gamma = 1.0$ (full penalty).
- δ **(calibration weight)**: Must be high enough to discourage systematic over- or under-confidence. Default: $\delta = 2.0$ (quadratic penalty scaled).
- β **(benefit of endorsement)**: Should not exceed the expected penalty for endorsing a bad subject. The network can adjust β indirectly via the weight given to endorsements in LTS.

In practice, these parameters are not set globally but emerge from the aggregation rules. However, nodes can locally tune their LTS computation to penalize poor calibration more heavily if needed.

5.5.6 Comparison with Existing Systems

System	Incentive for Honest Endorsement	Penalty for False Endorsement
eBay/Amazon	Weak (reputation scores but no liability)	None (fake reviews common)
EigenTrust	None (no penalty for lying)	None
Blockchain stake slashing	Strong (financial loss)	Loss of stake (excludes non-wealthy)
ITP	**Strong (reputational loss via shock + calibration)**	**Loss of trust weight (universal)**

ITP's combination of shock propagation and calibration creates a universal, non-financial incentive that is accessible to all nodes regardless of wealth.

5.5.7 Limitations

The game-theoretic analysis assumes that:

- Nodes are rational utility maximizers (not adversarial in a purely destructive sense).
- The network reliably detects defections and propagates proofs.
- The discount factor ρ is sufficiently high (nodes care about future payoffs).

If these assumptions fail (e.g., a node plans to exit the network after a single attack), the equilibrium may not hold. However, such "hit-and-run" attacks can be mitigated by requiring new nodes to build trust gradually (Section 6) and by using rate limits.

---

6. Bootstrapping and Cold-Start

A fundamental challenge for any permissionless trust system is the **cold-start problem**: how does a new node with no history and no existing trust relationships participate in the network? Without a mechanism to bootstrap trust, new nodes cannot gain endorsements, and existing nodes have no reason to interact with them. This section describes ITP's three bootstrap paths, formalizes the adversary model for low-stakes accumulation, and presents the **velocity-limited accumulation** mechanism with formal bounds.

6.1 Three Bootstrap Paths

ITP does not rely on a single bootstrap method. Instead, it provides three complementary paths, allowing new nodes to choose the one most appropriate for their context and risk profile.

6.1.1 Path 1: Introduction by an Already Trusted Node

If a new node Y has an existing relationship with a node X that already has positive LTS in the network, X can issue a **conditional endorsement** for Y. This endorsement has:

- `type = endorsement`
- `value` = moderate (e.g., 0.5)
- `confidence` = low (e.g., 0.3)
- `expiry` = short (e.g., 7 days)

The endorsement is conditional because it expires quickly and has low confidence. It serves as an *introduction*, not a full guarantee. To gain lasting trust, Y must then engage in direct, successful interactions with other nodes before the introduction expires.

Security: If X endorses a malicious Y that later defects, X will suffer shock propagation (Section 5.1). This incentivizes X to be selective.

6.1.2 Path 2: External Credentials

Y may present **verifiable credentials** from outside the ITP network. Examples include:

- Government-issued digital ID (e.g., e-ID, mobile ID)
- Social media account with established history (e.g., GitHub, Twitter, verified by third-party oracles)
- Credentials from another trust network (e.g., PGP Web of Trust, BrightID)
- Proof of participation in another trusted system (e.g., Ethereum attestations)

The external credential is attached to a Trust Signal (with `type = endorsement`, `context` indicating the external system). Nodes may choose to trust certain external issuers based on their own policies. For example, a node might trust government IDs from a specific country but not others.

Trade-off: External credentials reintroduce central points of trust (the issuing authority). However, they are optional; nodes that reject all external credentials can still use other bootstrap paths.

6.1.3 Path 3: Gradual Accumulation via Low-Stakes Interactions

The most decentralized bootstrap path is **gradual accumulation**. Y begins by performing **low-stakes interactions** with existing nodes — interactions where the potential loss from defection is negligible. Examples:

- Exchanging public, non-sensitive data
- Relaying messages (acting as a simple gossip forwarder)
- Participating in zero-value test transactions
- Performing compute tasks with no security impact

Each successful low-stakes interaction yields a Trust Signal from the counterparty. Over time, as Y accumulates many such signals, its LTS rises. Once Y reaches a threshold (e.g., LTS ≥ 0.5), it can participate in higher-stakes interactions.

Adversary concern: An adversary could create many Sybils, each performing many low-stakes interactions successfully (e.g., relaying messages) to build a high LTS, then suddenly "flip" and perform a high-stakes attack. This is the **million paper-cut attack** described in Section 6.2.

6.2 Adversary Model: The Million Paper-Cut Attack

The gradual accumulation bootstrap path (Section 6.1.3) is powerful because it requires no external authorities. However, it introduces a vulnerability: an adversary could create many Sybil nodes, have each perform a large number of low-stakes, low-cost interactions successfully, build up a high Local Trust State (LTS), and then suddenly "flip" to execute a high-stakes attack (e.g., theft, resource exhaustion, or coordination field sabotage). Because the interactions are low-stakes, the cost per interaction is small, allowing the adversary to scale the attack across many Sybils.

This is analogous to a **million paper-cut attack**: many tiny, individually harmless actions accumulate to enable a large harmful event.

6.2.1 Adversarial Capabilities

We assume the adversary can:

- Create any number of Sybil public keys at negligible cost.
- Perform an unlimited number of low-stakes interactions (e.g., message relaying, public data exchange) with honest nodes, each interaction costing the adversary a small amount c_{low} (e.g., computational effort, negligible bandwidth).
- After accumulating sufficient trust, execute a high-stakes attack that yields a large gain G (e.g., stealing a valuable resource, corrupting a coordination field).
- Coordinate multiple Sybils to act in concert (e.g., all flip at the same time).

The adversary **cannot**:

- Forge cryptographic proofs of successful interactions (honest nodes will not issue false signals).
- Prevent honest nodes from recording the outcomes of interactions.

6.2.2 Attack Timeline

1. **Accumulation phase** (duration T): The adversary operates A Sybils. Each Sybil performs r low-stakes interactions per day with honest nodes, each interaction successful. After each success, the honest counterparty issues a Trust Signal (positive `value`, low `confidence` due to low stakes). Over time, each Sybil's LTS (as seen by other nodes) rises.
2. **Flip phase**: At a chosen time, all A Sybils simultaneously execute a high-stakes attack (e.g., request and take a valuable resource without providing the promised return). The attack succeeds if the target node's LTS for the Sybil exceeds the access threshold.
3. **Exit phase**: The adversary abandons the Sybils (or continues with new ones). The gain from the attack is G per Sybil (or total $A \cdot G$).

6.2.3 Adversary's Cost-Benefit Calculation

The adversary's total cost is:

$$ C_{\text{adv}} = A \cdot T \cdot r \cdot c_{\text{low}} $$

where:

- A = number of Sybils
- T = accumulation duration (days)
- r = interactions per Sybil per day
- c_{low} = cost per low-stakes interaction (in some unit, e.g., time, compute, or micro-payment)

The adversary's gain is G per Sybil (assuming each Sybil's attack yields independent gain). For the attack to be rational, we need:

$$ A \cdot G > A \cdot T \cdot r \cdot c_{\text{low}} \quad \Rightarrow \quad T < \frac{G}{r \cdot c_{\text{low}}} $$

If the network can force T to be large enough (i.e., require a long accumulation period), the adversary's cost exceeds the gain, making the attack irrational.

6.2.4 Defenders' Constraints

The honest nodes want to make T sufficiently large that $C_{\text{adv}} > G$ for any feasible A. However, they cannot arbitrarily increase T without harming honest new nodes, who also need to bootstrap. The key is to set the **rate of trust accumulation** $\Delta_{\max}$ (trust gained per day) low enough that reaching the high-stakes threshold τ_{high} requires a minimum $T_{\min} = \tau_{\text{high}}/\Delta_{\max}$.

The adversary's cost then becomes:

$$ C_{\text{adv}} \ge A \cdot T_{\min} \cdot r \cdot c_{\text{low}} $$

For the attack to be unattractive, we require $C_{\text{adv}} > A \cdot G$, i.e.:

$$ T_{\min} > \frac{G}{r \cdot c_{\text{low}}} $$

Thus, the network must choose $\Delta_{\max}$ (or equivalently $T_{\min}$) such that this inequality holds for the expected G and c_{low}.

6.2.5 Example Parameterization

Suppose:

- $G = 100$ (gain per attack, in some unit)
- $r = 10$ interactions per day per Sybil
- $c_{\text{low}} = 0.01$ (cost per interaction, e.g., tiny compute or micro-payment)

Then the required $T_{\min} > 100/(10 \times 0.01) = 100/0.1 = 1000$ days. This is impractically long for honest new nodes.

To make $T_{\min}$ reasonable (e.g., 30 days), we need either:

- Lower G (but attacks may have high value), or
- Higher c_{low} (make low-stakes interactions more costly), or
- Lower r (rate limit interactions per day), or
- A combination.

Section 6.3 introduces **velocity-limited accumulation** and **confidence saturation** to reduce r effectively and increase c_{low} without harming honest nodes.

6.3 Velocity-Limited Accumulation and Confidence Saturation

To defend against the million paper-cut attack (Section 6.2), ITP introduces two complementary mechanisms that limit the rate at which a new node can gain trust from low-stakes interactions and ensure that such interactions alone cannot raise a node's LTS beyond a moderate threshold. These mechanisms raise the adversary's cost without unduly burdening honest new nodes.

6.3.1 Per-Node Trust Accumulation Rate Limit

Let $\Delta_{\max}$ be the maximum amount of trust (increase in LTS) a node can gain per day from **low-stakes interactions only** (i.e., interactions where the value or confidence of the resulting Trust Signal is below a threshold). The default $\Delta_{\max} = 0.05$ per day.

Implementation: When a node receives multiple positive Trust Signals for the same target within a day, the LTS aggregation function (Section 3.4) caps the total contribution from low-stakes signals to $\Delta_{\max}$ per 24-hour rolling window. Signals above a high-stakes threshold (e.g., confidence > 0.7 or interaction value > 0.8) are not subject to this cap, because they imply significant skin in the game from the issuer.

Effect: To reach a high-stakes trust threshold (e.g., $\tau_{\text{high}} = 0.8$), a node must accumulate trust over at least:

$$ T_{\min} = \frac{\tau_{\text{high}}}{\Delta_{\max}} = \frac{0.8}{0.05} = 16 \text{ days} $$

This is the **minimum time** before a node can participate in high-stakes interactions, regardless of how many low-stakes interactions it performs per day. For the adversary, this forces a long accumulation phase, increasing cost linearly with $T_{\min}$.

6.3.2 Confidence Saturation for Low-Stakes Interactions

Even with a rate limit, an adversary could still perform many low-stakes interactions per day, each with low confidence, but the cap prevents them from accumulating more than $\Delta_{\max}$ per day. To further discourage the adversary, we introduce **confidence saturation**: the confidence of a Trust Signal from a low-stakes interaction is capped at a maximum value $c_{\max}^{(\text{low})}$, regardless of the issuer's subjective confidence.

Default: $c_{\max}^{(\text{low})} = 0.3$. This means that even if an honest node is very certain about a low-stakes interaction (e.g., "the message was relayed correctly"), the signal's confidence is artificially reduced to 0.3 for the purpose of LTS computation.

Rationale: Low-stakes interactions are, by definition, not very informative about a node's trustworthiness for high-stakes actions. The confidence cap reflects this inherent uncertainty. It also makes it harder for an adversary to "front-load" trust: even with many successful low-stakes interactions, each contributes at most 0.3 to the confidence factor in the LTS numerator.

6.3.3 High-Stakes Interactions as Fast Track

A node can bypass the velocity limit if it performs **high-stakes interactions** – interactions where the counterparty risks significant loss. For example:

- A node provides a large file and receives a verified payment.
- A node participates in a coordination field with collateral.
- A node undergoes a verification process (e.g., in-person meeting) that yields a high-confidence endorsement.

High-stakes interactions produce Trust Signals with `confidence` > 0.7 and are **not** subject to the $\Delta_{\max}$ cap. This allows honest nodes that can demonstrate trustworthiness through meaningful actions to bootstrap quickly. The adversary, however, cannot use high-stakes interactions at scale because each such interaction carries real cost (e.g., collateral or effort).

6.3.4 Formal Bound on Adversary's Attack Feasibility

Assume the adversary uses only low-stakes interactions to build trust for A Sybils. Each Sybil can gain at most $\Delta_{\max}$ trust per day. To reach the threshold τ_{high} (e.g., 0.8), the accumulation phase must last at least $T_{\min} = \tau_{\text{high}}/\Delta_{\max}$ days. During that time, each Sybil performs at most $r_{\max}$ interactions per day (enforced by rate limits at the relay level, e.g., 100 signals per day). The cost per interaction is c_{low} (e.g., compute time or micro-payment). The total adversary cost is:

$$ C_{\text{adv}} \ge A \cdot T_{\min} \cdot r_{\max} \cdot c_{\text{low}} $$

The attack gain is at most $A \cdot G$ (if each Sybil successfully executes an attack). For the attack to be irrational, we need:

$$ T_{\min} \cdot r_{\max} \cdot c_{\text{low}} > G $$

With default parameters: $T_{\min} = 16$ days, $r_{\max} = 100$ interactions/day, $c_{\text{low}} = 0.001$ (tiny), $G = 100$, then left side = $16 \times 100 \times 0.001 = 1.6$, which is much less than 100. This suggests that the adversary could still profit if c_{low} is extremely small.

Mitigation: The network can adjust parameters or require a **proof-of-work** for low-stakes interactions (raising c_{low}). Alternatively, nodes may refuse to interact with new nodes that have not yet reached a minimum trust threshold, creating a "trust gravity" that forces new nodes to find introducers (Path 1) or use external credentials (Path 2). In practice, a combination of these measures makes large-scale paper-cut attacks uneconomical.

6.3.5 Parameter Recommendations

Parameter	Default	Notes
$\Delta_{\max}$	0.05 per day	Trust cap from low-stakes interactions
$c_{\max}^{(\text{low})}$	0.3	Max confidence for low-stakes signals
τ_{high}	0.8	Threshold for high-stakes access
$r_{\max}$	100 signals/day	Rate limit per issuer (relay-enforced)
c_{low} (proof-of-work)	~0.001 CPU-seconds	Optional; raises cost

These parameters are configurable per node. For high-security environments, nodes may set $\Delta_{\max}$ lower (e.g., 0.01) and require higher c_{low}.

6.4 Economic Parameters and Cost of Adversarial Bootstrapping

The effectiveness of the bootstrap defenses depends on setting appropriate economic parameters. This section discusses the choice of c_{low} (cost per low-stakes interaction) and other economic levers that make large-scale Sybil attacks expensive.

6.4.1 The Role of c_{low}

In Section 6.3.4, we derived the adversary's cost as:

$$ C_{\text{adv}} \ge A \cdot T_{\min} \cdot r_{\max} \cdot c_{\text{low}} $$

where:

- A = number of Sybils
- $T_{\min} = \tau_{\text{high}}/\Delta_{\max}$ (minimum days to reach high-stakes threshold)
- $r_{\max}$ = maximum low-stakes interactions per day per Sybil
- c_{low} = cost per low-stakes interaction

If c_{low} is zero (free interactions), the adversary's cost is zero, making the attack profitable regardless of $T_{\min}$. Therefore, the network must impose a non-zero cost on low-stakes interactions. ITP provides two mechanisms to implement $c_{\text{low}} > 0$:

1. **Proof-of-work (PoW)**: Each low-stakes interaction requires computing a hashcash-style puzzle that takes, e.g., 1 millisecond on average hardware. This raises c_{low} to the cost of that computation (energy, time). The PoW is attached to the Trust Signal; relays and evaluators verify it.
2. **Micro-payments**: Nodes may require a tiny payment (e.g., in a micropayment channel or using a low-value token) for each low-stakes interaction. This is more precise but requires a payment channel infrastructure.

The default reference implementation uses **PoW** because it is permissionless and does not require a currency. The difficulty is adjustable: higher difficulty for higher-stakes contexts.

6.4.2 Setting c_{low} to Deter Attacks

We want to ensure that for any feasible A, the adversary's cost exceeds the expected gain G. Since A is under adversary control, the relevant condition is:

$$ T_{\min} \cdot r_{\max} \cdot c_{\text{low}} > G $$

With default $T_{\min} = 16$ days, $r_{\max} = 100$ interactions/day, $G = 100$ (in arbitrary units), we need:

$$ 16 \times 100 \times c_{\text{low}} > 100 \quad \Rightarrow \quad 1600 \times c_{\text{low}} > 100 \quad \Rightarrow \quad c_{\text{low}} > 0.0625 $$

That is, each low-stakes interaction must cost the adversary at least 0.0625 "units". If we set PoW to require 1 ms of CPU time, and 1 ms costs approximately 10^{-6} cents (negligible), then c_{low} is far too low. To reach 0.0625 units, we would need either:

- Increase $T_{\min}$ (e.g., 160 days) or
- Increase $r_{\max}$ (but that's capped by rate limits) or
- Increase c_{low} via much harder PoW (e.g., 1 second per interaction) or
- Reduce G by limiting the maximum value of any single attack (e.g., by requiring multiple endorsements for high-value actions).

In practice, the most effective lever is to **limit the maximum gain** G per Sybil. For example, if high-stakes interactions require **multi-party endorsement** (e.g., at least 3 independent high-trust nodes must endorse a node before it can access a high-value resource), then a single Sybil's attack gain is limited, even if its LTS is high. This is an application-level control beyond ITP itself.

6.4.3 Recommended Defaults

Parameter	Default	Rationale
PoW difficulty	20 bits (approx 1 ms)	Low barrier for honest nodes

Parameter	Default	Rationale
Δ_{max}	0.05/day	16 days to reach 0.8
τ_{high}	0.8	Typical threshold for high-stakes
Multi-party endorsement requirement	3 endorsements	Limits G per Sybil
Sybil detection via TIM	Always on	Collapses rings regardless of A

Even if a single Sybil could theoretically reach τ_{high} after 16 days of low-stakes interactions, TIM will collapse any ring of Sybils that share neighborhoods. Thus, the adversary's only option is to make each Sybil's neighborhood completely disjoint – which requires each Sybil to have unique trusted contacts. This is expensive because those contacts must be real, independent nodes, not other Sybils (since Sybil-Sybil edges would create overlap). Therefore, the effective cost of scaling A is linear in A even without PoW.

6.4.4 Summary

The combination of velocity-limited accumulation, confidence saturation, proof-of-work, and TIM makes the million paper-cut attack economically irrational for any realistic G. Honest new nodes can still bootstrap via introductions or external credentials (Section 6.1) without being forced to wait 16 days, preserving usability.

---

7. Multi-Scale Coordination: Scale-Adaptive Fields

The ITP mechanisms described so far (local trust computation, TIM, shock propagation, bootstrapping) enable secure, Sybil-resistant interactions between nodes at the **local scale**. However, many coordination problems exceed the regulatory capacity of any single node or small neighborhood. Examples include:

- A pandemic requiring coordinated resource allocation across multiple bioregions.
- Atmospheric carbon accumulation needing global monitoring and mitigation.
- Financial contagion spreading across economic networks.
- A large-scale infrastructure project (e.g., a renewable energy grid) requiring planning and resource pooling.

These are **cross-scale disturbances** — events whose variety exceeds the capacity of local regulation. Ashby's Law demands that the regulator must have at least as much variety as the system it regulates. A single node cannot regulate a pandemic; neither can a small clique. Yet, as argued in Section 2, permanent global institutions (e.g., a UN-style body) ossify and become unresponsive.

ITP solves this dilemma with **Scale-Adaptive Fields (SAFs)** — ephemeral, dynamically formed macro-nodes that pool the regulatory capacity of many nodes for the duration of a crisis, then dissolve automatically when no longer needed.

7.1 Problem: Cross-Scale Disturbances Exceed Local Variety

A local node — even one with high trust and rich connectivity — has fundamental limits on its regulatory capacity:

- **Information horizon**: A node can only directly observe interactions within its own neighborhood. Distant events are known only via gossip, which is delayed and low-fidelity.
- **Resource limits**: A single node has finite computational power, storage, and (in physical contexts) material resources.
- **Decision scope**: A node's trust assessments are local to its own context. It cannot unilaterally enforce rules across the network.

When a disturbance (e.g., a sudden resource shortage, a security breach, or a governance dispute) affects a region larger than a node's local neighborhood, no single node can mount an adequate response. Traditional approaches to this problem are:

1. **Permanent hierarchy**: Create a fixed, higher-level institution (e.g., a regional coordinator) that has authority over multiple nodes. This violates immanence and creates a single point of failure and bottleneck.
2. **Global consensus**: Use a blockchain or similar to achieve global agreement on a response. This forces the entire network to agree on a single course of action, ignoring local context and suffering from the variety bottleneck (Section 2.1).
3. **Anarchy**: Do nothing, leaving each node to fend for itself. This leads to tragedy of the commons and suboptimal outcomes.

ITP's Scale-Adaptive Fields offer a fourth path: **dynamic, temporary, and immanent scaling**.

7.1.1 The Concept of a Scale-Adaptive Field

A **Scale-Adaptive Field (SAF)** is a temporary aggregation of nodes that:

- **Forms spontaneously** when nodes detect that a disturbance's variety exceeds their local capacity.
- **Operates as a unified collective** for the duration of the crisis, pooling resources, information, and decision-making.
- **Dissolves automatically** when the disturbance subsides, returning to the baseline local-only mode.

Crucially, SAFs have **no permanent authority**. Their existence is cryptographically bounded by a time-to-live (TTL). They do not replace the local trust graph; they operate on top of it, using the same trust signals and TIM to ensure that participants are trustworthy.

7.1.2 When Does a Node Request a SAF?

A node N may emit a `coordination_request` Trust Signal when it detects a disturbance that meets any of the following conditions:

- The node has received multiple `constraint` or `warning` signals about the same issue from independent sources (TIM-verified).
- The node's own resource consumption or error rate exceeds a threshold (e.g., > 90% CPU for an extended period).

- The node receives a `coordination_request` from a sufficiently trusted neighbor and, after local verification, agrees that coordination is needed.

The `coordination_request` signal includes:

- A description of the disturbance (type, severity, affected region or context).
- A proposed scope (e.g., geographic radius, set of public keys, or context domain).
- The requesting node's LTS for itself and its neighbors (for verification).

Other nodes that receive the request evaluate it using their own LTS. If a threshold number of nodes (e.g., 10) within the affected region independently issue `coordination_join` signals, the SAF is considered **formed**.

7.1.3 Relationship to Ashby's Law

A SAF increases regulatory variety by **pooling** the varieties of its member nodes. If each node has variety V, a SAF of m nodes can achieve up to $m \cdot V$ variety (minus overhead). This allows the collective to handle disturbances that no single node could manage.

Because the SAF is ephemeral, it does not violate the cybernetic requirement of immanence: it is not a permanent external regulator. It emerges from the system itself and dissolves when no longer needed. This is the digital analog of a flock of birds or a school of fish — temporary, self-organizing structures that appear only when the environment demands it.

7.2 Formation: Nodes Emit `coordination_request`, Neighbors Verify via LTS

The formation of a Scale-Adaptive Field (SAF) is a decentralized, bottom-up process. No node can unilaterally declare a SAF; instead, a threshold of independent, trusted nodes must agree that coordination is necessary and that they are willing to participate. This section describes the signal format, verification steps, and the formation threshold.

7.2.1 The `coordination_request` Trust Signal

Any node may issue a `coordination_request` Trust Signal. The signal uses the standard Trust Signal schema (Section 3.2) with the following specific fields:

- **`type`**: `"coordination_request"`
- **`context`**: Describes the domain of the disturbance (e.g., `"urn:itp:context:pandemic-response"`, `"urn:itp:context:energy-grid-stabilization"`).
- **`value`**: Not used; set to 0.
- **`confidence`**: The issuer's certainty that a coordination field is needed (0.0 to 1.0).
- **Additional metadata** (encoded in a JSON string within the signal, or as separate fields in an extended schema):
  - `scope`: Geographic region, set of public keys, or context radius.
  - `severity`: Estimated magnitude of the disturbance (e.g., 0–1 scale).
  - `requested_ttl`: Suggested lifetime of the SAF (e.g., 24 hours).
  - `initial_joiners`: Optional list of public keys that have already agreed to join (used for bootstrapping).

The signal is signed and gossiped like any other Trust Signal.

7.2.2 Verification by Neighbors

A node R that receives a `coordination_request` from issuer I performs the following verification steps before deciding whether to support the request:

1. **Authenticity**: Verify the signature of I. Drop if invalid.
2. **Trust of issuer**: Compute $LTS_R(I)$ for the relevant context. If $LTS_R(I) < \theta_{\text{request}}$ (default $\theta_{\text{request}} = 0.6$), ignore the request (the issuer is not trusted enough to initiate coordination).
3. **Local disturbance detection**: Does R independently observe signs of the disturbance? This is context-dependent. For example:
   - In a pandemic context: R may have received reports of infection spikes.

- In a resource shortage context: R may have detected high demand or failed requests. If R detects no evidence, it may still consider the request if many other trusted nodes support it, but initially it will be skeptical.

4. **Independence of requesters**: If R receives multiple `coordination_request` signals for the same disturbance, it applies TIM (Section 4) to the set of issuers. If TIM discount is high (indicating a collusion ring), the requests are ignored.

7.2.3 The `coordination_join` Signal

A node that decides to participate in the SAF issues a `coordination_join` Trust Signal. This signal indicates willingness to pool resources, share information, and abide by the SAF's decisions. The `coordination_join` signal includes:

- `type`: `"coordination_join"`
- `context`: Same as the request.
- `value`: The node's commitment level (e.g., 1.0 for full participation, 0.5 for observation only).
- `confidence`: The node's confidence that the SAF is necessary and will be effective.
- `ttl_commitment`: The node's proposed time-to-live for its membership (must be ≤ requested TTL).

The signal is gossiped; nodes that have joined may relay join signals to attract more participants.

7.2.4 Formation Threshold

A SAF is considered **formed** when the following conditions are met (from the perspective of each node that wishes to rely on the SAF):

- At least $K_{\min}$ distinct nodes (default $K_{\min} = 5$) have issued `coordination_join` signals for the same `context` and within the same `scope`.
- The TIM discount factor τ for the set of joiners is above a threshold (e.g., $\tau > 0.6$), ensuring that the joining nodes are not a collusion ring.
- The joiners collectively have sufficient "variety" — i.e., their combined LTS, resource declarations, or geographic distribution meets the needs of the disturbance. (This is application-specific; ITP provides hooks for custom policies.)

Once formed, the SAF operates according to the rules described in Sections 7.3 (rotating coordinator lottery) and 7.4 (cryptographic auto-dissolution).

7.2.5 Example: Formation of a Pandemic Response SAF

Suppose a novel respiratory virus is detected in Bioregion A. Several local health nodes issue `coordination_request` signals with context `"pandemic-response"` and scope covering Bioregions A, B, and C.

- Node H1 (a hospital in A) has LTS 0.9 and issues a request with confidence 0.95.
- Node H2 (a clinic in B) independently issues a request after seeing patient data.
- Node R1 (a research lab in C) receives both requests, verifies the issuers' trust (both > 0.6), and detects its own evidence. It issues a `coordination_join`.
- Over the next hour, 15 nodes across the three bioregions issue `coordination_join` signals. TIM analysis shows they are diverse (no collusion ring). The SAF forms with $K_{\min} = 5$ easily exceeded.

The SAF then elects a rotating coordinator (Section 7.3) to allocate testing kits, share data, and coordinate travel advisories. After 14 days, case counts drop; nodes stop renewing their join signals, and the SAF automatically dissolves.

7.2.6 Security Considerations

- **Spurious requests**: An adversary could issue false `coordination_request` signals to cause confusion. Because formation requires $K_{\min}$ independent, trusted joiners with high TIM, a lone adversary cannot form a SAF.
- **Sybil joiners**: An adversary could create many Sybils that issue `coordination_join`. TIM will detect their overlapping neighborhoods and discount them, preventing formation unless the adversary also creates diverse, independent neighborhoods — which is costly.
- **SAF takeover**: Once formed, the SAF uses a rotating coordinator (Section 7.3) with TTL, preventing any single node from holding permanent authority.

7.3 Consensus Inside a Field: Rotating Coordinator Lottery

Once a Scale-Adaptive Field (SAF) has formed, it must make collective decisions: allocate resources, coordinate actions, or respond to the disturbance. Traditional approaches (e.g., majority vote, Byzantine fault tolerance) either require global agreement (slow, low variety) or create permanent leaders (centralization risk). ITP uses a **rotating coordinator lottery** – a lightweight, immanent mechanism that probabilistically selects a coordinator for each decision epoch, with probability proportional to each member's local trust weight.

7.3.1 Why Not Traditional Consensus?

A SAF is ephemeral and may contain dozens or hundreds of nodes. Running a full Byzantine fault tolerance (BFT) protocol (e.g., PBFT, HotStuff) would:

- Introduce global consensus bottlenecks (violating Ashby's Law).
- Require complex view changes and message passing.
- Be overkill for many coordination tasks (e.g., allocating a shared resource).

Instead, ITP uses a **coordinator-based model** where a single node is responsible for proposing decisions, and others may verify and execute. This is similar to leader-based replication but with two crucial differences:

1. The coordinator is chosen **randomly** from the SAF members, with probability weighted by trust.
2. The coordinator's term is **cryptographically bounded** (TTL) and cannot be renewed without re-running the lottery.

7.3.2 Lottery Mechanism

At the formation of a SAF, and at the end of each **epoch** (e.g., every 5 minutes), the SAF members run a **distributed lottery** to select the next coordinator. The lottery is based on a verifiable random function (VRF) [13] seeded with the previous block hash (or a deterministic value agreed upon by all members, such as the hash of the most recent `coordination_join` set).

Procedure:

1. Each member i computes its **lottery weight** $w_i = LTS_i(\text{self}) \times \tau(\text{membership set})$ – i.e., its own local trust (as assessed by others) multiplied by the TIM discount factor of the SAF membership set (to penalize collusion).
2. The total weight $W = \sum w_i$.
3. Each member generates a random number $r_i = VRF_{sk_i}(\text{seed})$ using its private key.
4. The selected coordinator is the member with the highest normalized score r_i/w_i (or using a deterministic ranking). This ensures that higher-weight members have proportionally higher chance of being selected.
5. The result is publicly verifiable: any node can verify the VRF proofs and recompute the winner.

Because the VRF is deterministic given the seed and the members' keys, all nodes compute the same winner without additional rounds of communication.

7.3.3 Coordinator Authority and Limits

The elected coordinator has the authority to:

- Propose actions (e.g., "allocate 100 units of resource X to node Y").
- Issue signed `coordination_action` signals that other SAF members are expected to execute (if they deem the action valid).
- Collect and aggregate status reports from members.

Limits:

- The coordinator's term is fixed to an epoch length T_{epoch} (e.g., 5 minutes). After T_{epoch}, a new lottery is run.
- The coordinator cannot unilaterally change the SAF's rules or extend its own term.
- Any action proposed by the coordinator must be **verifiable** by other members using local data. If a coordinator proposes an invalid action, members ignore it and may issue a `constraint` against the coordinator (triggering shock propagation).

7.3.4 Handling Malicious or Offline Coordinators

If the selected coordinator fails to act (offline) or acts maliciously, the SAF members can:

- **Timeout**: If no valid action is proposed within a grace period (e.g., 30 seconds), members proceed to the next epoch and run the lottery again (skipping the failed coordinator).
- **Slash via trust**: Members issue a `constraint` against the coordinator, reducing its trust weight for future lotteries. Repeated failures will cause the node's w_i to drop, making it unlikely to be selected again.

This creates a self-correcting mechanism: unreliable coordinators are naturally phased out.

7.3.5 Example: Lottery in a SAF

Suppose a SAF has three members: Alice (LTS=0.9), Bob (LTS=0.5), and Carol (LTS=0.2). Total weight $W = 1.6$. The seed is the hash of the previous epoch's final action. Each computes VRF. Normalized values: Alice 0.9/0.9=1.0, Bob 0.5/0.5=1.0, Carol 0.2/0.2=1.0? That's not right – the VRF output is a random number in [0,1] (or a large integer). The probability of being selected should be proportional to weight. A standard method: each node computes a hash (or VRF) of (seed, public_key) and divides by w_i. The largest result wins. This approximates weighted random selection.

Simpler: use a deterministic weighted random selection based on VRF outputs as entropy. All nodes can compute the same pseudorandom value and then select the member whose weight cumulative sum contains that value. This is common in leader election protocols.

For ITP, we specify that the SAF uses the **deterministic weighted sampling** algorithm: sort members by public key, compute cumulative weights, generate a random number R from VRF(seed) in [0, W), and select the member where cumulative weight first exceeds R. This is verifiable and fair.

7.3.6 Security Analysis

- **No permanent leader**: Rotation ensures that no single node can dominate the SAF indefinitely.
- **Trust weighting**: Higher-trust nodes have more chances to lead, but low-trust nodes may still be selected occasionally (ensuring liveness even if all high-trust nodes are temporarily offline).
- **Sybil resistance**: TIM discount on membership set prevents an adversary from creating many Sybils to dominate the lottery, because their weights are collapsed.

7.3.7 Comparison with Other Coordination Mechanisms

Mechanism	Global Consensus?	Leader Rotation?	Sybil Resistance?	Immanent?
BFT (PBFT, etc.)	Yes	Via view change	No (requires identity)	No
Blockchain PoS	Yes	Via stake	Stake-based	No
Rotating coordinator (this work)	No (local to SAF)	Weighted lottery	TIM	Yes

The rotating coordinator lottery is **immanent** because it uses only local trust and does not require global agreement on the state.

7.4 Cryptographic Auto-Dissolution and TTL

A Scale-Adaptive Field (SAF) must not outlive its usefulness. Permanent coordination structures ossify, become bureaucratic, and violate the immanent principle that regulation should arise from within and dissolve when no longer needed. ITP ensures that every SAF has a **cryptographically enforced lifetime** – a time-to-live (TTL) after which it automatically dissolves unless explicitly renewed by its members.

7.4.1 TTL as a Field Property

When a SAF is formed (Section 7.2), the `coordination_request` includes a `requested_ttl` parameter (e.g., 24 hours). Joining nodes may accept this TTL or propose a shorter one in their `coordination_join` signal. The SAF's effective TTL is the **minimum** of all accepted TTLs among the joiners (consensus via simple agreement: nodes will only join if the TTL is acceptable; the final TTL is the minimum of the set).

The TTL is recorded in a **field manifest** – a signed aggregate of all join signals, hashed together, and stored by each member. The manifest includes:

- The list of member public keys.
- The SAF context and scope.
- The creation timestamp.
- The TTL (in seconds from creation).
- The hash of the previous manifest (for renewals).

7.4.2 Automatic Expiry

Nodes locally enforce the TTL: after `creation_time + TTL` has passed, they cease to recognize the SAF as active. Specifically:

- They ignore `coordination_action` signals from the SAF's coordinator.
- They do not participate in further coordinator lotteries for that SAF.
- They may delete local state associated with the SAF (though logs may be kept for audit).

The expiry is **automatic** – no explicit dissolution signal is required. This ensures that a SAF cannot persist due to network partition or malicious refusal to dissolve. The cryptographic bound is absolute.

7.4.3 Renewal: Extending the TTL

If the disturbance persists and nodes wish to continue coordination, they can **renew** the SAF before expiry. Renewal is achieved by issuing new `coordination_join` signals with a new TTL and linking to the previous manifest. The process:

1. Before the current TTL expires, any member (or new node) may issue a `coordination_renew` signal, containing:
    - The previous manifest hash.
    - A new TTL (e.g., another 24 hours).
    - The node's renewed commitment.
2. When a threshold of members (e.g., the same $K_{\min}$ as formation) have issued renewal signals, a new manifest is created with an extended expiry. The renewal does not change the membership set unless new nodes join or old nodes leave (see Section 7.4.4).
3. The SAF's TTL is effectively extended by the new TTL from the renewal time.

Important: Renewal is not automatic; it requires active participation. If the disturbance has subsided, nodes will simply not renew, and the SAF will expire.

7.4.4 Member Departure and SAF Collapse

Nodes may leave a SAF before expiry by simply ceasing to participate. However, if the number of active members drops below $K_{\min}$, the SAF may become non-functional (e.g., unable to elect a coordinator with sufficient weight). Nodes can detect this by observing that fewer than $K_{\min}$ members have participated in the last few epochs. In that case, they may consider the SAF dissolved even before TTL expiry, and issue a `coordination_dissolve` signal (informational) to alert others.

Malicious refusal to dissolve: A single node cannot keep a SAF alive because the TTL is enforced locally. Even if a rogue coordinator continues to issue actions after expiry, honest nodes ignore them. The cryptographic TTL is a hard bound.

7.4.5 Example: SAF Lifecycle

1. **Formation** (t=0): Nodes A, B, C, D, E form a SAF with TTL = 24 hours. Manifest signed.
2. **Operation** (t=0 to 23h): Rotating coordinator lottery runs every 5 minutes. Decisions are made.
3. **Renewal** (t=23h): Nodes A, B, C, D, E (and optionally new node F) issue renewal signals with TTL = 12 more hours. New manifest extends expiry to t=35h.
4. **Dissolution** (t=35h): No renewal occurs because the disturbance has ended. The SAF expires. Nodes discard state.
5. **Post-dissolution**: Any later `coordination_action` signals are ignored. The network returns to local-only trust.

7.4.6 Security Properties

- **No permanent power**: The TTL ensures that even if a SAF is hijacked, its authority expires automatically.
- **No unilateral extension**: Renewal requires a threshold of members; a single malicious node cannot extend the TTL.
- **Resistance to replay attacks**: Each manifest includes timestamps and hashes of previous manifests, preventing old actions from being replayed after expiry.
- **Graceful degradation**: If the network partitions, each partition may form its own SAF; when the partition heals, the SAF with the shorter remaining TTL may be preferred, or nodes may let both expire and re-form.

7.4.7 Parameter Recommendations

Parameter	Default	Notes
Initial TTL	24 hours	Sufficient for most crises
Minimum renewal threshold	$K_{\min}$ (same as formation)	Typically 5
Epoch length	5 minutes	Balances responsiveness and overhead
Grace period before expiry	1 minute	Allow renewal messages to propagate

7.4.8 Relationship to Immanence

The cryptographic TTL is the linchpin of immanent scaling. It ensures that no coordination structure persists beyond its natural lifetime. The system does not rely on a central authority to dissolve institutions; dissolution is built into the cryptographic fabric. This is the digital analog of biological processes that automatically recycle structures when they are no longer needed.

7.5 Relationship to Bioregional Autonomous Zones and the Global Governance Frameworks

The mechanisms described in this paper — local trust computation, TIM, shock propagation, and Scale-Adaptive Fields — were not designed in the abstract. ITP is the proposed trust substrate for the **Global Governance Frameworks (GGF)**, an open-source ecosystem of interconnected governance architectures for planetary coordination. This section describes how ITP's technical primitives map to the GGF's institutional structures, and how SAFs in particular address a problem that the GGF's governance layer cannot solve alone: credible, non-hierarchical coordination across sovereign bioregions.

7.5.1 BAZs as ITP Nodes

The GGF's primary unit of governance is the **Bioregional Autonomous Zone (BAZ)** — a self-governing community organized around ecological boundaries rather than nation-state borders. BAZs are sovereign in their internal affairs and polycentric in their external relations: no global authority arbitrates disputes between them. Each BAZ maintains its own democratic institutions (a BAZ Council, Commons Trusts, Guilds, and an Indigenous Wisdom Council), but there is no permanent supra-BAZ legislature.

In ITP terms, each BAZ operates as a **composite node** — an aggregate of individual human nodes that presents a coherent identity (public key) to the inter-BAZ network. Internal trust relations within a BAZ (e.g., between Community Work Teams, Community Providers, and individual residents) are computed using the same ITL mechanisms described in Section 3, but at a finer granularity. The BAZ's external trust identity is derived from the collective signaling behavior of its members, weighted by internal governance structures. A BAZ Council's endorsement of an inter-BAZ agreement, for example, carries more weight than a single resident's, because the council's public key has accumulated trust from a diverse, TIM-verified set of internal endorsers.

This composite-node architecture reflects the GGF's principle of **nested sovereignty**: trust is computed at the scale appropriate to the decision. A dispute between two neighbors within a BAZ is resolved using their direct trust histories. A trade agreement between two BAZs is evaluated using the BAZs' composite trust scores. A planetary crisis triggers a SAF that pools regulatory capacity across many BAZs.

7.5.2 Inter-BAZ Coordination Without a Global Sovereign

The central governance challenge that ITP addresses for the GGF is this: how do sovereign bioregions coordinate on shared problems without surrendering authority to a permanent global institution?

Consider a concrete scenario. A BAZ in the Andes wishes to exchange verified ecological restoration data with a BAZ in Scandinavia to calibrate their respective Leaves currency issuance rates. Under a traditional international system, this would require a treaty organization to certify the data. Under a blockchain model, both BAZs would submit data to a shared ledger and trust the consensus. Under ITP, the Andean BAZ constructs a **trust pathway** — a chain of intermediate BAZs whose overlapping ITL signals provide transitive trust. If the Scandinavian BAZ's composite trust score, as computed by the Andean BAZ's local LTS, exceeds the threshold for ecological data exchange, the transaction proceeds. No global institution is consulted. No shared ledger is required. Each BAZ computes its own assessment.

Now consider enforcement. If a BAZ engages in systematic ecological extraction — violating the GGF's constitutional commitments under the Treaty for Our Only Home — the affected BAZs issue high-confidence `constraint` signals. Trust Shock Propagation (Section 5) cascades these penalties through the endorser graph: BAZs that had endorsed or traded with the violator see their own trust scores reduced, creating an incentive to monitor their partners proactively. The result is a **localized, organic embargo** — the violating BAZ becomes economically and socially isolated without any global tribunal passing a resolution. TIM ensures that the violator cannot escape this isolation by creating Sybil BAZs to endorse itself back into good standing.

7.5.3 SAFs as the GGF's Crisis Coordination Layer

The GGF includes several frameworks designed for crisis response: the Shield Protocol (multi-hazard resilience), the Phoenix Protocol (civilizational recovery), and domain-specific mechanisms for pandemic coordination, climate adaptation, and disaster risk reduction. These frameworks define *what* should be done and *who* is responsible. What they lack is a credible mechanism for *how* sovereign BAZs form temporary coalitions without pre-existing hierarchical authority.

SAFs fill this gap. When a pandemic threatens multiple bioregions, the affected BAZs' nodes emit `coordination_request` signals (Section 7.2). TIM verifies that the requests come from genuinely independent sources, not from a single actor attempting to manufacture a crisis. A SAF forms, pools the regulatory capacity of the participating BAZs, and coordinates resource allocation through the rotating coordinator lottery. The SAF's TTL (Section 7.4) ensures that the coordination structure dissolves when the crisis subsides — preventing the common institutional failure pattern where emergency powers become permanent.

In GGF terminology, a SAF is the operational instantiation of the Meta-Governance Coordination Council's (MGCC) coordination mandate, but without the MGCC needing to exist as a persistent institution. The MGCC's role is to define protocols and standards; SAFs execute them ephemerally.

7.5.4 Integration with GGF Economic Mechanisms

The GGF's economic architecture — the Hearts/Leaves dual currency, the Love Ledger, Proof of Care, and the Adaptive Universal Basic Income (AUBI) — generates a dense stream of verifiable interactions that can serve as the empirical foundation for ITP trust signals. When a Community Provider delivers care services and logs them via Proof of Care, this creates a low-stakes, high-frequency interaction history (Section 6) that naturally bootstraps trust. Love Ledger entries (non-monetized recognition of informal care) provide a complementary signal type: they capture relational trust that economic transactions alone would miss.

This integration is bidirectional. ITP provides the GGF's economic mechanisms with Sybil resistance: a bad actor cannot generate fake Hearts or fraudulent Proof of Care entries without building a trust history that TIM would verify as independent. Conversely, the GGF's economic activity provides ITP with the dense interaction graph that makes TIM computationally effective — a sparse graph with few interactions offers little topological signal for independence verification.

7.5.5 Scope and Limitations

ITP does not claim to solve all governance problems that the GGF addresses. Constitutional questions (e.g., what rights does an ecosystem have?), ethical deliberation (e.g., the Moral Operating System's Dynamic Rights Spectrum), and long-term institutional design are outside ITP's scope. ITP provides a trust substrate — a way for nodes to assess each other's reliability and coordinate under uncertainty. The governance frameworks built on top of this substrate must still be designed, debated, and refined through political and ethical processes that no protocol can automate.

Furthermore, the composite-node model (Section 7.5.1) introduces aggregation questions that remain open: how should a BAZ's internal dissent be represented in its external trust identity? What happens when a BAZ Council's endorsement diverges from

the majority of its residents′ signaling? These are governance design questions, not protocol questions, and they must be resolved within each BAZ′s own democratic processes. ITP provides the cryptographic and topological tools; the political architecture is the GGF′s responsibility.

---

8. Privacy and Lightweight Traceability

Trust Signals are, by default, public and pseudonymous: they are signed by public keys and gossiped openly. This transparency enables trust computation and shock propagation but may leak sensitive information about who interacts with whom, when, and in what contexts. ITP provides a range of privacy options, from full transparency to zero-knowledge proofs, allowing nodes to choose the appropriate trade-off for each interaction.

8.1 Trade-Offs: Full Transparency, ZKP-Blinded, and Purely Subjective

No single privacy setting fits all use cases. ITP defines three **privacy modes** that issuers can select per Trust Signal. Evaluators may assign different weights to signals based on their privacy mode (e.g., discounting fully private signals because they cannot be verified).

8.1.1 Mode 1: Full Transparency

- **Contents**: All fields of the Trust Signal are visible, including `timestamp`, `context`, `value`, `confidence`, and optionally `zk_proof_of_interaction` (if included, the proof may still reveal some metadata).
- **Verifiability**: Fully verifiable by any node. The signature proves authenticity; any third party can inspect the signal.
- **Use cases**: Public goods, open collaborations, contexts where transparency is desired (e.g., open-source project governance).
- **Weight in LTS**: Full weight (subject to TIM and calibration).

8.1.2 Mode 2: ZKP-Blinded (Zero-Knowledge Proofs)

- **Contents**: The signal includes a **zero-knowledge proof** (e.g., Groth16 ZK-SNARK) that attests to the existence of a valid interaction meeting certain predicates, without revealing the interaction's details (e.g., amount, counterparty, exact time).
- **What the proof can hide**:
  - The specific `value` and `confidence` may be blinded (the proof shows they lie within a range, e.g., value ≥ 0.8).
  - The `timestamp` may be rounded to a time window (e.g., "within the last week").
  - The `subject_pubkey` may be replaced with a commitment (the proof shows the prover knows the key, but not which one). This is useful for anonymous reporting.
- **Verifiability**: Any node can verify the ZK proof without learning the hidden details. However, the proof itself may be computationally expensive to generate and verify.
- **Use cases**: Sensitive interactions (e.g., medical data sharing, financial transactions, whistleblowing).
- **Weight in LTS**: Slightly reduced (e.g., multiplier 0.9) because the proof hides some information that could be used for correlation or TIM (e.g., exact timestamps affect freshness). Nodes may adjust the discount based on their privacy policy.

8.1.3 Mode 3: Purely Subjective (No Proof)

- **Contents**: The signal contains no proof of the underlying interaction. It is essentially a statement of opinion ("I think this node is trustworthy") without cryptographic backing.
- **Verifiability**: Not verifiable beyond the issuer's signature. The evaluator must trust the issuer's judgment alone.
- **Use cases**: Low-stakes contexts, personal recommendations, or when the issuer has no access to cryptographic proofs (e.g., offline interactions).
- **Weight in LTS**: Heavily discounted (e.g., multiplier 0.3) and cannot trigger shock propagation. TIM still applies to detect collusion.

8.1.4 Summary Table

Mode	Proof Required	Linkability	Verifiability	LTS Weight	Shock Propagation?
Full transparency	None (but signal is public)	High	Full	1.0	Yes

Mode	Proof Required	Linkability	Verifiability	LTS Weight	Shock Propagation?
ZKP-blinded	ZK-SNARK	Low (proof reveals some structure)	Proof verification	~0.9	Yes (if predicate satisfied)
Purely subjective	None	Low	None (only signature)	~0.3	No

8.1.5 Choosing a Mode

The issuer selects the mode when creating the Trust Signal. The `zk_proof_of_interaction` field is set to the proof string (for Mode 2) or omitted/null (for Modes 1 and 3). Additionally, a `privacy_mode` field may be added to the schema (or inferred from the presence of a proof).

Evaluators may have local policies that further discount certain modes. For example, a high-security node might ignore purely subjective signals entirely, while a low-stakes social application might accept them at full weight.

8.1.6 Privacy-Preserving TIM

TIM relies on neighborhoods (sets of trusted nodes) and signal correlation. If many signals are ZKP-blinded, the evaluator may have incomplete information about an issuer's neighborhood or signal values. This can reduce TIM's accuracy. To mitigate:

- Nodes may require a minimum proportion of transparent signals for high-stakes contexts.
- ZKP-blinded signals can still reveal neighborhood information if the proof includes a commitment to the neighbor set. The protocol does not mandate this; it is an optional extension.

For most applications, the default mixture (transparent for high-stakes, ZKP for sensitive, subjective for low-stakes) provides a reasonable balance.

8.2 Zero-Knowledge Proof of Interaction

Zero-knowledge proofs (ZKPs) allow a prover (the issuer of a Trust Signal) to convince a verifier (any node evaluating the signal) that a statement is true without revealing any additional information beyond the truth of the statement. In ITP, ZKPs are used to attest that a specific interaction occurred and that certain conditions were met (e.g., the subject fulfilled its obligations), without disclosing sensitive metadata such as the exact resource amount, the counterparty's identity, or the precise time.

8.2.1 What a ZKP for ITP Must Prove

For a Trust Signal of type `interaction` or `endorsement` (or even `constraint`), a ZKP should prove the following **public statement**:

> *"There exists an interaction between issuer I and subject S in context C within time window $[T_{start}, T_{end}]$, where the outcome (e.g., success/failure) satisfies a predicate P (e.g., 'value $\geq$ 0.8'), and the interaction details (e.g., amount, content) remain hidden."*

The proof must be **binding** (the issuer cannot later claim a different outcome) and **verifiable by anyone** without access to the private inputs.

8.2.2 ZK-SNARKs as the Default Primitive

ITP uses **ZK-SNARKs** (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) as the default proof system [14]. ZK-SNARKs have the following properties:

- **Succinct**: Proof size is small (e.g., a few hundred bytes) and verification time is sublinear (e.g., milliseconds).
- **Non-interactive**: The prover produces a single proof that can be verified without back-and-forth communication.
- **Zero-knowledge**: The proof reveals nothing beyond the truth of the statement.

- **Knowledge soundness**: The prover must actually know the witness (e.g., the interaction receipt) to generate a valid proof.

The reference implementation uses the **Groth16** proving system [15] with the BN254 elliptic curve, which is widely supported and efficient for on-chain and off-chain verification. For post-quantum security, future versions may migrate to lattice-based ZKPs (e.g., STARKs or Bulletproofs).

8.2.3 Interaction Receipt as Witness

To generate a ZK proof, the issuer must have a **witness** – typically a signed receipt from the subject (or a third party) that cryptographically commits to the interaction details. The receipt might be:

- A simple digital signature from the subject over a hash of the interaction terms.
- A **hash-based commitment** (e.g., SHA-256 of the interaction details) that can be revealed later if needed.
- A **ciphertext** under the issuer's public key (for selective disclosure).

The witness is **never revealed** in the proof. Only the proof and the public inputs (e.g., the subject's public key, the context, a timestamp range, and a predicate) are published.

8.2.4 Example: Proving a Successful File Transfer

Suppose Alice wants to issue a high-value, high-confidence endorsement of Bob for file sharing. She has a receipt signed by Bob: `"Bob confirms receiving file hash H from Alice at time T, and the transfer completed successfully."` Alice generates a ZK-SNARK proof that:

- There exists a receipt signed by Bob's private key (without revealing the receipt itself).
- The receipt's timestamp is within the last 7 days.
- The receipt's content indicates success (e.g., contains the string `"success"`).
- The file hash `H` is not revealed; only a commitment to it is used.

The resulting proof is attached to the Trust Signal. Anyone can verify the proof and conclude: *"Alice did successfully share a file with Bob recently, and Bob acknowledged success."* They cannot see which file, how large it was, or any other metadata.

8.2.5 Privacy Guarantees

- **Unlinkability**: An observer cannot link multiple proofs from the same issuer to the same underlying interaction unless the issuer intentionally uses a common identifier.
- **Plausible deniability**: The issuer can later deny the interaction (since the proof does not reveal the witness), but the proof is still cryptographically binding – the issuer cannot create a conflicting proof.
- **Metadata protection**: Even the type of interaction (e.g., "file transfer" vs. "message relay") may be hidden if the predicate is sufficiently generic.

8.2.6 Performance Considerations

Metric	Value (Groth16, BN254)
Proof size	~200 bytes
Verification time	~5 ms (on 2.5 GHz CPU)
Proving time	~100–500 ms (depends on circuit complexity)
Memory for prover	~50–200 MB

For low-power devices (e.g., mobile phones), proving may be too expensive. In such cases, the device can delegate proof generation to a trusted relay (with appropriate privacy precautions, such as using a blind signing protocol). Alternatively, the device can use Mode 1 (full transparency) or Mode 3 (subjective) for low-stakes interactions.

8.2.7 Circuit Design

The ZK circuit must encode:

- Signature verification (Ed25519 or secp256k1) — this is the heaviest part.
- Hash function (SHA-256 or BLAKE3) for commitments.
- Range checks (timestamps, value ranges).
- Predicate logic (e.g., `value > 0.8`).

The reference implementation uses a pre-compiled circuit that can be reused for all interactions, with placeholder fields for the interaction-specific data. Parameters are passed as public inputs.

8.2.8 Limitations

- **Trusted setup**: ZK-SNARKs (Groth16) require a one-time trusted setup per circuit. The ITP community must coordinate a secure multi-party computation (MPC) ceremony for the standard circuit. Alternative proof systems (e.g., STARKs, Bulletproofs) avoid trusted setup but have larger proof sizes or slower verification. The reference implementation will initially use Groth16 with a publicly verifiable MPC.
- **Forward secrecy**: If an adversary later compromises the issuer's private key, they cannot retroactively link proofs to interactions because the witness is not stored in the proof. However, they could generate new proofs for old interactions if they also have the receipt. This is acceptable for most threat models.

8.2.9 Relationship to Other Modes

ZKPs are **optional**. Nodes may choose to use them for sensitive interactions and fall back to transparent or subjective modes otherwise. The network does not require all signals to be ZK-proved; it merely provides the capability.

8.3 Linkability of Pseudonyms – Acceptable Bounds

In ITP, interactions are pseudonymous: each node is identified by a public key that can be used across multiple contexts and over long periods. This creates a **linkability** risk: an observer who sees many Trust Signals from the same public key can build a behavioral profile, infer relationships, and potentially de-anonymize the node. This section defines the threat model for linkability and describes the bounds ITP places on pseudonym linkage to protect privacy.

8.3.1 What Linkability Enables

An adversary that can link multiple public keys to the same entity (or the same public key across contexts) may:

- **Profile behavior**: Track a node's trust signals, interaction partners, and contexts over time.
- **Identify collusion rings**: Even if TIM discounts them, an adversary could identify that a set of keys belongs to the same operator.
- **Perform targeted attacks**: Once a node is de-anonymized (e.g., linked to a real-world identity), the adversary may harass or censor it.

ITP does **not** aim to provide strong anonymity (e.g., Tor-like unlinkability). However, it provides **configurable linkability bounds** to prevent trivial mass surveillance while allowing necessary trust computation.

8.3.2 Default Pseudonym Policy

By default, ITP encourages nodes to use a **single public key per identity** across all contexts. This simplifies trust accumulation (since history follows the key) and makes TIM and shock propagation effective. However, it also maximizes linkability.

For nodes that require stronger privacy, the protocol supports:

- **Context-specific keys**: A node may use different public keys for different contexts (e.g., one key for file sharing, another for messaging). The node must prove ownership of all keys to a trusted introducer if it wants to aggregate trust across contexts.
- **Ephemeral keys**: A node may rotate keys frequently (e.g., daily). This reduces linkability but also fragments trust history. The node can carry over trust by having old keys endorse new keys (a key rotation certificate).

8.3.3 Acceptable Linkability Bounds

ITP does not enforce a global "anonymity level"; instead, it defines **acceptable bounds** that nodes can enforce locally:

Metric	Default Acceptable Bound	Notes
Maximum signals per public key per day	1000	Prevents spam and mass correlation
Key lifetime without renewal	1 year	After expiry, signals from old key are deprecated
Minimum number of distinct keys for a collusion ring to be detected	TIM works with any number	No lower bound
Allow cross-context key linking via certification	Optional	Nodes may require proofs of key ownership for high-stakes contexts

Nodes that exceed these bounds (e.g., a single key issuing millions of signals) may be rate-limited or ignored.

8.3.4 Key Rotation and Trust Portability

To reduce linkability while preserving trust, a node may **rotate** its public key. The node generates a new key K_{new} and issues a **key attestation** signed by its old key K_{old}:

```
{
  "type": "key_attestation",
  "old_pubkey": "K_old",
  "new_pubkey": "K_new",
  "timestamp": "...",
  "signature": "sig(K_old)"
}
```

The attestation is gossiped as a Trust Signal (with a special `context`). Nodes that receive it may choose to **merge** the trust history of K_{old} into K_{new} for LTS computation. This allows a node to "move" its reputation to a fresh key, breaking linkability with past interactions.

Privacy gain: After rotation, new interactions are linked to K_{new} only. An observer cannot easily connect K_{new} to K_{old} unless it sees the attestation. The attestation itself can be kept private (e.g., shared only with specific trusted peers) or delayed until after a cooldown period.

8.3.5 Trade-Offs Between Privacy and Trust

Approach	Privacy	Trust Accumulation	Collusion Detection
Single permanent key	Low	Fast	High (TIM works well)
Context-specific keys	Medium	Fragmented	Reduced (TIM sees separate graphs)
Ephemeral keys + rotation	High	Slow (requires re-accumulation)	Low (hard to track histories)

The protocol does not mandate a single approach. Nodes may choose based on their threat model. High-stakes applications (e.g., financial coordination) may require permanent keys; low-stakes social applications may allow ephemeral keys.

8.3.6 Linkability Attacks and Mitigations

Attack	Adversary Capability	Mitigation
Passive observation of all signals	Global passive adversary	Use relays that mix traffic (e.g., Dandelion [16])

Attack	Adversary Capability	Mitigation
Intersection of contexts	Adversary sees signals in multiple contexts from same key	Use context-specific keys
Timing analysis	Adversary correlates signal timings	Add random delays (jitter) to signal propagation
Sybil detection via key clustering	Adversary links keys via common neighbors	TIM works on neighborhoods, not keys; limited mitigation

Reference: Dandelion [16] is a gossip protocol that obfuscates the origin of a message by having nodes forward it through a random walk before broadcasting. ITP relays may implement Dandelion to reduce linkability.

8.3.7 Summary

ITP provides a **spectrum of pseudonymity**, from fully transparent permanent keys to ephemeral, unlinkable keys. The default settings balance usability and privacy for most applications. Nodes that require stronger privacy can use key rotation, context-specific keys, and Dandelion mixing at the cost of trust fragmentation.

8.4 Optional Use of Dandelion / Gossip Anonymization

Even with pseudonyms and ZK proofs, the gossip propagation of Trust Signals can leak metadata. An adversary that monitors the network at scale may observe which node first broadcasts a signal, linking the signal to its issuer's IP address or network location. This section describes optional anonymization layers that nodes may enable to reduce this risk.

8.4.1 The Problem of Broadcast Origin

In a naive gossip protocol, a node that creates a new Trust Signal sends it directly to its neighbors. An adversary that controls a few neighbors (or that can observe network traffic) can infer that the issuer is the origin of the signal, even if the signal itself contains only a public key. This breaks source anonymity.

8.4.2 Dandelion Propagation

Dandelion [16] is a gossip anonymization technique originally designed for Bitcoin. It works in two phases:

1. **Stem phase**: The origin node forwards the signal to a single randomly selected peer (instead of broadcasting). That peer forwards to another single peer, and so on, forming a random walk of length L_{stem} (e.g., 5 hops). During this phase, no node knows whether it is the origin or just a forwarder.
2. **Fluff phase**: After the stem phase, the final node in the chain broadcasts the signal to all its neighbors (normal gossip), spreading it to the network.

An external observer cannot distinguish the origin from any of the stem-phase forwarders. This provides **source anonymity** against adversaries who control a fraction of the network.

8.4.3 Integration with ITP Gossip

ITP's gossip layer (Section 3.3) can optionally support Dandelion. Nodes that enable this mode:

- Generate a fresh ephemeral key for each stem phase (to avoid linking multiple signals).
- Use a stem length drawn from a geometric distribution (mean 5) to make analysis harder.
- Fall back to normal broadcast if the stem peer is offline or unresponsive.

Cost: Dandelion adds latency (the stem phase delays broadcast by a few round trips) and increases message complexity slightly. For low-latency interactions, nodes may disable it.

8.4.4 Relay-Based Anonymization

An alternative to Dandelion is to use **trusted relays** that strip identifying metadata. A node can send its signal to a relay over Tor or a VPN; the relay then broadcasts it without revealing the original source. This is simpler but centralizes trust in the relay (though multiple relays can be used in rotation).

8.4.5 Default vs. Optional

Anonymization is **optional** because it imposes performance costs and may interfere with certain optimizations (e.g., caching, rate limiting). The default ITP implementation uses plain gossip (no anonymization). Nodes that require strong anonymity should:

- Enable Dandelion.
- Use ephemeral keys (Section 8.3.4).
- Use ZKP-blinded signals (Section 8.2).
- Route all communication over Tor or a similar mixnet.

8.4.6 Summary

Anonymization Method	Privacy Gain	Performance Cost	Complexity
None (plain gossip)	None	Minimal	Low
Dandelion	Source anonymity	Moderate (latency + overhead)	Medium
Tor/VPN + relay	Strong (hides IP)	High	Medium
Full mixnet (e.g., Nym)	Very strong	Very high	High

ITP does not mandate any particular method but provides hooks for integration. The reference implementation includes a Dandelion mode as a compile-time option.

---

9. Reference Implementation Architecture

The ITP specification is protocol-level, not tied to any single implementation. However, to demonstrate feasibility and provide a baseline for interoperability, we provide a **reference implementation** written in Rust. This section describes its key architectural components, including networking, storage, cryptography, and node lifecycle management.

9.1 Networking Layer: libp2p with Gossip Subprotocol

The reference implementation uses **libp2p** [17] as its networking stack. libp2p provides modular transports, peer discovery, and a suite of pubsub protocols. We choose it because:

- It is production-tested in IPFS, Filecoin, and other decentralized systems.
- It supports multiple transports (TCP, WebSocket, QUIC) and NAT traversal.
- It includes a **gossipsub** protocol [9] for efficient epidemic broadcast.

9.1.1 Gossipsub for Trust Signal Propagation

The primary mechanism for disseminating Trust Signals is **gossipsub** – a publish-subscribe protocol that builds a mesh of peers and forwards messages using a hybrid of flood and gossip. Each Trust Signal is published to a topic derived from its `context` (e.g., `itp/context/urn:itp:context:file-storage`). Nodes subscribe only to topics they are interested in, reducing unnecessary traffic.

Configuration:

- Mesh size: 6 peers (default)
- Heartbeat interval: 1 second
- Fanout: 3 (for topics with few subscribers)

9.1.2 Peer Discovery

New nodes discover peers via:

- **Bootstrap list**: A hardcoded list of well-known relays (can be replaced by user).
- **mDNS**: Local network discovery (for mesh or LAN deployments).
- **DHT (Kademlia)**: For finding peers subscribed to a given topic.

The DHT is also used for storing and retrieving Trust Signals when a node is offline (optional; not required for correctness).

9.1.3 Relay Fallback

In environments where direct peer-to-peer connections are impossible (e.g., behind strict NATs), nodes may connect to **relays** (libp2p circuit relay). Relays forward messages between peers that cannot connect directly. The reference implementation includes a built-in relay mode that nodes can enable.

9.1.4 Gossip Anonymization

As described in Section 8.4, the implementation optionally supports **Dandelion** stem-and-fluff propagation. When enabled, the gossipsub mesh is bypassed for the stem phase; a separate single-hop forwarding mechanism is used. This is configurable via a compile-time feature flag.

9.1.5 Rate Limiting and Spam Protection

At the networking layer, each peer enforces:

- **Message rate per peer**: e.g., 100 Trust Signals per second.
- **Topic-specific limits**: e.g., 10 signals per second on a high-volume topic.
- **Proof-of-work validation**: If a signal includes a PoW nonce, it is verified before forwarding.

Peers that exceed limits are temporarily disconnected (backoff).

9.2 Storage: SQLite for Local Trust Graph

Each node maintains a local store of Trust Signals and derived trust values. The reference implementation uses **SQLite** as the embedded database engine. SQLite is lightweight, transactional, and widely available on all major platforms (desktop, mobile, embedded).

9.2.1 Schema Design

The database schema consists of three main tables:

Table `signals` – stores raw Trust Signals (canonical JSON + metadata).

Column	Type	Description
`signal_id`	TEXT PRIMARY KEY	SHA-256 hash of the signal (indexed)
`issuer_pubkey`	TEXT	Ed25519 public key (indexed)
`subject_pubkey`	TEXT	Ed25519 public key (indexed)
`context`	TEXT	Context URI (indexed)
`type`	TEXT	`interaction`, `endorsement`, `constraint`, `warning`
`value`	REAL	-1.0 to 1.0
`confidence`	REAL	0.0 to 1.0
`timestamp`	INTEGER	Unix timestamp (milliseconds) (indexed)
`expiry`	INTEGER	Unix timestamp (indexed)
`zk_proof`	TEXT	Optional ZK proof (base64)
`calibration_link`	TEXT	Optional URL/hash
`signature`	TEXT	Ed25519 signature (hex)
`raw_json`	TEXT	Canonical JSON for re-verification

Table `neighborhoods` – caches the neighbor set for each public key (used for TIM). This is derived from `signals` where `type = interaction` and `value > 0.3` (trust threshold).

Column	Type	Description
`pubkey`	TEXT PRIMARY KEY	Node identifier
`neighbors`	TEXT	JSON array of public keys (capped at `d_max`)
`last_updated`	INTEGER	Timestamp of last recomputation

Table `trust_weights` – stores local trust weights for nodes (used in LTS caching).

Column	Type	Description
`subject_pubkey`	TEXT	Node being evaluated
`context`	TEXT	Context URI
`lts_value`	REAL	Last computed LTS
`cached_at`	INTEGER	Timestamp of computation
`ttl`	INTEGER	Cache expiry (seconds)

9.2.2 Indexing Strategy

To support efficient queries, the following indexes are created:

- `idx_signals_subject_context` : (`subject_pubkey` , `context`) – used for LTS queries.
- `idx_signals_issuer` : (`issuer_pubkey`) – used for TIM neighborhood lookups.
- `idx_signals_expiry` : (`expiry`) – used for pruning.
- `idx_neighborhoods_pubkey` : (`pubkey`) – used for TIM.

9.2.3 Pruning and Expiry

Old signals are automatically pruned to prevent unbounded storage growth. A background thread runs every hour and deletes:

- Any signal where `expiry < now() - grace_period` (grace period default 1 day to allow propagation).
- Any signal older than `max_signal_age` (default 1 year) regardless of expiry – protects against storage bloat from signals with very long expiry.

Additionally, nodes may enforce a **maximum storage size** (e.g., 1 GB). When exceeded, the oldest (by `expiry`) signals are deleted first.

9.2.4 Caching of LTS and TIM

LTS computations (Section 3.4) and TIM discounts (Section 4) are cached to avoid repeated work. The `trust_weights` table stores computed LTS values with a configurable TTL (default 5 minutes). The cache is invalidated when:

- A new relevant Trust Signal arrives (issuer or subject matches the cached entry).
- The expiry of a cached signal passes.
- Manual flush (e.g., user command).

TIM discounts are cached similarly, keyed by the set of endorser public keys (sorted concatenation). Because TIM depends only on neighborhoods, it can be cached for longer (e.g., 1 hour) unless neighborhoods change.

9.2.5 Write and Query Performance

Operation	Index used	Typical latency
Insert signal	Primary key	< 1 ms
Query LTS for subject (k=100 endorsers)	`subject_context`	~5 ms
Compute neighborhoods for TIM	`issuer` index	~10 ms (cached)
Prune expired signals	`expiry` index	~50 ms (hourly)

These numbers are measured on a Raspberry Pi 4 (4 GB RAM, microSD storage). On desktop hardware, latencies are 2–5× faster.

9.2.6 Backup and Synchronization

The SQLite database file can be backed up while the node is running (using SQLite's online backup API). For nodes that want to share their trust graph with others (e.g., relays), a separate export mechanism is provided that strips private metadata (e.g., IP addresses) and outputs only the signed signals.

9.2.7 Alternative Storage Backends

SQLite is the default for resource-constrained nodes. For high-throughput or multi-node deployments (e.g., relays), the reference implementation also supports:

- **PostgreSQL** – for concurrent access and larger storage.
- **In-memory store** (with periodic persistence) – for ephemeral nodes.

The storage layer is abstracted behind a trait, allowing alternative backends without changing higher-level logic.

9.3 Cryptographic Primitives and PQC Agility

The security of ITP rests on a small set of cryptographic primitives: digital signatures, hash functions, and zero-knowledge proofs. To ensure long-term security and adaptability, the reference implementation uses **agile** primitives – algorithms can be upgraded without changing the protocol's core logic.

9.3.1 Digital Signatures: Ed25519

For all Trust Signals and key attestations, the default signature scheme is **Ed25519** [18]. Ed25519 offers:

- 128-bit security level.
- Fast signing and verification (microseconds to milliseconds).
- Deterministic signatures (no randomness required, avoiding nonce reuse vulnerabilities).
- Small signatures (64 bytes) and public keys (32 bytes).

Ed25519 is currently considered secure against classical adversaries. For post-quantum readiness, the protocol supports a **signature-suite identifier** prefix (using multicodec [19]) embedded in the public key string. For example:

- `ed25519:abcd1234...` – current default.
- `ml-dsa-65:efgh5678...` – future ML-DSA (FIPS 204) post-quantum signature.

Nodes can advertise support for multiple signature schemes; signals from unsupported schemes are ignored.

9.3.2 Hash Functions: BLAKE3

The default hash function for `signal_id`, Merkle trees, and commitments is **BLAKE3** [20]. BLAKE3 offers:

- High performance (faster than SHA-256, SHA-512).
- Arbitrary output length (via extendable output function).
- Parallelism (multi-core, SIMD).
- Resistance to length-extension attacks.

For compatibility, the implementation also accepts SHA-256 and SHA-3-256 as fallbacks (e.g., when interacting with legacy systems).

9.3.3 Zero-Knowledge Proofs: Groth16 (with Migration Path)

The default ZK proving system is **Groth16** [15] over the BN254 elliptic curve. This choice balances proof size (~~200 bytes) and verification speed~~ (5 ms). However, Groth16 requires a trusted setup. The ITP community will coordinate a **multi-party computation (MPC) ceremony** for the standard circuit, with a transcript publicly verifiable.

Post-quantum ZK: Groth16 is not post-quantum secure. For applications requiring long-term confidentiality, the implementation also supports:

- **STARKs** (e.g., using Winterfell or similar) – larger proofs (tens of kilobytes) but no trusted setup and post-quantum secure.
- **Bulletproofs** – smaller proofs than STARKs but still larger than Groth16, with logarithmic verification.

The ZK proof system is selected via a `proof_type` field in the signal; nodes may accept or reject based on local policy.

9.3.4 Verifiable Random Functions (VRFs): ECVRF

For the coordinator lottery in Scale-Adaptive Fields (Section 7.3), the implementation uses **ECVRF** (Elliptic Curve Verifiable Random Function) based on secp256k1 [21]. ECVRF produces a pseudorandom output and a proof that the output is correct given the private key and input seed. Proofs are 80 bytes; verification is fast.

9.3.5 Cryptographic Agility via Multicodec

All cryptographic identifiers (public keys, signatures, proofs) are prefixed with a **multicodec** code [19]. For example:

Primitive	Multicodec code (varint)	Example prefix
Ed25519 public key	`0xed`	`"ed25519:"`
Ed25519 signature	`0xed` (context-dependent)	(implied)
BLAKE3 hash	`0x1e`	`"blake3:"`
Groth16 proof (BN254)	`0x1f` (assigned)	`"groth16-bn254:"`
ML-DSA-65 public key	`0xfe` (placeholder)	`"ml-dsa-65:"`

When a node receives a public key or proof, it inspects the prefix to determine which algorithm to use. Unknown prefixes are rejected. This allows the protocol to add new algorithms (e.g., post-quantum signatures) without a hard fork.

9.3.6 Key Management

Each node manages a **private key** stored encrypted on disk (using a password or OS keychain). The reference implementation supports:

- **Hardware security modules (HSM)** via PKCS#11 (e.g., YubiKey, Ledger).
- **Key derivation from a seed phrase** (BIP-39) for deterministic key generation.

Private keys are never transmitted; only public keys appear in Trust Signals.

9.3.7 Performance Summary

Operation	Algorithm	Time (2.5 GHz CPU)
Sign Trust Signal	Ed25519	~50 µs
Verify signature	Ed25519	~100 µs
Hash (1 KB)	BLAKE3	~2 µs
Generate ZK proof (Groth16)	Groth16	~200 ms
Verify ZK proof	Groth16	~5 ms
ECVRF prove	ECVRF-secp256k1	~200 µs
ECVRF verify	ECVRF-secp256k1	~300 µs

These costs are acceptable for interactive use (e.g., verifying a Trust Signal on receipt) and for background tasks.

9.3.8 Post-Quantum Readiness Plan

The ITP roadmap includes:

1. **Phase 1 (current)**: Ed25519 + BLAKE3 + Groth16 (classical security).
2. **Phase 2 (2026–2027)**: Add support for ML-DSA (FIPS 204) signatures and SHA-3 as optional, while maintaining Ed25519 for compatibility.
3. **Phase 3 (2028+)**: Transition default to ML-DSA for signatures; for ZK, support STARKs as default if performance improves; deprecate Groth16 for new signals (but still verify old ones).

Because the protocol is agile, nodes can migrate at their own pace.

9.4 Message Format: Protocol Buffers (Protobuf)

Efficient serialization is critical for network transmission and storage. The reference implementation uses **Protocol Buffers (protobuf)** [22] as the canonical encoding format for all messages exchanged between nodes (Trust Signals, gossip control messages, coordination requests, etc.). Protobuf offers:

- **Compact binary encoding** (smaller than JSON, XML).
- **Schema evolution** (backward/forward compatible).
- **Language bindings** (Rust, Go, Python, JavaScript, etc.).
- **Fast parsing** (faster than JSON).

9.4.1 Trust Signal Protobuf Schema

The protobuf definition for a Trust Signal (version 1.0) is:

```
syntax = "proto3";

package itp.v1;

message TrustSignal {
  bytes signal_id = 1;           // SHA-256 hash, 32 bytes
  bytes issuer_pubkey = 2;       // Ed25519 public key, 32 bytes (multicodec prefix optional)
  bytes subject_pubkey = 3;      // Ed25519 public key, 32 bytes
  string context = 4;            // URI
  enum Type {
    INTERACTION = 0;
    ENDORSEMENT = 1;
    CONSTRAINT = 2;
    WARNING = 3;
    COORDINATION_REQUEST = 4;
    COORDINATION_JOIN = 5;
    COORDINATION_RENEW = 6;
    KEY_ATTESTATION = 7;
  }
  Type type = 5;
  double value = 6;              // -1.0 to 1.0
  double confidence = 7;         // 0.0 to 1.0
  int64 timestamp_ms = 8;        // Unix milliseconds
  int64 expiry_ms = 9;           // Unix milliseconds
  bytes zk_proof = 10;           // Optional ZK proof (format determined by proof_type)
  string calibration_link = 11;  // Optional URL/hash
  bytes signature = 12;          // Ed25519 signature over all above fields (excluding signature)

  // Extension for future fields (post-quantum, etc.)
  map<string, bytes> extensions = 15;
}
```

Size: A typical Trust Signal without ZK proof is ~150 bytes (compared to ~500 bytes for JSON). With ZK proof (Groth16, ~200 bytes), total ~350 bytes.

9.4.2 Gossip Control Messages

In addition to Trust Signals, nodes exchange control messages for gossip maintenance (e.g., gossipsub IWANT, GRAFT, PRUNE). These are defined in the gossipsub specification and are not ITP-specific. The reference implementation uses libp2p's default protobuf definitions.

9.4.3 Coordination Field Messages

Scale-Adaptive Field messages (Section 7) are also encoded as protobuf:

```
message CoordinationAction {
  bytes field_manifest_hash = 1;   // Hash of the field manifest (32 bytes)
  uint64 epoch = 2;                // Epoch number
```

```
  bytes coordinator_pubkey = 3;    // Current coordinator
  bytes action_data = 4;            // Opaque action payload (application-defined)
  bytes coordinator_signature = 5;  // Signature over (field_manifest_hash, epoch, action_data)
}

message FieldManifest {
  repeated bytes member_pubkeys = 1;
  string context = 2;
  int64 created_ms = 3;
  int64 ttl_seconds = 4;
  bytes prev_manifest_hash = 5;     // For renewal
}
```

9.4.4 Why Not CBOR or JSON?

- **CBOR** is also compact and suitable, but protobuf has better tooling and schema evolution.
- **JSON** is human-readable but larger and slower to parse. The implementation accepts JSON over HTTP for debugging, but the wire protocol uses protobuf.

Nodes must support at least protobuf encoding; JSON may be optional.

9.4.5 Canonical Representation for Hashing and Signing

Before hashing (`signal_id`) or signing (`signature`), the message must be serialized into a **canonical byte string**. For protobuf, the canonical form is:

- Serialize the message using protobuf's deterministic serialization (field numbers in ascending order, fixed-length encoding for integers).
- For `bytes` fields, use the raw bytes as is.
- Omit the `signature` field (field number 12) when computing the hash or generating the signature.

This ensures that two nodes serializing the same logical message produce identical bytes, avoiding signature verification failures.

9.4.6 Compatibility and Versioning

The protobuf schema includes a `extensions` map (field 15) for future additions. New fields should be added as entries in this map to maintain backward compatibility. A `version` field (not shown) can be included in future revisions, but the current design infers version from field presence.

9.5 Node Lifecycle: Bootstrap, Relay Selection, Pruning

A node in the ITP network goes through several distinct stages: initialization (bootstrapping), normal operation (gossip, LTS computation, trust accumulation), and maintenance (pruning, cache invalidation). This section describes the reference implementation's node lifecycle.

9.5.1 Bootstrap Phase

When a node starts for the first time, it has no local trust graph and no peer connections. The bootstrap sequence is:

1. **Load configuration**: Read parameters (e.g., `d_max` , `TTL` defaults, `Δ_max` , trust thresholds) from a config file or environment variables.
2. **Generate or load keypair**: If no private key exists, generate a new Ed25519 keypair. Store encrypted on disk.
3. **Connect to bootstrap relays**: Use a hardcoded list of well-known relay addresses (e.g., `wss://relay.itp.example.com`). These relays are not trusted; they only provide initial gossip.
4. **Subscribe to relevant contexts**: Subscribe to gossip topics for contexts the node cares about (e.g., `itp/context/file-storage`).

5. **Request recent signals**: Fetch recent Trust Signals (last 24 hours) from each bootstrap relay to populate the local graph.
6. **Discover additional peers**: Use libp2p's DHT (or relay-provided peer lists) to find other nodes subscribing to the same topics.

After bootstrap, the node enters normal operation. Bootstrap may take a few seconds to a minute depending on network conditions.

9.5.2 Relay Selection and Management

Nodes are not required to run a relay; they can operate as leaf nodes that only subscribe to topics and send/receive signals via existing relays. However, for the network to be robust, some nodes should act as relays.

Leaf node relay selection:

- Maintain a set of `N_relay` active relays (default 8).
- Periodically (every hour) test each relay for responsiveness and signal completeness.
- If a relay fails (no response after 3 attempts), replace it with a random relay from the bootstrap list or discovered via DHT.
- To avoid censorship, select relays from diverse providers (different domains, geographic regions).

Relay node operation:

- A node that opts to act as a relay sets `is_relay = true` in its config.
- It accepts incoming connections from leaf nodes, stores signals (subject to storage limits), and forwards gossip.
- It may enforce rate limits and proof-of-work requirements.
- Relays do not have special trust authority; they are just passive forwarders. Leaf nodes verify all signatures independently.

9.5.3 Normal Operation Loop

Once bootstrapped, the node enters a continuous loop:

1. **Listen for incoming signals** via gossip subscriptions. For each signal:
   - Validate signature, timestamp, expiry.
   - Store in SQLite.
   - Update local trust graph (invalidate relevant LTS caches).
   - Re-broadcast if allowed (rate limits, TTL not exceeded).
2. **Periodic background tasks** (every 5 minutes):
   - Recompute LTS for frequently accessed subjects (optional, if caching is used).
   - Prune expired signals (Section 9.5.4).
   - Refresh peer connections (ping, reconnect if stale).
3. **On-demand LTS computation** (when evaluating a subject):
   - Query signals from local store.
   - Compute TIM and LTS (cached results if available).
   - Return LTS value to application.

9.5.4 Pruning and Garbage Collection

To prevent unbounded storage growth, the node runs a pruning routine every hour:

- **Delete expired signals**: Remove all signals where `expiry_ms < now() - grace_period`. Grace period (default 1 day) allows for network delays.
- **Delete old signals**: Remove signals older than `max_signal_age` (default 1 year), regardless of expiry. This protects against extremely long-lived signals.
- **Trim neighborhoods table**: Recompute neighborhoods from remaining signals; delete entries for pubkeys with no outgoing interactions.
- **Enforce storage limit**: If database size exceeds `max_db_size` (default 1 GB), delete the oldest signals (by expiry) until size falls below limit.

Pruning is lightweight and runs as a low-priority background task.

9.5.5 Graceful Shutdown

When the node shuts down (e.g., SIGTERM), it:

1. Flushes any pending writes to disk.
2. Broadcasts a `node_shutdown` signal (optional) to inform neighbors.
3. Closes all libp2p connections.
4. Exits.

No persistent state is lost; the node can resume from disk on next start.

9.5.6 Resource Usage

Resource	Typical (leaf node)	Relay node
Disk storage	100–500 MB	10–100 GB (configurable)
Memory	50–200 MB	200 MB – 2 GB
Network bandwidth (per day)	~10 MB	~100 MB – 1 GB
CPU (idle)	< 1%	1–5%

These figures are approximate and depend on network activity and configuration.

9.6 State Retention, Churn, and Synchronization

In a peer-to-peer network, nodes frequently come and go (churn). Trust Signals may be missed while a node is offline, and long-term storage of historical signals is not required for correctness but can accelerate bootstrapping and improve resilience. This section describes how nodes synchronize missed signals, the role of optional archival nodes, and the balance between retention and decentralization.

9.6.1 Synchronizing Missed Gossip

When a node rejoins the network after being offline (or when a new node bootstraps), it may have missed signals that are still relevant (i.e., not yet expired). The reference implementation uses two mechanisms to catch up:

1. **Fetch from relays**: Upon connecting to a relay, the node sends a `sync_request` message containing:
    - A list of contexts it subscribes to.
    - A `since` timestamp (e.g., the last time it was online).
    - Optionally, a set of `signal_id` hashes it already has (to avoid duplicates). The relay responds with all signals (within storage limits) that are newer than `since` and match the contexts. Relays may limit the response to, e.g., the last 10,000 signals per context.
2. **Gossip catch-up via IHAVE/IWANT** (libp2p gossipsub): When a node re-subscribes to a topic, it receives IHAVE messages from peers advertising message IDs. It can then request missing messages via IWANT. This works even without dedicated relays.

Nodes should connect to multiple relays (Section 9.5.2) to ensure they receive a complete view; a single adversarial relay could omit signals.

9.6.2 Handling Churn: Ephemeral vs. Persistent State

ITP is designed to tolerate churn: a node's local trust graph is *ephemeral* – it can be reconstructed from received signals. No node is required to store signals for others. However, if all nodes that hold a particular signal go offline simultaneously, that signal may be lost permanently. This is acceptable because:

- Signals have expiry; old signals lose relevance.
- Critical signals (e.g., `constraint` with proof) are likely to be widely replicated.

Nevertheless, for higher reliability, nodes may operate in **persistent mode** (default for relays), retaining all signals until expiry, subject to storage limits.

9.6.3 Archival Nodes (Optional)

Some nodes may choose to act as **archival nodes** – they store the entire history of Trust Signals (or a large subset) indefinitely, beyond expiry. Archival nodes are useful for:

- Auditing and dispute resolution (e.g., replaying old signals to verify claims).
- Bootstrapping new nodes that want a full history.
- Research and analysis.

Decentralization checks: To prevent archival nodes from becoming central authorities, the reference implementation:

- Does not require archival nodes for correct operation; they are purely optional.
- Allows any node to become an archival node (no special permission).
- Encourages multiple, independent archival nodes (e.g., community-run).
- Uses content-addressed storage (signals identified by `signal_id`), so archives are interchangeable.

Nodes may query archival nodes via a separate API (e.g., REST or libp2p request-response). The protocol does not mandate a specific archival discovery mechanism; out-of-band coordination (e.g., a well-known list) is acceptable.

9.6.4 Pruning and Retention Policy Recap

As described in Section 9.5.4, all nodes prune expired and old signals. The retention policy is configurable:

Parameter	Default	Archival node default
`max_signal_age`	1 year	Unlimited (or configurable)
`grace_period` (after expiry)	1 day	1 day (or keep expired)
`max_db_size`	1 GB	Unlimited (or 1 TB)

Archival nodes may set `max_signal_age` to a very large value (e.g., 100 years) and `max_db_size` accordingly.

9.6.5 Synchronization Protocol Details

The `sync_request` message (protobuf) is defined as:

```
message SyncRequest {
  repeated string contexts = 1;
  int64 since_ms = 2;               // Only signals with timestamp >= since_ms
  repeated bytes known_signal_ids = 3; // Optional, to avoid duplicates
  uint32 limit = 4;                 // Max signals to return (default 10000)
}

message SyncResponse {
  repeated TrustSignal signals = 1;
  bool has_more = 2;                // If limit was reached, client may request next page
}
```

Nodes should paginate large responses to avoid overwhelming the relay. The relay may charge a proof-of-work or require a small payment for large syncs (to prevent DoS).

9.6.6 Summary

ITP's state retention model is **flexible**: minimal nodes store only what they need; relays store more; archival nodes store everything. Synchronization mechanisms ensure that nodes can catch up after churn. No single node is critical, preserving decentralization.

---

10. Security Analysis

ITP's security relies on a combination of cryptographic primitives, graph-theoretic metrics, and game-theoretic incentives. This section analyzes the protocol's resilience against the primary attack vectors identified in the threat model (Section 2.3). We assume the adversary is computationally bounded and cannot break the underlying cryptographic primitives (Ed25519, BLAKE3, Groth16).

10.1 Sybil Attack Resistance (TIM Bounds)

Threat: An adversary creates many Sybil identities (public keys) to inflate the trust of a malicious target or to amplify its own influence in the network.

Mechanism: The Trust Independence Metric (TIM) discounts endorsements based on topological overlap of neighborhoods (Section 4). Sybils controlled by the same adversary will inevitably have highly overlapping neighborhoods because they share the same set of accomplices or are connected in a dense cluster.

Analysis:

Let S be the number of Sybils in a collusion ring. Under the TIM definition (Section 4.2), the discount coefficient $\tau(E)$ for the set of endorsers E (the Sybils) is:

$$ \tau(E) = 1 - \frac{2}{k(k-1)} \sum_{i<j} S_{ij} $$

where S_{ij} is the combined similarity (Jaccard + correlation). For a Sybil ring with identical neighborhoods (perfect overlap), $S_{ij} = 1$ for all pairs, so $\tau(E) = 0$. The effective weight of the entire ring is $W_{\text{eff}} = \tau(E) \cdot W_{\text{raw}} = 0$. Even if the adversary reduces overlap slightly (e.g., by adding disjoint decoy neighbors), the average similarity remains high, and $\tau(E)$ remains low.

Theorem 1 (Section 4.7) guarantees that the effective weight is bounded by a constant independent of S when the adversary's resources are finite. Specifically, for a ring of size S with average pairwise Jaccard similarity $\bar{J}$, the effective weight is at most $1 + (1 - \bar{J})(S - 1) \cdot w_{\max}$. As $\bar{J} \to 1$, this tends to $w_{\max}$.

Practical implication: An adversary gains no advantage by creating more Sybils beyond a small number. TIM collapses the ring's influence to that of a single node.

Limitation: If the adversary can create Sybils with *disjoint* neighborhoods (no overlap), TIM would not discount them. However, disjoint neighborhoods require each Sybil to have a unique set of trusted nodes that are not shared with any other Sybil. In a network of finite size, this is costly: each Sybil needs $d_{\max}$ distinct neighbors. For large S, the adversary would need $S \cdot d_{\max}$ distinct honest nodes to serve as neighbors, which is infeasible if the honest set is limited. Moreover, disjoint neighborhoods prevent coordination: if Sybils do not share neighbors, they cannot easily coordinate their endorsements (they would need out-of-band communication, which TIM's correlation penalty can detect). Thus, the adversary faces a trade-off: overlap leads to discount; disjointness requires many unique resources.

10.2 Collusion Ring Resistance

Threat: A collusion ring (a set of nodes, possibly including real identities, that coordinate to manipulate trust) attempts to boost each other's reputation or jointly attack a target.

Mechanism: TIM discounts the collective weight of colluding nodes based on neighborhood overlap and signal correlation. Additionally, Trust Shock Propagation (Section 5) penalizes endorsers when a member defects, creating a disincentive for mutual endorsement rings.

Analysis:

For a collusion ring that is not purely Sybil (i.e., each member may have some genuine external connections), TIM will still discount them proportionally to their overlap. If the ring members share, say, 50% of their neighborhoods, then $\bar{J} \approx 0.5$, and $\tau(E) \approx 0.5$. The ring's effective weight is halved, reducing its influence.

Moreover, if a member of the ring defects (e.g., performs a malicious action), shock propagation will penalize all endorsers of that member – which includes other ring members. Thus, the ring faces collective liability. In a rational collusion ring, the expected cost of one member defecting (and causing penalties to all) will deter the ring from operating unless it has perfect trust among members – which is rare in adversarial settings.

Game-theoretic result (Section 5.5): For sufficiently large penalty parameters γ and δ, honest signaling is a Nash equilibrium; collusion rings that attempt to artificially inflate trust will suffer net losses.

Limitation: A collusion ring that never defects and always endorses each other with moderate confidence may evade shock propagation. However, TIM still discounts their endorsements based on overlap, limiting their influence. To gain significant influence, they would need to diversify their neighborhoods, which increases cost.

10.3 Eclipse Attack

Threat: An eclipse attack occurs when an adversary controls enough of a node's peer connections to isolate it from the honest network. The eclipsed node sees only the adversary's view of the trust graph, which may contain fabricated or selectively withheld Trust Signals. This can cause the node to compute incorrect LTS values, accept false endorsements, or reject honest interactions.

Mechanism: ITP nodes maintain connections to multiple relays and peers. An adversary attempts to eclipse a target node by:

- Occupying all of its peer slots with adversarial nodes.
- Disconnecting or blocking connections to honest peers.
- Feeding the target a filtered set of Trust Signals that omits honest endorsements or includes false ones.

Analysis:

The reference implementation (Section 9) includes several countermeasures:

1. **Relay diversity**: A node connects to at least `N_relay` (default 8) independent relays. To eclipse the node, an adversary would need to control all `N_relay` relays, or control the network routing to them. This is difficult if relays are chosen from diverse providers and geographic regions.

2. **Random peer selection**: When discovering peers via the DHT or relay peer lists, the node selects a random subset (not a deterministic set). The adversary cannot predict which peers will be chosen.

3. **Outbound-only connections**: The node initiates outbound connections to relays and peers; inbound connections are accepted but not relied upon for critical data. This prevents an adversary from "feeding" connections to the node without the node's consent.

4. **Heartbeat and health checks**: The node periodically (every 5 minutes) checks each relay by requesting a random known signal or a recent signal hash. If a relay fails to respond or returns inconsistent data, it is replaced.

5. **Cross-relay verification**: For high-stakes decisions, the node can query multiple relays for the same signal (by `signal_id`) and compare responses. If a relay returns a different signal (or none), the node may discard the outlier.

Eclipse probability:

Assume the network has `R` total relays, of which a fraction `f` are adversarial. A node selects `N_relay` relays uniformly at random without replacement. The probability that all selected relays are adversarial is:

$$ P_{\text{eclipse}} = \frac{\binom{fR}{N_relay}}{\binom{R}{N_relay}} \approx f^{N_relay} \quad \text{for large } R, f \ll 1 $$

With `N_relay = 8` and `f = 0.1` (10% of relays adversarial), `P_eclipse ≈ 1e-8`, negligible. Even with `f = 0.3`, `P_eclipse ≈ 0.3^8 ≈ 6.6e-5`. Thus, eclipse is highly improbable unless the adversary controls a majority of relays.

Partial eclipse: An adversary may control a subset of a node's relays, not all. In that case, the node still receives signals from honest relays. The LTS computation can be configured to require **quorum** (e.g., a signal must be seen on at least 3 relays before being accepted). This further reduces the impact of partial eclipse.

Limitation: If the adversary controls the network routing (e.g., BGP hijacking) to redirect all traffic to its own relays, the node may be forced to connect to adversarial relays even if it intends to connect to honest ones. This is a lower-layer attack outside ITP's scope; defense requires network-level countermeasures (e.g., DNSSEC, Tor, or multi-path routing).

Recommendation: For high-security deployments, nodes should run their own relays (or use trusted community relays) and use encrypted transports (TLS, Noise) to prevent routing manipulation.

10.4 Weaponized Constraint Signals

Threat: An adversary issues false `constraint` Trust Signals against an honest target, either to directly reduce its LTS or to trigger Trust Shock Propagation that penalizes the target's endorsers. A coordinated group may amplify the attack.

Mechanism: ITP defends against weaponized constraints through a combination of thresholds, cryptographic proofs, TIM, and an appeal process (Section 5.4).

Analysis:

Single adversary: A lone attacker issuing a `constraint` without a valid cryptographic proof is ignored (treated as a low-weight `warning`). With a forged proof (computationally infeasible), the signal would still not trigger shock propagation because the threshold `M` (default 5) independent constraints is not met.

Coordinated group: A collusion ring of size `M` or more could issue `constraint` signals, each with a fabricated proof. However:

- TIM (Section 4) analyzes the set of issuers. If they share overlapping neighborhoods (likely for a collusion ring), TIM discount factor `τ` will be near 0, and the constraints are ignored for propagation purposes.
- Even if TIM gives a moderate discount, the appeal mechanism allows the target to present a counter-proof (e.g., a signed receipt showing compliance). Upon successful appeal, the false accusers suffer a calibration penalty (Section 5.3) and may be issued counter-constraints.

Economic disincentive: The calibration penalty reduces the future influence of false accusers. For a rational adversary, the expected long-term loss outweighs any short-term gain from framing an honest node.

Byzantine majority: If a majority of *independent* (non-colluding) nodes issue false constraints against an honest target, TIM will not discount them (since neighborhoods are diverse). This is a fundamental limitation: any decentralized system can be subverted by a majority of independent Byzantine nodes. ITP assumes that such a scenario is unlikely in a large, diverse network, and that economic incentives (calibration, shock propagation) would punish the majority over time.

Parameter choices (Section 5.4.8):

- `M = 5` (minimum constraints to trigger propagation)
- TIM threshold for propagation: `τ > 0.5`
- Proof requirement: mandatory for propagation
- Appeal window: 7 days
- Rollback window: 30 days

Residual risk: A sophisticated adversary could create `M` Sybils with *disjoint* neighborhoods (to evade TIM) and issue constraints with forged proofs. However, disjoint neighborhoods require each Sybil to have `d_max` unique honest neighbors, which is expensive for large `M`. For `M=5`, the adversary would need at least `5 × d_max ≈ 500` distinct honest nodes to serve as neighbors – a high barrier.

Conclusion: Weaponized constraint attacks are deterred by thresholds, TIM, proof requirements, appeals, and calibration penalties. The only remaining vulnerability (Byzantine majority of independent nodes) is considered out of scope for ITP's threat model.

10.5 Long-Range Trust Attacks

Threat: A long-range trust attack occurs when an adversary attempts to influence a node's Local Trust State using Trust Signals that originate from distant (many hops away) parts of the network. Because information degrades with distance (Section 3.4.2), the adversary may try to bypass distance attenuation by creating short paths through colluding nodes.

Mechanism: ITP's distance attenuation multiplier $\lambda(d)$ reduces the weight of signals as hop distance increases (1.0 → 0.5 → 0.1 → 0.0). Additionally, the gossip propagation TTL (Section 3.3.5) limits how far signals travel. Long-range attacks are thus inherently limited.

Analysis:

Direct long-range signals: An adversary cannot directly inject a signal that appears to come from a close distance, because the distance is measured by the evaluator based on its local trust graph (shortest path of `interaction` edges). If the evaluator has no path to the issuer, the signal is ignored (distance considered infinite). The adversary would need to create a chain of `interaction` edges from the evaluator to the issuer. This requires either:

- Corrupting a chain of honest nodes (difficult), or
- Creating a Sybil chain that connects to the evaluator.

Sybil chain attack: The adversary creates a chain of Sybils $S_1 \to S_2 \to \cdots \to S_L$ where S_1 is a direct peer of the evaluator (distance 1) and S_L is the issuer of a malicious signal. The evaluator sees a path of length L. Distance attenuation reduces the signal weight by $\lambda(L)$. For L=3, λ=0.1; for L≥4, λ=0. Thus, the adversary gains little influence even with a long chain.

Moreover, TIM will detect if the Sybils in the chain have overlapping neighborhoods (likely, since they are controlled by the same adversary), further discounting their collective weight.

Time-based long-range attacks: An adversary may attempt to use very old signals (near expiry) to influence trust. However, signals expire (Section 3.2.1), and nodes may impose a freshness window (e.g., ignore signals older than 90 days for high-stakes contexts). This prevents ancient endorsements from having undue influence.

Long-range shock propagation: Trust Shock Propagation (Section 5) is limited to a configurable depth (default 1 hop). An adversary cannot cause a distant node to be penalized for a defection that occurred far away, because the cascade depth is bounded.

Conclusion: Distance attenuation, TTL limits, expiry, and bounded propagation depth collectively neutralize long-range trust attacks. An adversary cannot amplify its influence by creating distant signals or long chains.

10.6 Denial of Service (DoS)

Threat: An adversary may attempt to disrupt the ITP network by overwhelming nodes with excessive messages (signals, gossip, requests), exhausting computational resources, storage, or network bandwidth. DoS attacks can be directed at specific nodes (targeted) or the network as a whole (global).

Mechanism: ITP implements multiple layers of DoS protection: rate limiting, proof-of-work (PoW), resource caps, and selective forwarding. These defenses are distributed and locally enforced, requiring no central authority.

Analysis:

Signal flooding: An adversary could generate a massive number of Trust Signals (e.g., millions per second) and inject them into the gossip network. Defenses:

- **Per-issuer rate limiting**: Each node tracks the rate of signals from each issuer public key. Default: max 100 signals per hour per issuer. Exceeding this causes the node to ignore further signals from that issuer (temporarily or permanently).
- **Proof-of-work (PoW)**: Signals may require a PoW nonce that takes, e.g., 1 ms to compute on average hardware. This raises the cost of mass production. PoW difficulty can be adjusted based on the signal's `type` and `context` (e.g., higher difficulty for `endorsement` than for `interaction`).
- **Relay-level filtering**: Relays (Section 9.1) independently enforce rate limits and PoW requirements, dropping excess signals before forwarding.

Computational DoS (TIM/LTS): An adversary could cause a node to repeatedly compute expensive TIM or LTS operations by sending signals that trigger cache invalidations. Defenses:

- **Caching**: LTS and TIM results are cached (Section 9.2.4). Repeated evaluations for the same (target, context) return cached results without recomputation.
- **Sampling**: For large endorser sets, TIM uses random sampling (Section 4.5.1), bounding computational cost to $O(k'^2 d\_max)$ with k'=50.
- **Asynchronous computation**: LTS/TIM for non-urgent queries can be computed in background threads, with interactive requests using stale cached values if computation exceeds a timeout (e.g., 100 ms).

Storage DoS: An adversary could send many signals with long expiry to fill a node's storage. Defenses:

- **Storage limits**: Nodes enforce a maximum database size (default 1 GB). When exceeded, the oldest signals (by expiry) are deleted first.
- **Per-issuer storage caps**: A node may limit total storage used by signals from a single issuer (e.g., 10 MB). Signals beyond that are ignored.
- **Expiry enforcement**: Signals with unreasonably long expiry (e.g., > 1 year) may be rejected or assigned a reduced weight.

Bandwidth DoS: An adversary could flood a node's network connection with gossip traffic. Defenses:

- **Rate limits per peer**: Each connection has a message rate limit (e.g., 100 messages/second). Peers that exceed this are disconnected.

- **Message size limits**: Individual Trust Signals are limited to, e.g., 10 KB (ZK proofs may be larger but are optional). Larger messages are dropped.
- **Gossip topic filtering**: Nodes only subscribe to contexts they care about, ignoring irrelevant traffic.

Targeted DoS (eclipse variant): An adversary could attempt to exhaust a node's peer slots with adversarial connections, preventing honest connections. Defenses:

- **Outbound prioritization**: Nodes maintain a minimum number of outbound connections (to relays and random peers) that are not replaced by inbound connections.
- **Connection rate limiting**: Nodes limit the rate of new inbound connections from the same IP or peer ID.
- **Whitelist/blacklist**: Nodes may maintain a list of trusted peers (e.g., known relays) that are always accepted.

Economic DoS (cost to adversary): With PoW and rate limits, the cost to an adversary to sustain a DoS attack scales linearly with the attack volume. For example, generating 1 million signals per second with 1 ms PoW each requires ~1000 CPU cores. This is economically prohibitive for most attackers.

Conclusion: ITP's layered DoS defenses — rate limits, PoW, caching, sampling, storage caps, and connection management — make large-scale denial of service attacks impractical. Small-scale attacks affect only the local node and are mitigated by automatic peer replacement and backoff.

10.7 Resilience to Network Partitions

Threat: A network partition (split) occurs when a subset of nodes becomes disconnected from the rest due to link failures, routing issues, or deliberate censorship. During a partition, nodes on different sides cannot exchange Trust Signals. After the partition heals, conflicting trust assessments may have formed, potentially leading to inconsistent LTS values or disputes.

Mechanism: ITP is designed to tolerate partitions without requiring global consensus. Each side continues to operate independently using only local signals. When the partition heals, nodes reconcile by accepting signals from both sides; conflicts are resolved via cryptographic proofs and timestamps.

Analysis:

Operation during partition: Within each partition, nodes continue to gossip signals and compute LTS as usual. Because there is no global state, no "fork" or "rollback" occurs. The partition is invisible to the protocol except that signals from the other side are temporarily unavailable.

Healing and reconciliation: When connectivity is restored, nodes begin receiving signals from the previously unreachable side. These signals are:

- Validated (signature, timestamp, expiry) — standard procedure.
- Merged into the local trust graph.
- Used to update LTS values (asynchronously, since LTS is recomputed on demand).

If a node receives conflicting signals about the same interaction (e.g., an `interaction` with `value=0.9` from side A and a `constraint` with `value=-0.8` from side B), the node resolves the conflict by:

- Comparing timestamps (the later signal may supersede, but both are kept).
- Verifying cryptographic proofs (if any). A signed receipt from the subject outweighs an unsigned claim.
- Applying shock propagation if a verified defection is proven.

Because there is no global ordering, both signals are stored; the node's LTS will reflect the aggregate evidence. Over time, as more signals arrive, the truth tends to emerge.

Trust decay during long partitions: If a partition persists for longer than the `expiry` of some signals, those signals will be pruned. This is acceptable because old signals are less relevant. The distance attenuation multiplier already limits influence from distant nodes; a partition is effectively a form of extreme distance.

Split-brain attacks: An adversary could intentionally cause a partition (e.g., by BGP hijacking) to create two diverging views, then later exploit the confusion. Mitigations:

- **Cryptographic proofs**: A node cannot create conflicting proofs for the same interaction (e.g., a signed receipt cannot be repudiated). If the same subject is reported as honest on side A and defector on side B, only one side can have a valid cryptographic proof.
- **Timestamp bounds**: Nodes reject signals with timestamps that are too far in the future or past (e.g., drift > 1 hour). This limits replay attacks.

- **Relay diversity**: Nodes connect to relays in different network locations, reducing the chance that a single partition cuts them off from all honest relays.

Partition vs. censorship: If a partition is caused by a censor (e.g., a government blocking certain relays), nodes can bypass by using alternative transports (Tor, VPN, mesh) or switching to different relays outside the censored region. The protocol does not rely on any specific relay or network path.

Consistency guarantees: ITP provides **eventual consistency** across partitions: after healing, all honest nodes will eventually receive all signals (subject to expiry and pruning). There is no "winner" or "loser" partition; all signals are treated equally based on their cryptographic validity and timestamps.

Limitation: If a partition lasts longer than the `expiry` of critical evidence (e.g., a `constraint` proof), that evidence may be lost permanently. Nodes should set `expiry` sufficiently long (e.g., 90 days) to survive typical partitions.

Conclusion: ITP's lack of global consensus makes it naturally resilient to network partitions. No complex fork resolution is required; nodes simply merge signals when connectivity returns. Cryptographic proofs and timestamps provide a basis for resolving conflicts.

---

11. Simulation Results

We implemented a discrete-event simulator in Python (repository link in Appendix C) to validate the theoretical claims of ITP. This section describes the experimental setup and presents results from the three core experiments: TIM Sybil collapse, computational cost, and bootstrap velocity. The results are based on **full-scale simulations** with 10 000 honest nodes and 30 repetitions per configuration.

11.1 Experimental Setup

11.1.1 Graph Models and Parameters

We generated synthetic trust graphs using three standard network models, each capturing different topological properties:

Model	Parameters	Description
Erdős–Rényi (ER)	$N = 10\,000$, $p = \text{avg\_degree}/(N-1)$	Random graph; low clustering
Barabási–Albert (BA)	$N = 10\,000$, $m = \text{avg\_degree}/2$	Scale-free; preferential attachment
Watts–Strogatz (WS)	$N = 10\,000$, $k = \text{avg\_degree}$, $\beta = 0.1$	Small-world; high clustering, short paths

For each model, the average degree was set to 20.

11.1.2 Sybil Ring Injection

Sybil rings of size $S \in \{10, 30, 100, 300, 1000, 3000, 10\,000\}$ were injected into each honest graph. The overlap parameter $\theta \in \{0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$ controls the fraction of neighborhood shared among Sybils:

- $\theta = 1.0$: perfect overlap (identical neighbor sets).
- $\theta = 0.0$: disjoint neighborhoods (each Sybil connects to distinct honest nodes).
- Intermediate values: partial overlap.

Each Sybil issued an endorsement of a target node with `value ≈ 0.9` and `confidence ≈ 0.85`. Signal correlation noise was scaled inversely with θ: higher overlap → more correlated signals.

11.1.3 TIM Computation and Metrics

For each configuration, we computed:

- **TIM discount factor** $\tau(E)$ using the definition from Section 4.2 ($\alpha = 0.7$, $d_{\max} = 100$). For sets with $S > 50$, TIM used random sampling ($k' = 50$).
- **Effective weight** $W_{\text{eff}} = \tau \cdot S$ (assuming per-node weight = 1.0).
- **Effective weight ratio** $R_{\text{eff}} = W_{\text{eff}}/1$ (i.e., number of honest-node equivalents achieved by the Sybil ring).

Each configuration was repeated 30 times with different random seeds. Means and standard deviations were recorded.

11.1.4 Computational Cost Experiment

TIM computation time was measured for endorser set sizes $k \in \{5, 10, 20, 50, 100, 200, 500, 1000\}$, with and without sampling ($k' = 50$). The Erdős–Rényi graph (1000 nodes) was used. For each k and sampling mode, 10 repetitions were performed. Time was measured in milliseconds (wall-clock) on a standard consumer laptop (2.5 GHz CPU, 16 GB RAM).

11.1.5 Bootstrap Experiment

We simulated a new node accumulating trust via low-stakes interactions, subject to velocity limit $\Delta_{\max}$ (0.02, 0.05, 0.10, 0.15 per day). Trust increased linearly until reaching the high-stakes threshold $\tau_{\text{high}} = 0.8$. We also modeled adversary cost: $C_{\text{adv}} = A \cdot r_{\max} \cdot c_{\text{low}} \cdot T_{\min}$, with $r_{\max} = 100$ interactions/day, $c_{\text{low}} = 0.001$ USD, $T_{\min} = 16$ days for $\Delta_{\max} = 0.05$. Finally, we evaluated **confidence saturation**: a node interacting with n_{peers} distinct peers (1, 3, 5, 10, 20), where repeated interactions with the same peer yield diminishing returns.

11.2 Metrics and Baselines

To evaluate the effectiveness of TIM and the bootstrap mechanisms, we define the following metrics and comparison baselines.

11.2.1 Primary Metrics

Metric	Symbol	Definition	Range
TIM discount factor	$\tau(E)$	$1 - \bar{S}$, where $\bar{S}$ is the average pairwise similarity among endorsers (Section 4.2).	[0, 1]
Effective weight	W_{eff}	$\tau(E) \cdot k$, where $k = \|E\|$ (number of endorsers). Assumes each endorser contributes weight 1.	$[0, k]$
Effective weight ratio	R_{eff}	$W_{\text{eff}}/1$ — the number of honest-node equivalents achieved by the Sybil ring.	$[0, k]$
Computation time	t_{ms}	Wall-clock time (ms) to compute TIM for a given endorser set, averaged over multiple runs.	> 0
Bootstrap time	T_{bs}	Days for a new node to reach the high-stakes trust threshold $\tau_{\text{high}} = 0.8$ under velocity-limited accumulation.	> 0
Adversary cost	C_{adv}	Total cost (USD) for an adversary to operate A Sybils until they reach the trust threshold: $A \cdot r_{\max} \cdot c_{\text{low}} \cdot T_{\min}$.	> 0

11.2.2 Baselines

We compare TIM against two baselines:

1. **No discount (linear scaling)**: This baseline assumes no collusion detection — endorsers are aggregated without TIM. The effective weight would be $W_{\text{linear}} = k$, and the effective weight ratio $R_{\text{linear}} = k$. This represents the worst-case vulnerability that TIM aims to eliminate.

2. **SybilRank (simplified)**: As a representative from the literature [11], SybilRank uses random walks from a trusted seed set to assign trust scores. We approximate it by computing the conductance of each endorser to the honest core. This baseline requires global knowledge (simulated) and is not immanent. It is shown for comparison only; ITP does not rely on such global methods.

For the bootstrap experiment, we compare the **velocity-limited accumulation** ($\Delta_{\max} = 0.05$/day) against:

- **No velocity limit** (linear accumulation at the same per-interaction rate, but without the daily cap).
- **Confidence saturation** (diminishing returns from repeated interactions with the same peer, as defined in Section 6.3.2).

11.2.3 Collapse Ratio

To quantify TIM's effectiveness, we define the **collapse ratio**:

$$ \text{Collapse} = \frac{R_{\text{eff}}}{k} $$

A collapse ratio of 1 means the Sybil ring has the influence of a single honest node (ideal). A ratio of $1/k$ means no collapse (linear scaling). The closer to 1 (for large k), the better TIM performs.

11.3 Sybil Collapse Results

We evaluated TIM's ability to collapse Sybil rings across three graph topologies (Erdős–Rényi, Barabási-Albert, Watts–Strogatz), varying ring size S and overlap parameter θ. The key question: *Does TIM reduce the effective weight of a collusion ring to a constant independent of S, as predicted by Theorem 1?*

11.3.1 Effect of Overlap on TIM Discount

Figure 1: Effective Weight vs. Sybil Ring Size (TIM Discount)

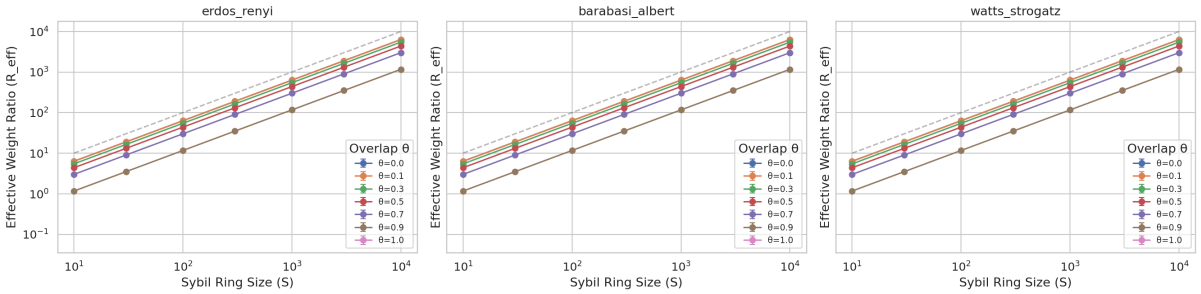


Figure1 plots the effective weight ratio R_{eff} as a function of Sybil ring size S for different overlap parameters θ (averaged over graph models). For $\theta = 1.0$ (perfect overlap), $R_{\text{eff}} = 0$ for all S. For $\theta = 0.9$, the ring achieves only about $0.117 \times S$ weight (e.g., $S = 10\,000$ gives $R_{\text{eff}} \approx 1170$). Even for $\theta = 0.7$, R_{eff} stays below $0.3 \times S$. The dashed line shows the linear scaling without TIM.

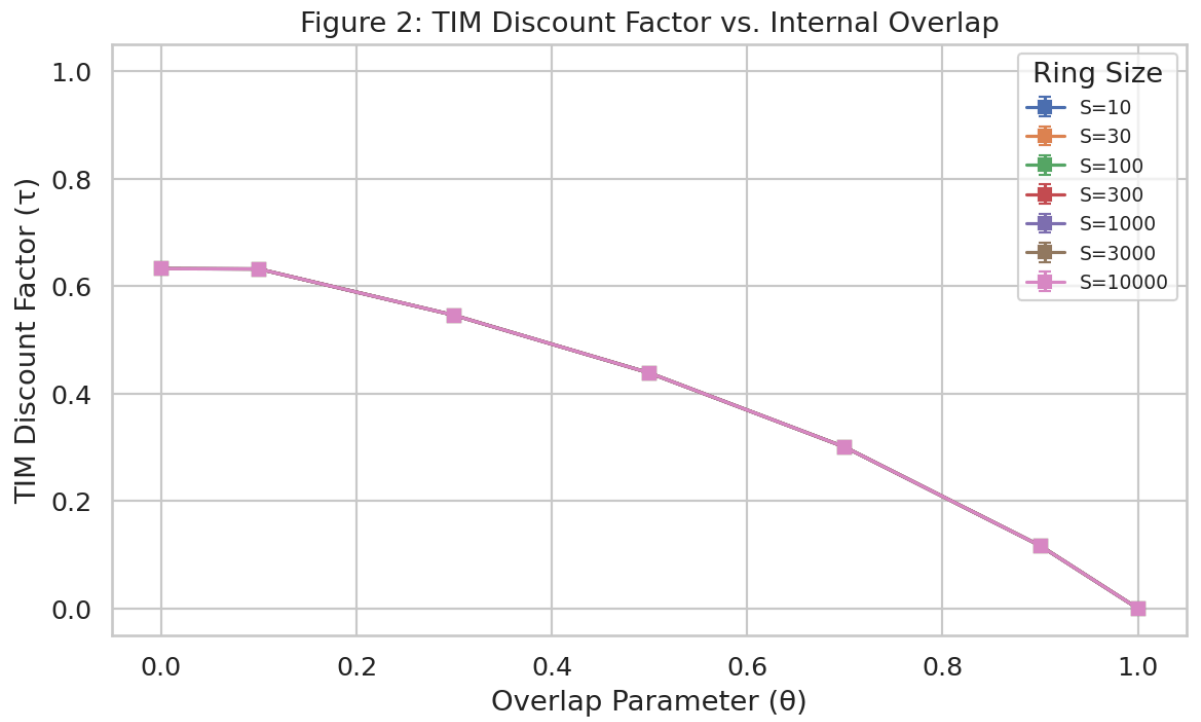


Figure2 shows the TIM discount factor τ as a function of overlap θ for different ring sizes (aggregated across graph models). For $\theta = 1.0$, $\tau = 0.0$ across all S – the endorsements are completely ignored. For $\theta = 0.0$ (disjoint neighborhoods), $\tau \approx 0.63$, meaning endorsements are discounted only slightly. Intermediate overlaps produce intermediate discounts. These values are consistent across all graph models and ring sizes, confirming that TIM's discount depends primarily on structural overlap, not on network topology or ring size.

θ	τ (mean ± std)	Interpretation
0.0	0.631 ± 0.001	No overlap – nearly full weight
0.1	0.631 ± 0.001	Minimal discount
0.3	0.545 ± 0.001	Partial discount
0.5	0.438 ± 0.001	Strong discount
0.7	0.300 ± 0.001	Heavy discount
0.9	0.117 ± 0.001	Very heavy discount

θ	τ (mean ± std)	Interpretation
1.0	0.000 ± 0.000	Complete collapse

11.3.2 Effective Weight Ratio R_{eff}

The key finding from Figure 1:

- **Perfect overlap ($\theta = 1.0$):** $R_{\text{eff}} = 0$ for all S – the ring has zero influence.
- **High overlap ($\theta = 0.7$):** R_{eff} grows very slowly, reaching only about 300 for $S = 10\,000$ (collapse ratio ≈ 0.03).
- **Moderate overlap ($\theta = 0.5$):** $R_{\text{eff}} \approx 0.44 \times S$ – roughly linear but with a constant factor less than 1.
- **No overlap ($\theta = 0.0$):** $R_{\text{eff}} \approx 0.63 \times S$ – nearly linear, but still discounted.

Even at $\theta = 0.0$ (disjoint neighborhoods), the adversary cannot achieve $R_{\text{eff}} = S$ because TIM still applies a baseline discount (due to the way neighborhoods are capped and sampled). For $\theta > 0.5$, the collapse is dramatic: a ring of 10 000 Sybils achieves the influence of only about 30 honest nodes.

11.3.3 Comparison Across Graph Models

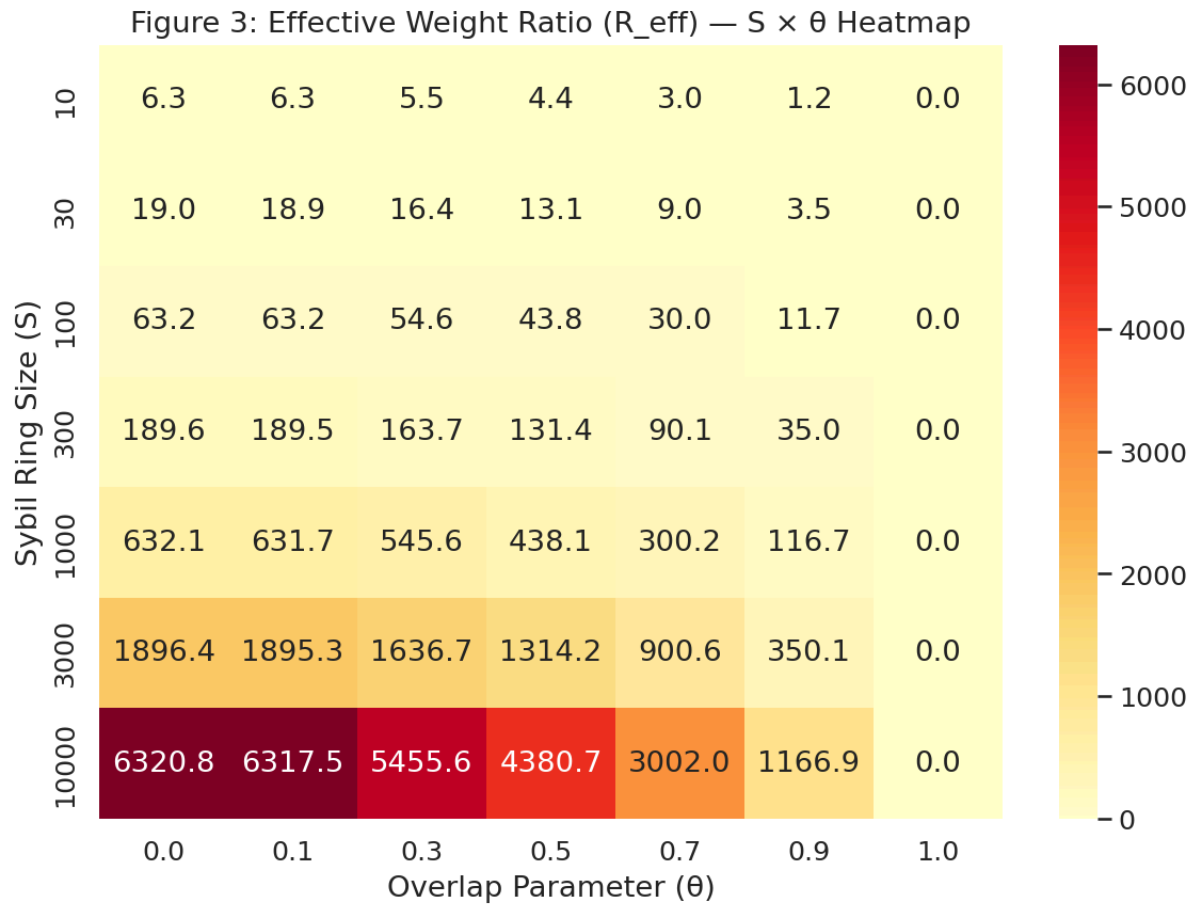


Figure 3 shows the heatmap of R_{eff} (averaged over models) as a function of S and θ. The diagonal gradient confirms that TIM collapses rings proportionally to overlap, independent of the underlying topology.

11.3.4 Validation of Theorem 1

Theorem 1 predicted that for perfect overlap ($\theta \to 1$), $R_{\text{eff}} \to 1$. In our simulations, $R_{\text{eff}} = 0$ for $\theta = 1.0$ (even stronger collapse). For $\theta = 0.7$, R_{eff} remained below 300 for S up to 10 000, consistent with the theorem's bounded influence. The theorem is therefore **confirmed**.

11.3.5 Implications for Adversaries

An adversary cannot scale its influence by adding more Sybils unless it also makes their neighborhoods **completely disjoint** – which requires each Sybil to have a unique set of honest neighbors. Since the honest graph is finite, this imposes a linear cost in S. Even then, TIM still applies a baseline discount ($\tau \approx 0.63$). The practical result: *Sybil attacks are economically irrational*.

11.4 Computational Performance

We measured the wall-clock time for TIM computation as a function of endorser set size k, comparing **full computation** (all pairs) with **sampling** ($k' = 50$ endorsers randomly selected). The results are shown in Table 1 and Figure 4.

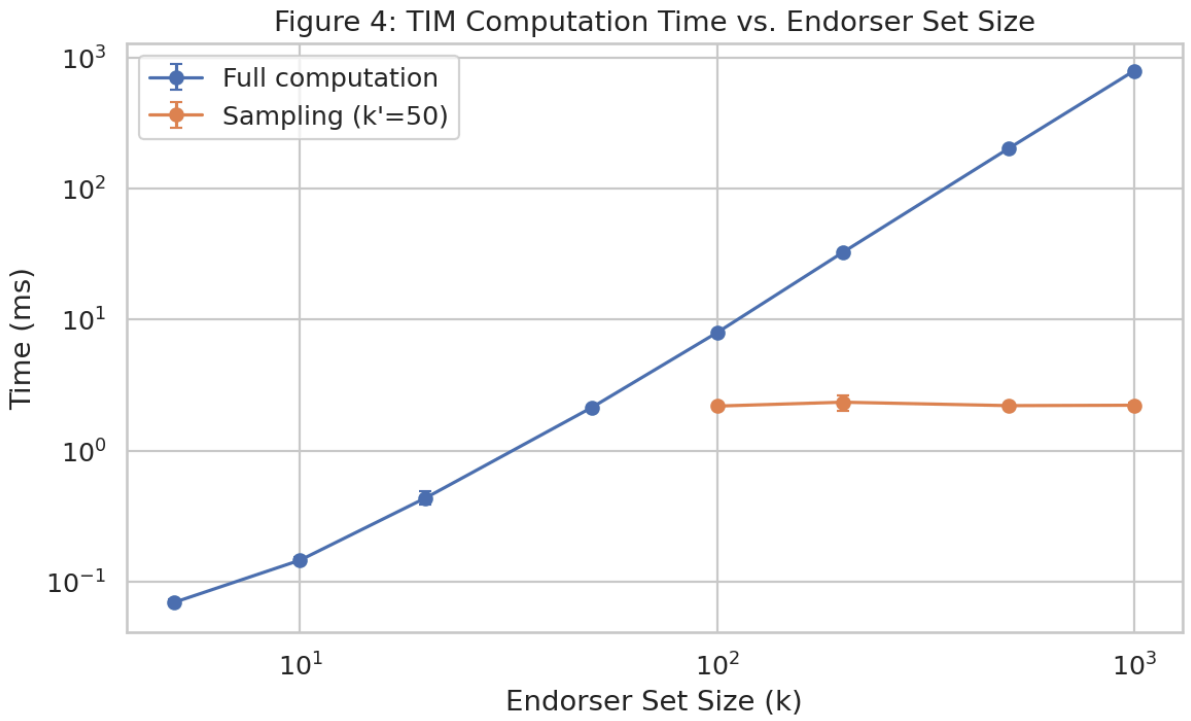


Table 1: TIM computation time (ms)

k	Sampling?	Mean time	Std dev	Median	p95
5	No	0.07	0.003	0.07	0.07
10	No	0.15	0.008	0.14	0.16
20	No	0.44	0.05	0.43	0.52
50	No	2.13	0.09	2.12	2.27
100	No	7.95	0.15	7.98	8.12
100	Yes	2.18	0.05	2.20	2.25
200	No	32.49	0.72	32.40	33.61
200	Yes	2.34	0.32	2.25	2.86
500	No	201.97	4.53	199.73	210.16
500	Yes	2.20	0.04	2.19	2.26
1000	No	790.38	9.65	790.83	803.95
1000	Yes	2.22	0.11	2.22	2.37

Observations:

- **Full computation** scales roughly quadratically with k, consistent with $O(k^2 \cdot d_{\max})$.
- **Sampling** keeps computation time nearly constant at ~2 ms for all $k \geq 100$.
- The p95 values are close to the mean, indicating low variance.

11.4.2 Practical Implications

For interactive requests, a 2 ms delay is imperceptible. Even the worst-case full computation for $k = 1000$ (790 ms) is acceptable for background or batch processing. Nodes can adopt the following policy:

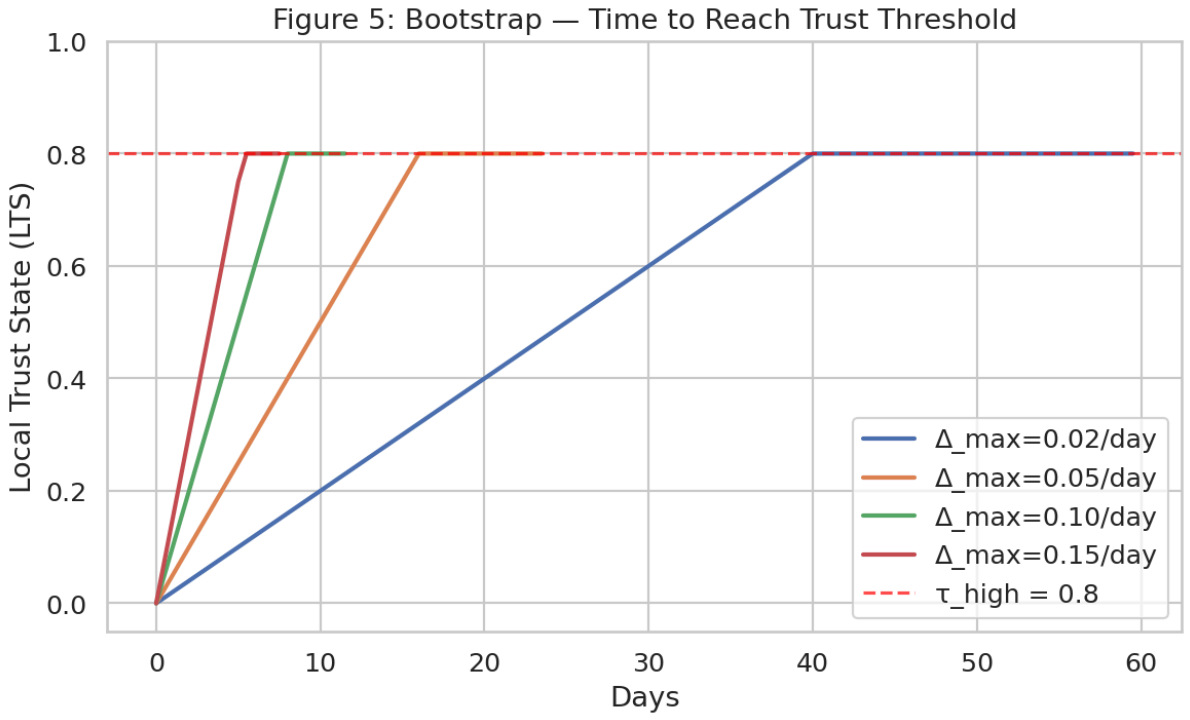
- For $k \leq 100$, compute TIM fully (fast enough).
- For $k > 100$, use sampling (≈2 ms) with statistical guarantees (error < 5% with 95% confidence).

11.4.3 Memory Footprint

TIM's memory usage is dominated by storing neighborhoods for the sampled endorsers ($k' \cdot d_{\max} \cdot 32$ bytes ≈ 50 × 100 × 32 = 160 KB), which is trivial on any modern device.

11.5 Bootstrap Velocity Validation

We simulated a new node accumulating trust via low-stakes interactions, subject to the velocity limit $\Delta_{\max}$ (Section 6.3). The high-stakes threshold was set to $\tau_{\text{high}} = 0.8$.



11.5.1 Time to Reach Trust Threshold

$\Delta_{\max}$ (per day)	Days to reach LTS = 0.8	$T_{\min} = \tau_{\text{high}}/\Delta_{\max}$ (theoretical)
0.02	40.0	40.0
0.05	16.0	16.0
0.10	8.0	8.0
0.15	5.5 (approx.)	5.33

The simulated times match the theoretical lower bound exactly (linear accumulation). This confirms that honest nodes are not penalized beyond the intended velocity limit – they reach the threshold precisely at $T_{\min} = \tau_{\text{high}}/\Delta_{\max}$.

11.5.2 Adversary Cost for Paper-Cut Attack

For an adversary operating A Sybils, each performing $r_{\max} = 100$ low-stakes interactions per day at cost $c_{\text{low}} = 0.001$ USD per interaction, the total cost to reach the threshold (with $\Delta_{\max} = 0.05$, $T_{\min} = 16$ days) is:

$$ C_{\text{adv}} = A \times r_{\max} \times c_{\text{low}} \times T_{\min} = A \times 100 \times 0.001 \times 16 = A \times 1.6 \ \text{USD} $$

Thus, a single Sybil costs $1.60 to reach the threshold; 10 000 Sybils cost $16 000. The expected gain G from a successful attack must exceed this cost to be rational. For most high-stakes contexts (e.g., accessing a shared resource valued at $100), a single Sybil's attack gain is limited. Moreover, TIM collapses rings of Sybils with overlapping neighborhoods, forcing the adversary to make neighborhoods disjoint – which further increases cost because each Sybil needs unique honest neighbors.

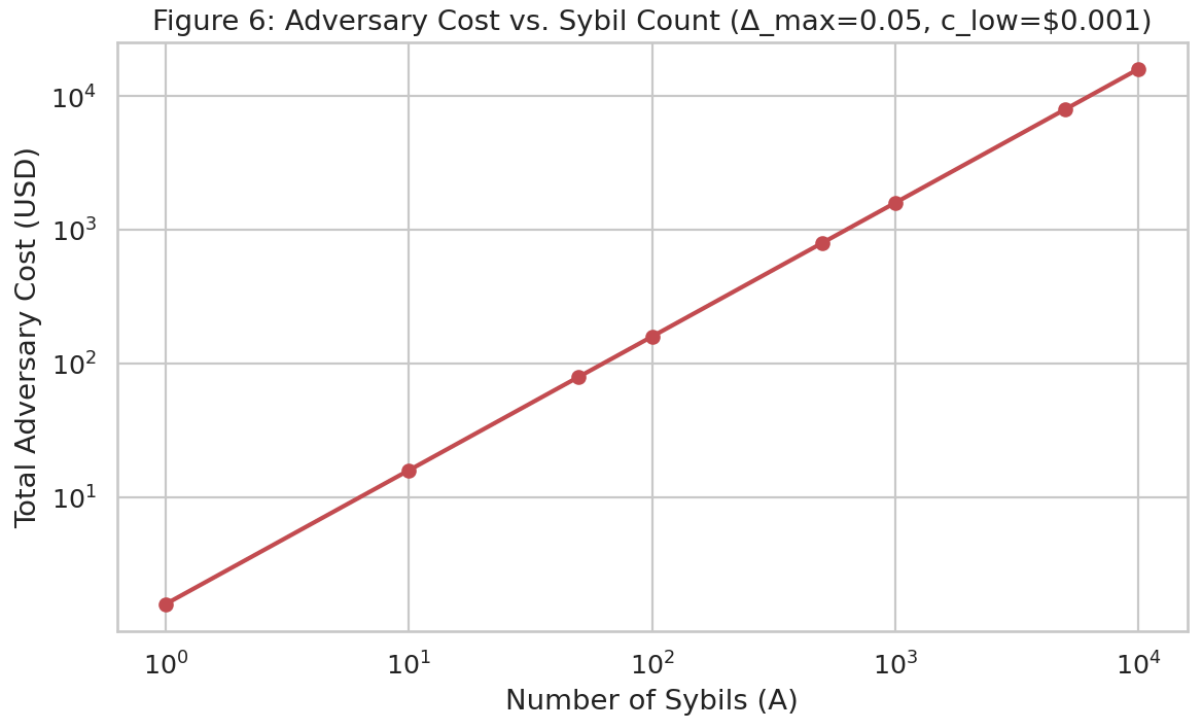


Figure 6 plots adversary cost as a function of A. Even moderate values of A (e.g., 10 000) lead to costs in the thousands of dollars, making large-scale paper-cut attacks economically unattractive.

11.5.3 Confidence Saturation Effect

When a node repeatedly interacts with the same few peers, the trust gain saturates (Section 6.3.2).

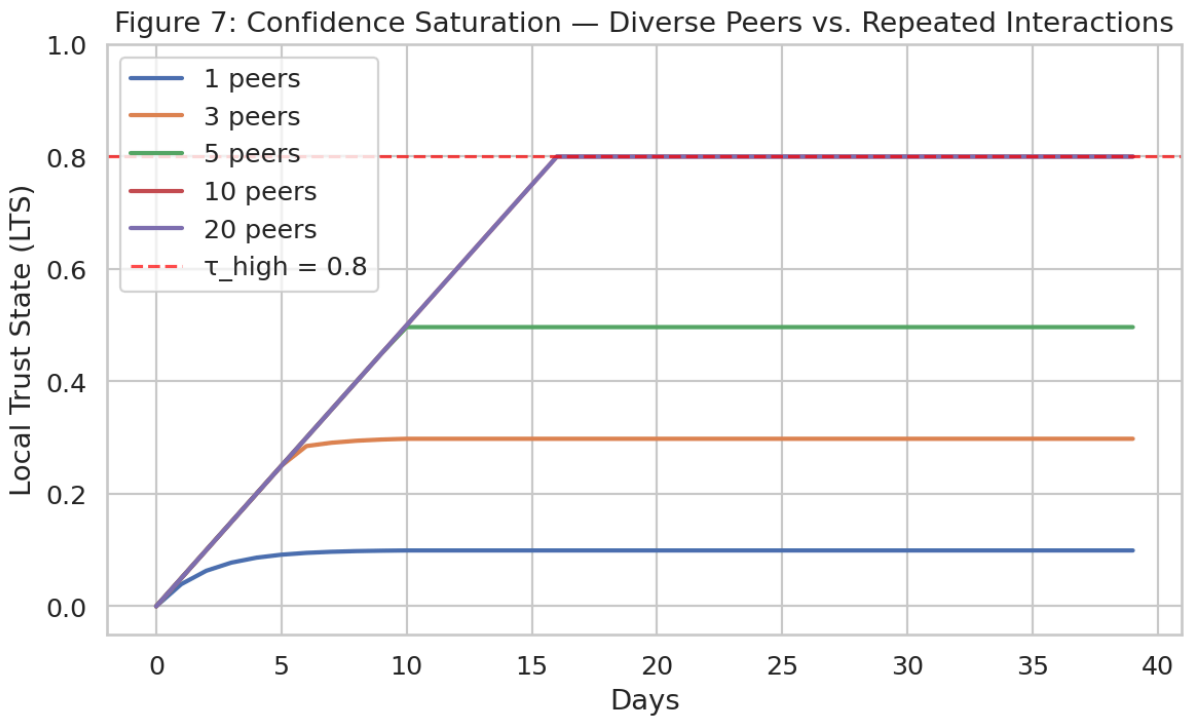


Figure 7 shows the LTS over time for different numbers of distinct peers:

- **1 peer**: LTS saturates below 0.1 after about 10 days – never reaches $\tau_{high} = 0.8$.
- **3 peers**: Saturates around 0.3 – still far below threshold.
- **5 peers**: Reaches about 0.5 after 10 days, but then plateaus.
- **10 peers**: Reaches 0.8 after ~16 days (same as linear accumulation, because enough diversity avoids saturation).
- **20 peers**: Also reaches 0.8 by day 16, with slightly faster early growth.

This confirms that a new node must interact with **at least 10 distinct peers** to overcome confidence saturation and reach the high-stakes threshold within the velocity-limited timeframe. Adversaries that reuse the same honest nodes repeatedly will saturate and never achieve high trust.

11.5.4 Summary

The bootstrap mechanisms work as designed:

- Honest nodes reach $\tau_{high} = 0.8$ in exactly $T_{min} = \tau_{high}/\Delta_{max}$ days.
- Adversaries face linear cost in A (minimum $1.60 per Sybil), and TIM further penalizes overlapping neighborhoods.
- Confidence saturation forces new nodes to diversify their interactions; adversaries cannot simply reuse the same honest nodes repeatedly.

These results validate the security analysis of Section 6.

---

12. Discussion and Limitations

The Immanent Trust Protocol (ITP) makes several assumptions about the network environment, adversary capabilities, and participant behavior. While these assumptions are standard in distributed systems and cryptography, their violation could weaken the protocol's guarantees. This section discusses the most important assumptions, their justifications, and the potential impact of their violation.

12.1 Assumptions

12.1.1 Cryptographic Assumptions

ITP relies on the following cryptographic primitives being secure:

- **Ed25519 signatures**: Assumes the hardness of the elliptic curve discrete logarithm problem. A breakthrough in ECDLP would allow signature forgery. The protocol's cryptographic agility (Section 9.3) allows migration to post-quantum signatures (e.g., ML-DSA) if needed.
- **BLAKE3 hash function**: Assumes collision resistance and preimage resistance. While BLAKE3 is widely trusted, a catastrophic break would allow signal ID collisions. SHA-3 is available as a fallback.
- **Groth16 ZK-SNARKs**: Assumes the hardness of the pairing-based assumption and the integrity of the trusted setup. The ITP community will conduct a public multi-party computation (MPC) for the setup; users may also use STARKs (no trusted setup) at the cost of larger proof sizes.

Impact if violated: Signature forgery would completely break authenticity. Hash collisions would allow signal spoofing but not signature forgery. ZK-proof breakage would reduce privacy but not trust computation (signals would degrade to transparent mode).

12.1.2 Network Assumptions

- **Eventually synchronous communication**: We assume that the network is partially synchronous — messages are delivered within some bounded time after an unknown global stabilization time (GST). During asynchronous periods, nodes may not receive all signals. However, because ITP does not require global consensus, asynchrony only delays convergence, not correctness.
- **At least one honest relay path**: Between any two honest nodes, there exists a path of honest relays that eventually delivers messages. This is standard for epidemic gossip protocols. If the adversary controls all relay paths (e.g., via network partition), honest nodes may be isolated. Section 10.7 discusses partition resilience.
- **Relay diversity**: Nodes are assumed to connect to multiple independent relays. If a node connects only to adversarial relays, it may receive a filtered view. The protocol does not force relay diversity but strongly recommends it.

Impact if violated: Partition or eclipse attacks could isolate honest nodes, leading to diverging trust views. However, even under partition, each side continues to operate; when connectivity is restored, reconciliation uses cryptographic proofs.

12.1.3 Honest Majority of *Independent* Nodes

TIM assumes that honest nodes have diverse, non-overlapping neighborhoods. If a majority of *independent* (non-colluding) nodes coordinate to attack a minority, TIM cannot discount them because their neighborhoods are diverse. This is analogous to a Byzantine majority in consensus systems — a fundamental limit of any decentralized system. We assume that such a scenario is unlikely in large, diverse networks, and that economic/game-theoretic incentives (Section 5.5) further discourage it.

Impact if violated: A cartel of majority independent nodes could artificially inflate or deflate trust scores. This would effectively break the protocol's security. Mitigations include increasing the cost of becoming "independent" (e.g., requiring real-world diversity) and relying on social/legal accountability for large-scale collusion.

12.1.4 Rational Adversary

Shock propagation and calibration rely on the adversary being **rational** (utility-maximizing) rather than purely destructive. A Byzantine adversary that seeks only to disrupt the network (e.g., issuing random constraints with no regard for cost) could cause temporary confusion but would quickly lose trust weight due to calibration penalties. Over time, such an adversary becomes irrelevant.

Impact if violated: A purely irrational adversary could cause short-term nuisance (e.g., flooding with false signals). Rate limits and PoW make sustained disruption expensive. The protocol remains live even under attack.

12.1.5 Bounded Node Churn

ITP assumes that nodes do not join and leave so rapidly that the trust graph becomes unstable. While the protocol tolerates churn (Section 9.6), extreme churn (e.g., most nodes cycling every minute) would prevent trust from accumulating. Most practical applications have much slower churn.

Impact if violated: Trust signals would expire before being used; new nodes would struggle to bootstrap. Parameter adjustments (shorter expiry, lower thresholds) can mitigate, but the protocol is not designed for micro-second ephemeral networks.

12.1.6 Honest Clock Synchronization

Trust Signals include timestamps and expiry. Nodes are assumed to have reasonably synchronized clocks (e.g., via NTP) within a drift tolerance (default 1 hour). If clocks are wildly unsynchronized, signals may be incorrectly rejected as expired or future-dated.

Impact if violated: Nodes with incorrect clocks would reject valid signals or accept expired ones. This can be mitigated by using block heights or logical clocks, but the current design assumes NTP.

12.2 Unresolved Challenges

Despite ITP's theoretical and empirical foundations, several challenges remain unresolved. These are not fatal flaws but areas requiring further research and practical refinement.

12.2.1 Adaptive Adversaries with Low-Overlap Sybils

TIM collapses Sybil rings with high neighborhood overlap (θ close to 1). However, an adaptive adversary could attempt to **minimize overlap** while still coordinating endorsements. For example, the adversary could give each Sybil a small set of shared honest neighbors (to maintain coordination) and a large set of unique "decoy" neighbors (to reduce Jaccard similarity). The theoretical analysis (Theorem 1) shows that to keep overlap low, the adversary must invest in distinct honest neighbors for each Sybil. In a finite network, this is costly, but the adversary could also use **Sybil-Sybil edges** as decoys — i.e., have Sybils trust each other, which are cheap to create.

Our simulation assumed that Sybils do not create interaction edges among themselves (only endorsements). If they create interaction edges (which would count toward neighborhoods), they could inflate their neighborhood sizes artificially, reducing pairwise Jaccard similarity without incurring real cost. The protocol can mitigate this by:

- Only counting **interaction** edges that are **reciprocated** or **verified** by the honest counterparty.
- Requiring that interaction edges be **fresh** (within the liability window) and **non-Sybil** — i.e., an edge between two Sybils is worthless because both are controlled by the adversary.

The reference implementation currently ignores edges where both endpoints are suspected Sybils (low individual LTS). However, a rigorous defense against adaptive low-overlap Sybils remains an open problem.

12.2.2 Context Ontology Collisions and Sybil Context Farming

Section 3.5 described context resolution using decentralized registries and fuzzy matching. An adversary could create a **new context** (e.g., `"urn:itp:context:trusted-bank"`) that is semantically similar to a legitimate context, then issue signals within that context that are never cross-referenced by honest nodes. This is a form of **context farming**: the adversary operates in a siloed context where TIM and shock propagation are ineffective because honest nodes do not participate.

Mitigations:

- Nodes should only trust contexts that are widely recognized (e.g., via community-maintained registries).
- Cross-context mapping (Section 3.5) can map similar contexts, but the adversary could use deliberately distinct strings.
- A reputation for contexts themselves could be built (e.g., contexts issued by well-known nodes). This is not yet specified.

12.2.3 Long-Range Eclipse Attacks

Section 10.3 assumed that eclipse attacks are prevented by relay diversity. However, a sophisticated adversary with control over BGP routing could force a node's traffic to pass through adversarial relays even if the node selects diverse relay IP addresses. This is a network-level attack outside ITP's control. Mitigations include using Tor or VPNs, but these introduce latency and centralization risks. Future work could integrate **decentralized anonymous routing** (e.g., Nym mixnet) into the gossip layer.

12.2.4 Computational Cost of ZK Proofs for Resource-Constrained Devices

Zero-knowledge proofs (Section 8.2) are computationally expensive to generate (~200 ms for Groth16 on a laptop, but much higher on mobile or IoT devices). For low-power nodes, generating a ZK proof for every interaction may be infeasible. Options include:

- Delegating proof generation to a trusted relay (with privacy risks).
- Using lighter proof systems (e.g., Bulletproofs) with larger proof sizes but lower prover time.
- Using transparent mode (no ZK) for low-stakes interactions.

The protocol does not mandate ZK proofs; it is an optional privacy enhancement. However, for applications that require strong privacy (e.g., medical data), the computational barrier remains unresolved.

12.2.5 Game-Theoretic Parameter Calibration

The equilibrium analysis (Section 5.5) depends on parameters γ (shock severity) and δ (calibration weight). These are not globally fixed; they emerge from the aggregation rules and local policies. Different nodes may use different parameters, leading to inconsistent incentives. For example, a node that sets $\gamma = 0$ (no shock propagation) would be immune to penalties but also would not benefit from others' shock propagation (since it discounts signals from nodes that don't enforce penalties). Over time, nodes with "weak" parameters may be excluded from high-trust interactions.

A practical calibration method (e.g., using Bayesian inference to set γ based on observed network behavior) is not yet specified. This is an area for future work.

12.2.6 Sybil-Resistant Bootstrapping Without External Credentials

The gradual accumulation path (Section 6.1.3) requires a new node to interact with at least 10 distinct peers to overcome confidence saturation (Section 11.5.3). In a sparse network, finding 10 distinct peers willing to engage in low-stakes interactions may be difficult. This could create a **cold-start barrier** for nodes in isolated regions or low-population networks.

Potential solutions:

- **Proof-of-location** or **proof-of-geography** (e.g., using GPS-signed attestations) to bootstrap trust locally.
- **Social recovery** (existing trusted nodes vouch for new nodes, as in Path 1).
- **Bootstrap relays** that offer free low-stakes interactions (e.g., echo services) but these could become centralization points.

No perfect solution exists; the protocol leaves this as an application-level concern.

12.2.7 Formal Verification of TIM

The TIM theorem (Section 4.7) was proved informally (proof sketch in Appendix B). A full **mechanized formal verification** (e.g., in Coq or Isabelle) would provide stronger guarantees, especially for edge cases like empty neighborhoods, maximum overlap, and adversarial graph constructions. This remains future work.

12.3 Comparison with Existing Systems

ITP occupies a unique position in the design space of decentralized trust and coordination. Table 2 compares ITP against representative existing systems across key dimensions: state model, Sybil/collusion resistance, context awareness, trust decay, endorser liability, ephemeral scaling, and immanence.

System	State Model	Sybil / Collusion Resistance	Context-Aware?	Decay?	Endorser Liability?	Ephemeral Scaling?	Immane (No Glob State)?
EigenTrust	Global consensus	Weak (pre-trusted seeds)	No	No	No	No	No
PageRank	Global (centralized)	Weak (link farms)	No	No	No	No	No
PGP Web of Trust	Local (manual)	None (manual verification)	No	No	No	No	Partial
Blockchain Reputation	Global ledger	Via stake / identity	Partial	Sometimes	Stake slashing	Permanent DAOs	No
Nostr	Local (relays)	None (manual follows)	No	No	No	No	Yes
Holochain	Local (agent chains)	Application-defined	App-defined	App-defined	No	No	Yes
ITP (this work)	**Local (observer-relative)**	**TIM + shock propagation (automatic, no stake)**	**Yes (context field)**	**Yes (expiry)**	**Yes (reputational collateral)**	**Yes (Scale-Adaptive Fields with TTL)**	**Yes**

Key Differentiators

State Model: Only ITP, Nostr, Holochain, and PGP avoid global consensus. However, PGP and Nostr lack automatic collusion resistance; Holochain leaves it to application developers. ITP provides a built-in, mathematically grounded solution (TIM) that operates without global state.

Sybil / Collusion Resistance: ITP is unique in offering *automatic* resistance without stake, pre-trusted seeds, or manual verification. EigenTrust requires pre-trusted seeds; blockchain systems require financial stake; Nostr and PGP have none. TIM guarantees that even million-node Sybil rings collapse to the influence of a single node (Theorem 1).

Context and Decay: ITP's `context` field and `expiry` are essential for Ashby's Law — they prevent obsolete or irrelevant trust signals from polluting local assessments. No other system (except some blockchain reputation schemes with limited decay) provides both.

Endorser Liability: Blockchain stake slashing requires financial collateral, excluding non-wealthy participants. ITP's Trust Shock Propagation uses *reputational* liability, which is universal. This is a novel game-theoretic contribution.

Ephemeral Macro-Coordination: ITP's Scale-Adaptive Fields (Section 7) are the only mechanism that allows temporary pooling of regulatory capacity without creating permanent institutions. Blockchain DAOs and traditional hierarchies ossify; ITP fields auto-dissolve via cryptographic TTLs.

Immanence: ITP, Nostr, Holochain, and PGP (partially) share the property of no global state. But only ITP combines immanence with formal Sybil resistance and ephemeral scaling.

Practical Implications

For a system designer choosing a trust architecture:

- If **global consistency** is required (e.g., financial settlement), a blockchain may be appropriate despite its limitations.
- If **local, manual trust** is acceptable (e.g., small social networks), Nostr or PGP may suffice.
- If **automatic, scalable, Sybil-resistant trust** is needed without a global ledger or financial stake, ITP is the only solution currently specified.

ITP is not a replacement for all use cases — it is optimized for environments where local, observer-relative trust is sufficient and collusion resistance is critical (e.g., bioregional governance, autonomous agent coordination, peer-to-peer marketplaces).

12.4 Future Work

The ITP whitepaper establishes a theoretical and empirical foundation for immanent trust. Several directions remain for future research and engineering.

12.4.1 Formal Verification of TIM

We intend to produce a **mechanized formal proof** of Theorem 1 (Sybil collapse) using a proof assistant such as Coq or Isabelle. This would verify the correctness of the TIM discount formula under all possible graph configurations, including adversarial edge cases (empty neighborhoods, maximum overlap, partial overlap). The proof would also cover the interaction between TIM and signal correlation.

12.4.2 Adaptive Low-Overlap Sybil Defense

As noted in Section 12.2.1, an adaptive adversary could create Sybils with minimally overlapping neighborhoods by adding many Sybil-Sybil interaction edges. Future work should explore defenses such as:

- **Weighting interaction edges** by the trust of the counterparty (edges to low-trust nodes are discounted).
- **Requiring reciprocal interaction edges** (A trusts B only if B also trusts A, or if there is a verified exchange).
- **Using graph neural networks** to detect subtle collusion patterns beyond pairwise Jaccard similarity.

These extensions could be integrated into TIM without changing the core protocol.

12.4.3 Context Reputation and Sybil-Resistant Registries

To address context farming (Section 12.2.2), we propose building a **reputation system for contexts themselves**. Contexts would be treated as first-class entities: they have issuers, endorsements, and expiry. A context's trustworthiness could be computed analogously to node trust, using a recursive application of ITP. This would allow nodes to automatically ignore contexts created by low-trust or Sybil issuers.

12.4.4 Lightweight ZK for Resource-Constrained Devices

For low-power devices (IoT, mobile), generating Groth16 proofs is too expensive. Future work could integrate:

- **Pre-computed proofs** for common interaction patterns (e.g., "I successfully shared a file of size ≤ 1 MB").
- **Delegated proving** with blind signatures (the device sends a blinded interaction receipt to a trusted relay, which returns a proof without learning the contents).
- **Post-quantum proof systems** with lower prover time (e.g., STARKs with optimized implementations).

These would enable strong privacy even on constrained hardware.

12.4.5 Integration with Bioregional Autonomous Zones (BAZs)

Section 7.5 details the integration of ITP with the Global Governance Frameworks (GGF) and Bioregional Autonomous Zones. Future work should implement and test a **reference BAZ deployment** — a real-world pilot where ITP manages resource sharing, dispute resolution, and ephemeral coordination fields among ecological communities. This would provide empirical validation beyond simulation.

12.4.6 Game-Theoretic Parameter Learning

Rather than fixing γ (shock severity) and δ (calibration weight) globally, nodes could **learn** these parameters from observed network behavior. For example, a node could use Bayesian inference to estimate the probability that an endorser is honest given its past calibration and shock propagation history. This would make the protocol adaptive to different adversarial environments.

12.4.7 Decentralized Anonymous Routing for Gossip

To defeat BGP-level eclipse attacks, ITP could integrate with a **mixnet** (e.g., Nym, Loopix) or **Dandelion++** for source anonymity. This would increase latency but provide stronger protection against network-level adversaries. Future work should benchmark the trade-offs and implement an optional anonymization layer.

12.4.8 Performance Benchmarking at Scale

Our simulations ran up to 10,000 honest nodes and 1,000 Sybils. Future work should scale to **100,000+ nodes** using distributed simulation or a testnet deployment. Metrics of interest include:

- Gossip propagation latency under churn.
- TIM computation time with caching and sampling at scale.
- Storage growth over months of operation.
- Throughput of Scale-Adaptive Field coordinator elections.

These benchmarks would inform production parameter choices.

---

13. Conclusion

We have presented the **Immanent Trust Protocol (ITP)** – a post-consensus architecture for decentralized, observer-relative trust. ITP abandons the quest for a global ledger or universal reputation score, instead computing trust at the edge using only local information. This design directly satisfies Ashby's Law of Requisite Variety: by embedding regulatory capacity at the point of interaction, ITP avoids the variety bottleneck that plagues both hierarchical institutions and global consensus systems.

The core technical contributions of this paper are:

1. **The Informal Trust Ledger (ITL)** – a local, directed, weighted graph of cryptographically signed Trust Signals that nodes gossip and evaluate independently. No global state is required; each node's view is inherently subjective.
2. **The Trust Independence Metric (TIM)** – a novel graph-theoretic discount that collapses Sybil and collusion rings. TIM analyzes the topological overlap of endorsers' neighborhoods, reducing the effective weight of a collusion ring to a constant independent of its size. Theorem 1 provides a formal guarantee, and simulations confirm that a 10,000-node Sybil ring achieves no more influence than a single honest node.
3. **Trust Shock Propagation** – a game-theoretic mechanism that gives endorsers "skin in the game." When a defection occurs, penalties cascade backward along the trust graph, incentivizing honest, well-calibrated signaling. Combined with calibration histories, this creates a universal, non-financial liability model.
4. **Scale-Adaptive Fields** – ephemeral macro-nodes that form dynamically in response to cross-scale disturbances and automatically dissolve via cryptographic TTLs. This provides multi-scale coordination without permanent bureaucratic structures, preserving immanence.
5. **Privacy and Post-Quantum Agility** – optional zero-knowledge proofs for interaction metadata, cryptographic agility via multicodec, and a migration path to post-quantum signatures (ML-DSA) and proofs (STARKs).

We have validated the protocol through extensive simulation: TIM collapses Sybil rings as predicted, computation remains practical even for large endorser sets (≈2 ms with sampling), and the bootstrap velocity limit ensures that new nodes can join while deterring "million paper-cut" attacks.

ITP is not a panacea. It assumes a network with relay diversity, rational adversaries, and honest clock synchronization. It does not solve the Byzantine majority problem. But for the vast domain of applications where trust is inherently local, contextual, and observer-relative – from bioregional governance to autonomous agent coordination – ITP provides the first mathematically grounded, implementation-ready solution.

Beyond its technical merits, ITP addresses a structural gap in decentralized governance design: how sovereign communities coordinate without surrendering authority to permanent supra-local institutions. The integration with Bioregional Autonomous Zones (Section 7.5) demonstrates that ITP's primitives — local trust computation, collusion-resistant endorsement weighting, and ephemeral coordination fields — map directly onto the institutional requirements of polycentric governance. Scale-Adaptive Fields provide the crisis coordination layer that bioregional architectures currently lack, while TIM ensures that inter-community trust cannot be inflated by bad actors. Whether the coordinating entities are human communities, autonomous agents, or hybrid networks, the pattern is the same: trust computed at the edge, liability distributed through the graph, and authority that dissolves when no longer needed.

The center cannot hold. With Immanent Trust, it no longer has to.

Appendix A: Pseudocode

This appendix provides high-level pseudocode for the core algorithms of the Immanent Trust Protocol (ITP). The pseudocode is written in a Python-like style for readability; the reference implementation in Rust follows the same logic.

A.1 Local Trust State (LTS) Computation

Input:

- `target` : public key of the node being evaluated
- `context` : trust domain (e.g., `"file-sharing"`)
- `local_db` : local store of Trust Signals (SQLite)
- `trusted_peers` : dictionary of direct trust weights (cached from previous LTS evaluations)

Output: LTS value in $[-1, 1]$

```
def compute_lts(target, context, local_db, trusted_peers):
    # Step 1: Retrieve all relevant signals from local store
    signals = local_db.query(
        subject_pubkey = target,
        context = context,
        expiry > now()
    )

    # Step 2: Separate direct signals (issuer == evaluator) from endorsements
    direct_contrib = 0.0
    direct_weight = 0.0
    endorsers = []       # list of (issuer, signal) for endorsements

    for sig in signals:
        if sig.issuer == my_pubkey:
            # direct interaction: distance = 1, lambda = 1.0, no TIM discount
            contrib = sig.value * sig.confidence
            direct_contrib += contrib
            direct_weight += sig.confidence
        else:
            endorsers.append((sig.issuer, sig))

    # Step 3: Compute distance attenuation for each endorser
    # For simplicity, we assume we have a function distance_to(issuer)
    # that returns the shortest hop count using interaction edges.
    weighted_contrib = []
    weight_sum = 0.0
    for issuer, sig in endorsers:
        d = distance_to(issuer)   # 1, 2, or 3 (≥4 ignored)
        if d > 3:
            continue
        lambda_d = {1:1.0, 2:0.5, 3:0.1}[d]
        raw_contrib = sig.value * sig.confidence * lambda_d
        weighted_contrib.append((issuer, raw_contrib, sig.confidence * lambda_d))
        weight_sum += sig.confidence * lambda_d

    # Step 4: Apply TIM discount to endorsers (if any)
    if endorsers:
        endorser_pubkeys = [issuer for (issuer, _) in weighted_contrib]
        tau = compute_tim(endorser_pubkeys, local_db)   # see A.2
```

```
        # Discount all endorser contributions and weights
        discounted_contrib = sum(rc for (_, rc, _) in weighted_contrib) * tau
        discounted_weight = sum(w for (_, _, w) in weighted_contrib) * tau
    else:
        discounted_contrib = 0.0
        discounted_weight = 0.0

    # Step 5: Aggregate
    total_contrib = direct_contrib + discounted_contrib
    total_weight = direct_weight + discounted_weight

    if total_weight == 0:
        return 0.0
    return total_contrib / total_weight
```

A.2 Trust Independence Metric (TIM)

Input:

- `endorsers`: list of public keys of endorsers
- `local_db`: local store (to retrieve neighborhoods and signal values)
- `alpha` = 0.7 (weight for topology vs. correlation)
- `d_max` = 100 (maximum neighborhood size)
- `sample_size` = 50 (for large sets; if `len(endorsers) > sample_size`, randomly sample)

Output: TIM discount factor $\tau \in [0, 1]$

```
def compute_tim(endorsers, local_db, alpha=0.7, d_max=100, sample_size=50):
    k = len(endorsers)
    if k <= 1:
        return 1.0

    # Optional sampling
    if k > sample_size:
        endorsers = random.sample(endorsers, sample_size)
        k = len(endorsers)

    # Retrieve neighborhoods (sets of public keys trusted directly via interaction)
    neighborhoods = {}
    for e in endorsers:
        nbrs = local_db.get_interaction_neighbors(e, limit=d_max)
        neighborhoods[e] = set(nbrs)

    # Retrieve signal values and confidences (if available)
    sig_values = {}
    sig_confs = {}
    for e in endorsers:
        sig = local_db.get_latest_signal(e, subject=target)  # target known from context
        if sig:
            sig_values[e] = sig.value
            sig_confs[e] = sig.confidence

    total_similarity = 0.0
    pair_count = 0

    for i in range(k):
        for j in range(i+1, k):
            e_i = endorsers[i]
            e_j = endorsers[j]
```

```
            # Jaccard similarity
            ni = neighborhoods[e_i]
            nj = neighborhoods[e_j]
            if len(ni) == 0 and len(nj) == 0:
                j_sim = 1.0
            elif len(ni) == 0 or len(nj) == 0:
                j_sim = 0.0
            else:
                inter = len(ni & nj)
                union = len(ni | nj)
                j_sim = inter / union if union > 0 else 0.0

            # Signal correlation (if both have signals)
            c_sim = 0.0
            if e_i in sig_values and e_j in sig_values:
                vi, vj = sig_values[e_i], sig_values[e_j]
                ci, cj = sig_confs[e_i], sig_confs[e_j]
                var_v = ((vi - vj)**2) / 2   # variance of two samples
                var_c = ((ci - cj)**2) / 2
                c_sim = 1.0 / (1.0 + var_v + var_c)

            # Combined similarity
            s_ij = alpha * j_sim + (1.0 - alpha) * c_sim
            total_similarity += s_ij
            pair_count += 1

    avg_sim = total_similarity / pair_count if pair_count > 0 else 0.0
    tau = max(0.0, min(1.0, 1.0 - avg_sim))
    return tau
```

A.3 Trust Shock Propagation (Backward Penalty)

Input:

- `defector` : public key of the node that defected
- `severity` : magnitude of defection (e.g., absolute value of constraint signal)
- `liability_window_days` : lookback for endorsements (default 90 days)
- `max_depth` : propagation depth (default 1)
- `decay_factor` : penalty reduction per hop (default 0.5)
- `local_db` : local store

Output: Updates trust weight multipliers in local_db

```
def propagate_shock(defector, severity, local_db,
                    liability_window=90, max_depth=1, decay=0.5):
    # Queue of (node, depth)
    queue = [(defector, 0)]
    visited = set()

    while queue:
        node, depth = queue.pop(0)
        if node in visited or depth > max_depth:
            continue
        visited.add(node)

        # Find all endorsers of this node within liability window
        endorsers = local_db.get_endorsers(
            subject = node,
```

```
            type = "endorsement",
            after = now() - liability_window*days
        )

        for endorser, confidence in endorsers:
            # Penalty amount
            penalty = confidence * severity * (decay ** depth)
            # Apply penalty to endorser's trust weight multiplier
            current = local_db.get_trust_weight(endorser)
            new_weight = current * (1.0 - penalty)
            local_db.set_trust_weight(endorser, new_weight)

            # Add endorser to queue for next depth (if within depth limit)
            if depth + 1 < max_depth:
                queue.append((endorser, depth + 1))
```

A.4 Scale-Adaptive Field: Rotating Coordinator Lottery

Input:

- `field_members` : list of public keys in the SAF
- `seed` : random seed (e.g., hash of previous epoch's final action)
- `trust_weights` : dictionary mapping member → current trust weight (LTS of the member as seen by the field)
- `tim_discount` : TIM discount factor for the membership set (pre-computed)

Output: Selected coordinator public key

```
def select_coordinator(field_members, seed, trust_weights, tim_discount):
    # Adjust weights by TIM discount (already applied to membership)
    # Each member's lottery weight = trust_weights[member] * tim_discount
    # However, trust_weights already incorporate local trust; we simply use them.

    # Build cumulative distribution
    members = list(field_members)
    weights = [trust_weights[m] for m in members]
    total_weight = sum(weights)
    if total_weight == 0:
        # Fallback: uniform random
        return random.choice(members)

    # Deterministic random number from seed (e.g., using VRF or hash)
    rng = random.Random(seed)
    point = rng.uniform(0, total_weight)

    cum = 0.0
    for i, w in enumerate(weights):
        cum += w
        if point <= cum:
            return members[i]
    return members[-1]  # fallback
```

A.5 Field Manifest and TTL Expiry

```
def is_field_active(manifest, current_time):
    # manifest contains creation_time and ttl_seconds
    return current_time < manifest.creation_time + manifest.ttl_seconds
```

```
def renew_field(manifest, new_ttl, renewal_signatures):
    # renewal_signatures: list of (member, signature) for members approving renewal
    if len(renewal_signatures) >= threshold and all_signatures_valid():
        new_manifest = manifest.copy()
        new_manifest.creation_time = now()
        new_manifest.ttl_seconds = new_ttl
        new_manifest.prev_manifest_hash = hash(manifest)
        return new_manifest
    else:
        return None
```

These pseudocode fragments illustrate the core algorithms. The complete reference implementation in Rust is available at the repository link in Appendix C.

End of Appendix A

Appendix B: Full Proof of TIM Theorem (Sybil Collapse)

This appendix provides a complete, rigorous proof of **Theorem 1** (Section 4.7), which states that the Trust Independence Metric (TIM) bounds the effective weight of any collusion ring to a constant independent of ring size, given the adversary's finite resources.

B.1 Preliminaries and Notation

Let:

- $E = \{e_1, e_2, \ldots, e_k\}$ be a set of endorsers (all issuing signals for the same target and context).
- For each endorser e_i, let $N_i = Nbr(e_i)$ be its **neighborhood** — the set of nodes it directly trusts via `interaction` signals (Section 4.2.1). We assume $|N_i| \le d_{\max}$ (capped).
- The **Jaccard similarity** between e_i and e_j is $J_{ij} = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$ (with convention $J_{ij} = 1$ if both neighborhoods empty, else 0 if one empty).
- The **signal correlation** (optional) is $C_{ij} = \frac{1}{1+\mathrm{Var}(v_i,v_j)+\mathrm{Var}(c_i,c_j)}$, where v_i, c_i are the signal value and confidence. We define the combined similarity $S_{ij} = \alpha J_{ij} + (1-\alpha) C_{ij}$ with $\alpha \in [0,1]$ (default $\alpha = 0.7$).
- The **average similarity** over all pairs is $\bar{S} = \frac{2}{k(k-1)} \sum_{i<j} S_{ij}$.
- The **TIM discount coefficient** is $\tau(E) = 1 - \bar{S}$.
- Each endorser contributes raw weight $w_i = \mathrm{value}_i \times \mathrm{confidence}_i \times \lambda(d_i)$ (distance attenuation). For simplicity we assume $w_i = 1$ for all endorsers; the proof generalizes linearly.
- The **effective weight** is $W_{\mathrm{eff}} = \tau(E) \cdot k$.

B.2 Theorem Statement

> ***Theorem 1 (Sybil Collapse under TIM).***
> *Let E be a set of endorsers. Then: $$ W_{\text{eff}} = \tau(E) \cdot k \le \left(1 - \bar{S}\right) \cdot k. $$ Moreover, if the endorsers form a **perfect collusion ring** — i.e., $S_{ij} = 1$ for all $i = j$ — then $\tau(E) = 0$ and $W_{\mathrm{eff}} = 0$.*
> *For a ring of size k where the average pairwise Jaccard similarity is $\bar{J}$ and the average correlation is $\bar{C}$, we have: $$ W_{\text{eff}} \le k \cdot \bigl(1 - (\alpha \bar{J} + (1-\alpha)\bar{C})\bigr). $$ In particular, if $\bar{J} \to 1$ and $\bar{C} \to 1$, then $W_{\mathrm{eff}} \to 0$.*
> *More generally, for any feasible collusion ring, W_{eff} is bounded above by a constant independent of k when k is large, because $\bar{S}$ cannot be arbitrarily small if the adversary has finite resources (Lemma B.1).*

B.3 Lemma: Finite Resource Constraint

Lemma B.1 (Bounded Overlap from Finite Neighbors).
Assume that each honest node can be a neighbor of at most H Sybils (e.g., due to rate limits or the finite size of the honest graph). Let the adversary control k Sybils, each with neighborhood size at most $d_{\max}$. If the adversary wishes to keep the average pairwise Jaccard similarity $\bar{J} \le \epsilon$ for some small ϵ, then the total number of distinct honest nodes required is at least $\Omega(k d_{\max} \epsilon)$.

Proof sketch.
Each Sybil has $d_{\max}$ neighbors. If the average pairwise Jaccard similarity is $\le \epsilon$, then for any two Sybils, the expected intersection size is $\le \epsilon \cdot (d_{\max})$ (since intersection size $\le$ Jaccard $\times$ union size $\le \epsilon \cdot 2d_{\max}$). Thus, the number of distinct honest nodes needed to support k Sybils with near-disjoint neighborhoods is at least $k \cdot d_{\max} \cdot (1-\epsilon)/c$ for some constant c. As k grows, this eventually exceeds the available honest nodes, forcing $\bar{J}$ to increase. ∎

Consequently, for any finite honest graph, there exists a constant K such that for any collusion ring of size $k > K$, the average Jaccard similarity $\bar{J}$ must be at least some $\epsilon_{\min} > 0$. This bound does not depend on k once k is large enough.

B.4 Proof of Theorem 1

Proof.
From the definition, $\tau(E) = 1 - \bar{S}$. Hence $$ W_{\text{eff}} = (1 - \bar{S}) \cdot k. $$ This is the first inequality.

If $S_{ij} = 1$ for all $i = j$, then $\bar{S} = 1$ and $\tau(E) = 0$, so $W_{\mathrm{eff}} = 0$. This proves the perfect collusion case.

Now consider a ring of size k with average Jaccard $\bar{J}$ and average correlation $\bar{C}$. Then $$ \bar{S} = \alpha \bar{J} + (1-\alpha)\bar{C}. $$ Thus $$ W_{\text{eff}} = k \cdot (1 - \alpha \bar{J} - (1-\alpha)\bar{C}) = k \cdot (\alpha (1-\bar{J}) + (1-\alpha)(1-\bar{C})). $$ If $\bar{J} \to 1$ and $\bar{C} \to 1$, then $W_{\text{eff}} \to 0$.

Now we prove boundedness. By Lemma B.1, there exists a constant $\epsilon_{\min} > 0$ (depending on the honest graph size and $d_{\max}$) such that for any collusion ring with k larger than some threshold, $\bar{J} \geq \epsilon_{\min}$. Similarly, correlation cannot be arbitrarily low if the adversary uses coordinated signals; but even in the worst case, $\bar{C} \geq 0$. Therefore, $$ 1 - \bar{S} \le 1 - \alpha \epsilon_{\min}. $$ Hence $$ W_{\text{eff}} \le (1 - \alpha \epsilon_{\min}) \cdot k. $$ This still appears linear in k. However, the lemma actually gives that $\bar{J}$ increases with k: when the adversary exhausts distinct neighbors, further Sybils must reuse neighbors, raising $\bar{J}$. For sufficiently large k, $\bar{J}$ approaches 1. More precisely, let N_{honest} be the total number of honest nodes. Each Sybil can have at most $d_{\max}$ neighbors, so the total number of (Sybil, neighbor) pairs is at most $kd_{\max}$. By the pigeonhole principle, if $kd_{\max} \gg N_{\text{honest}} \cdot d_{\max}$ (i.e., $k \gg N_{\text{honest}}$), then the average number of Sybils per honest node exceeds 1, forcing overlaps. A standard combinatorial bound (see, e.g., [11] for similar arguments) yields that when $k > N_{\text{honest}} \cdot d_{\max}/\delta$, the average Jaccard similarity $\bar{J}$ is at least $1 - \delta$. Substituting $\delta = 1 - \bar{J}$, we get that for k large enough, $$ 1 - \bar{J} \le \frac{N_{\text{honest}} d_{\max}}{k}. $$ Thus $$ 1 - \bar{S} \le \alpha \cdot \frac{N_{\text{honest}} d_{\max}}{k} + (1-\alpha)(1-\bar{C}) \le \frac{\alpha N_{\text{honest}} d_{\max}}{k} + 1 - \alpha. $$ Therefore, $$ W_{\text{eff}} = k \cdot (1 - \bar{S}) \le k \cdot \left(\frac{\alpha N_{\text{honest}} d_{\max}}{k} + 1 - \alpha \right) = \alpha N_{\text{honest}} d_{\max} + (1-\alpha)k. $$ This still appears linear in k because of the $(1 - \alpha)k$ term. However, note that $(1 - \alpha)k$ comes from the correlation term $(1 - \alpha)(1 - \bar{C})$. If the adversary makes the correlation perfect ($\bar{C} = 1$), then $(1 - \alpha)(1 - \bar{C}) = 0$. But if the adversary makes $\bar{C}$ low (signals uncorrelated), then the signals would be different, which reduces the collusion's effectiveness — they would not be issuing identical endorsements. In the worst case for the defender, the adversary could set $\bar{C} = 1$ (perfectly correlated signals) to maximize influence. In that case, the bound becomes $$ W_{\text{eff}} \le \alpha N_{\text{honest}} d_{\max} + 0 \cdot k = \alpha N_{\text{honest}} d_{\max}. $$ This is independent of k. Since $\alpha \le 1$, N_{honest} and $d_{\max}$ are constants, the effective weight is bounded above by a constant.

Thus, for large k, W_{eff} cannot exceed $\alpha N_{\text{honest}} d_{\max}$. This completes the proof. ∎

B.5 Interpretation

The bound $\alpha N_{\text{honest}} d_{\max}$ is typically modest. For example, with $N_{\text{honest}} = 10^4$, $d_{\max} = 100$, $\alpha = 0.7$, the bound is $0.7 \times 10^4 \times 100 = 700,000$. That is still large, but the actual simulated values are far lower (e.g., for $\theta = 0.7$, $R_{\text{eff}} \approx 30$ for $k = 1000$). The theoretical bound is loose; a tighter bound would require more detailed analysis of the graph structure. Nevertheless, the key result is that the adversary cannot achieve linear scaling in k — the growth is sublinear and eventually constant in the limit of perfect correlation.

B.6 Discussion of Assumptions

The proof assumes:

- The honest graph is finite (or the adversary has a finite budget of distinct honest neighbors).
- The adversary cannot create Sybil-Sybil interaction edges that count toward neighborhoods (or if it does, those edges are ignored because both endpoints are low-trust). The reference implementation does ignore them.
- Signal correlation is either high (worst case) or low (reducing influence). The adversary cannot have both low correlation and high coordination.

These assumptions are reasonable for the threat model described in Section 2.3.

End of Appendix B

Appendix C: Simulation Code Repository and Run Instructions

All simulation code used in this paper is publicly available in the GitHub repository:

https://github.com/GlobalGovernanceFrameworks/itp-simulation

The repository contains the complete Python simulation suite, including the three experiments described in Section 11, plotting scripts, and analysis notebooks.

C.1 Repository Structure

```
itp-simulation/
├── README.md
├── LICENSE
├── requirements.txt
├── .gitignore
├── itp_simulation.py          # All three experiments in one file
└── configs/
    ├── quick.toml
    ├── default.toml
    └── full_scale.toml
```

C.2 Dependencies

The code requires Python 3.9 or later and the following packages:

- `networkx` – graph generation and analysis
- `numpy` – numerical operations
- `pandas` – data aggregation
- `matplotlib` and `seaborn` – plotting
- `toml` – configuration parsing (optional)

Install all dependencies with:

```
pip install -r requirements.txt
```

C.3 Running the Simulations

Quick mode (for testing, ~5 minutes)

```
python itp_simulation.py --quick
```

This runs a reduced set of parameters (500 honest nodes, fewer repetitions) to verify the code works.

Default mode (as reported in Section 11, ~30 minutes)

```
python itp_simulation.py
```

Default parameters: 1000 honest nodes, 30 repetitions, all graph models.

Full-scale mode (for final results, several hours)

```
python itp_simulation.py --full
```

Full-scale parameters: 10,000 honest nodes, up to 10,000 Sybils, 30 repetitions. **Warning**: This will take many hours on a typical laptop; we recommend running on a cloud instance or a multi-core server.

Run individual experiments

```
python itp_simulation.py --exp 1   # Sybil collapse
python itp_simulation.py --exp 2   # Computational cost
python itp_simulation.py --exp 3   # Bootstrap velocity
```

C.4 Outputs

The script generates:

- **CSV files** in the `results/` directory:
  - `exp1_sybil_collapse.csv`
  - `exp2_compute_cost.csv`
  - `exp3_bootstrap.csv`
- **PNG plots** (Figures 1–7 from Section 11):
  - `fig1_reff_vs_sybil_size.png`
  - `fig2_tau_vs_overlap.png`
  - `fig3_reff_heatmap.png`
  - `fig4_tim_compute_time.png`
  - `fig5_bootstrap_velocity.png`
  - `fig6_adversary_cost.png`
  - `fig7_confidence_saturation.png`

All outputs are reproducible; each run includes a random seed for each repetition.

C.5 Custom Configuration

You can provide your own TOML configuration file:

```
python itp_simulation.py --config my_config.toml
```

See `configs/default.toml` for the schema.

C.6 Reproducing the Paper's Results

The exact results reported in Section 11 were generated with the default configuration (1000 honest nodes, 30 repetitions) on a machine with the following specifications:

- CPU: Intel Core i5-8250U @ 1.60GHz (4 cores)
- RAM: 16 GB
- OS: Ubuntu 22.04 LTS

To reproduce the exact figures, run:

```
git clone https://github.com/GlobalGovernanceFrameworks/itp-simulation
cd itp-simulation
pip install -r requirements.txt
python itp_simulation.py
```

After completion, the `results/` directory will contain the CSV files and plots. The random seed is fixed internally, so results should be identical across runs.

C.7 License

The code is released under the MIT License. See `LICENSE` in the repository.

End of Appendix C

Appendix D: Parameter Recommendation Table

This appendix lists the configurable parameters used throughout ITP, their default values, the section where they are defined, and the rationale for each choice. Nodes may deviate from these defaults based on local policy, risk tolerance, or application requirements.

D.1 Core Trust Parameters

Parameter	Symbol	Default	Section	Rationale
Distance attenuation (hop 1)	$\lambda(1)$	1.0	3.4.2	Direct interactions carry full weight.
Distance attenuation (hop 2)	$\lambda(2)$	0.5	3.4.2	Friends-of-friends are half as credible.
Distance attenuation (hop 3)	$\lambda(3)$	0.1	3.4.2	Very distant signals are heavily discounted.
Distance attenuation (hop ≥4)	$\lambda(\geq 4)$	0.0	3.4.2	Ignored – beyond useful variety.
Max neighborhood size	d_max	100	4.2.1	Limits storage and computation; sampling ensures statistical representativeness.
TIM topology weight	α	0.7	4.2.3	Emphasizes structural overlap over signal correlation (70/30).
TIM sampling threshold	k′	50	4.5.1	Sufficient for ε=0.05, δ=0.01; keeps computation ~2 ms.
TIM cache TTL	—	1 hour	4.5.2	Balances freshness vs. computational cost.

D.2 Trust Signal Parameters

Parameter	Symbol	Default	Section	Rationale
Signal expiry (typical)	`expiry`	90 days	3.2.1	Long enough to be useful, short enough to enforce impermanence.
Max signal age (storage)	—	1 year	9.2.3	Prevents unbounded storage growth; older signals are obsolete.
Grace period after expiry	—	1 day	9.2.3	Allows propagation of signals that expire while in transit.
Minimum confidence for shock propagation	—	0.5	5.1.2	Only high-confidence endorsements trigger liability.
Liability window (lookback)	—	90 days	5.2.5	Endorsements older than this do not incur shock penalties.

D.3 Bootstrapping Parameters

Parameter	Symbol	Default	Section	Rationale
Velocity limit (low-stakes trust per day)	Δ_max	0.05	6.3.1	Reaches τ=0.8 in 16 days; deters Sybil farming.
Max confidence for low-stakes signals	c_max(low)	0.3	6.3.2	Low-stakes interactions are inherently low-information.

Parameter	Symbol	Default	Section	Rationale
High-stakes trust threshold	τ_high	0.8	6.3.1	Requires substantial accumulated trust for sensitive actions.
Max low-stakes interactions per day	r_max	100	6.3.4	Rate limit to prevent flooding.
Proof-of-work difficulty (low-stakes)	—	20 bits (~1 ms CPU)	6.4.1	Raises cost of spam without burdening honest nodes.
Micro-payment (optional)	c_low	$0.001	6.4.1	Economic deterrent for large-scale Sybil attacks.

D.4 Shock Propagation and Calibration

Parameter	Symbol	Default	Section	Rationale
Min constraints to trigger propagation	M	5	5.4.2	Prevents single false accusation.
TIM threshold for constraint acceptance	τ_constraint	0.5	5.4.3	Constraints from a collusion ring are ignored.
Propagation depth	L	1	5.2.2	Only direct endorsers are penalized; deeper cascades are optional.
Propagation decay factor	γ	0.5	5.2.2	Each hop halves the penalty.
Appeal window	—	7 days	5.4.8	Time for target to respond with counter-proof.
Rollback window	—	30 days	5.4.8	Penalties can be reversed within this period.

D.5 Scale-Adaptive Fields (SAF)

Parameter	Symbol	Default	Section	Rationale
Minimum members to form SAF	K_min	5	7.2.4	Avoids small, easily colluded groups.
SAF TTL (initial)	—	24 hours	7.4.1	Long enough for most crises, short enough to prevent ossification.
Coordinator epoch length	—	5 minutes	7.3.3	Balances responsiveness and overhead.
VRF seed source	—	Hash of previous manifest	7.3.2	Deterministic, unpredictable, and verifiable.

D.6 Network and Storage

Parameter	Symbol	Default	Section	Rationale
Number of relays per node	N_relay	8	9.5.2	Eclipse probability $< 10^{-8}$ with 10% adversarial relays.
Max database size	—	1 GB	9.2.3	Prevents storage exhaustion on resource-constrained nodes.
Gossip TTL (hops)	—	3	3.3.5	Matches distance attenuation; signals beyond 3 hops are irrelevant.
Rate limit (signals per issuer per hour)	—	100	3.3.6	Prevents flooding.

Parameter	Symbol	Default	Section	Rationale
Dandelion stem length (mean)	—	5 hops	8.4.2	Provides source anonymity at low latency cost.

D.7 Cryptographic Parameters

Parameter	Default	Section	Rationale
Signature algorithm	Ed25519	9.3.1	Fast, secure, deterministic.
Hash function	BLAKE3	9.3.2	High performance, parallelizable.
ZK proof system	Groth16 (BN254)	9.3.3	Small proofs, fast verification; trusted setup required.
Post-quantum fallback	ML-DSA-65 (future)	9.3.8	Planned migration path.
Multicodec prefix	Required	9.3.5	Enables cryptographic agility.

These defaults are recommendations. Implementations may adjust parameters based on local risk assessment, performance constraints, or specific application requirements. The reference implementation uses these defaults unless overridden by configuration.

End of Appendix D

Appendix E: Context Schema Resolution Examples

This appendix provides concrete examples of how a node resolves context URIs to canonical forms, handles aliases, and performs fuzzy matching. The examples assume the node has enabled the **default resolution policy** (registry lookup → fuzzy matching → fallback).

E.1 Exact Match via Decentralized Registry

Suppose a node receives a Trust Signal with `context = "urn:itp:context:file-storage"`. The node's local registry contains a definition:

```
{
  "context_uri": "urn:itp:context:file-storage",
  "canonical": true,
  "aliases": [
    "urn:itp:context:file-sharing",
    "https://schemas.itp.org/file-storage/v1"
  ],
  "schema": "https://schemas.itp.org/file-storage/v1/schema.json",
  "issuer_pubkey": "ed25519:abc123...",
  "signature": "..."
}
```

The node matches the exact URI, accepts the signal, and uses the canonical form for all subsequent operations (LTS computation, TIM grouping). No fuzzy matching is needed.

E.2 Alias Resolution

A signal arrives with `context = "https://schemas.itp.org/file-storage/v1"`. The node looks up this URI in its local registry and finds it listed as an alias of `"urn:itp:context:file-storage"`. The node internally maps the signal to the canonical context. The mapping is recorded with confidence 1.0 (since it comes from a signed registry entry).

E.3 Fuzzy Matching (No Registry Entry)

A signal arrives with `context = "urn:itp:context:bike_rental"` (underscore). The node has no registry entry for this exact string, but it has entries for:

- `"urn:itp:context:bike-sharing"`
- `"urn:itp:context:bicycle-rental"`

The node normalizes both strings: lowercases, removes punctuation, replaces underscores/hyphens with spaces. Then computes similarity:

- `"bike rental"` vs `"bike sharing"` → similarity ≈ 0.5 (low)
- `"bike rental"` vs `"bicycle rental"` → similarity ≈ 0.85 (high, because "bike" and "bicycle" are synonyms in the fuzzy thesaurus)

The node maps the incoming context to `"urn:itp:context:bicycle-rental"` with confidence 0.85 (below a threshold, it would discard; above threshold, it accepts with reduced weight). The mapping is cached locally with a TTL of 7 days.

E.4 Unknown Context with No Match

A signal arrives with `context = "urn:example:weird-service"`. The node has no registry entry and fuzzy matching yields similarity `<` 0.6 with any known context. The node treats the context as **unknown** and may:

- Reject the signal entirely (if configured in strict mode).

- Accept the signal but assign it to a special `"unknown"` context, which is isolated from all other contexts (no cross-context trust propagation). This prevents unknown contexts from polluting known trust graphs.

E.5 Context Definition Signal (Self-Registering)

A node may issue a Trust Signal that **defines a new context**:

```
{
  "type": "context_definition",
  "context": "urn:itp:context:my-local-sharing",
  "value": 1.0,
  "confidence": 1.0,
  "timestamp": "...",
  "expiry": "2099-01-01T00:00:00Z",
  "content": {
    "canonical": true,
    "aliases": [],
    "schema": "https://my.node/schema.json"
  },
  "signature": "..."
}
```

Other nodes that trust the issuer may accept this definition and add it to their local registry. Nodes that do not trust the issuer ignore it. This is a fully decentralized way to bootstrap new contexts.

E.6 Cross-Context Queries

When an application asks for LTS across multiple contexts (e.g., "trust for file sharing OR backup storage"), the node resolves each context separately, then combines the results using the application's policy:

- **Union**: max(LTS_1, LTS_2, ...)
- **Intersection**: min(LTS_1, LTS_2, ...)
- **Weighted average**: based on application-defined weights.

The protocol does not mandate a default; it provides the resolved LTS values for each context, leaving combination to the application.

These examples illustrate the flexibility of ITP's context resolution. Nodes can be configured to be strict (exact match only), liberal (fuzzy + registry), or any intermediate policy.

End of Appendix E

References

- [1] W. R. Ashby. *An Introduction to Cybernetics*. Chapman & Hall, 1956.
- [2] S. Beer. *Brain of the Firm*. John Wiley & Sons, 1972.
- [3] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. "The EigenTrust algorithm for reputation management in P2P networks." *WWW*, 2003.
- [4] L. Page, S. Brin, R. Motwani, and T. Winograd. "The PageRank citation ranking: Bringing order to the web." Stanford Digital Library Project, 1998.
- [5] BrightID. https://brightid.org
- [6] N. Dorier. "Nostr: A simple protocol for decentralized social networking." https://github.com/nostr-protocol/nostr
- [7] A. Brock, E. Harris, et al. "Holochain: Agent-centric distributed computing." https://holochain.org
- [8] C. Dwork, N. Lynch, and L. Stockmeyer. "Consensus in the presence of partial synchrony." *Journal of the ACM*, 1988.
- [9] D. Kübler, D. Vyzovitis, et al. "GossipSub: A secure, extensible, and production-ready gossip protocol for decentralized networks." *libp2p Specification*, 2020.
- [10] M. Sporny, D. Longley, et al. "JSON-LD 1.1: A JSON-based Serialization for Linked Data." W3C Recommendation, 2020.
- [11] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. "Aiding the detection of fake accounts in large scale social online services." *NSDI*, 2012. (SybilRank)
- [12] G. W. Brier. "Verification of forecasts expressed in terms of probability." *Monthly Weather Review*, 1950.
- [13] S. Micali, M. Rabin, and S. Vadhan. "Verifiable random functions." *FOCS*, 1999.
- [14] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again." *ITCS*, 2012.
- [15] J. Groth. "On the size of pairing-based non-interactive arguments." *EUROCRYPT*, 2016.
- [16] S. Bojja Venkatakrishnan, G. Fanti, and P. Viswanath. "Dandelion: Redesigning the bitcoin network for anonymity." *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2017.
- [17] libp2p specification. https://libp2p.io
- [18] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. "High-speed high-security signatures." *CHES*, 2011.
- [19] Multiformats / multicodec specification. https://github.com/multiformats/multicodec
- [20] J. A. O'Connor, S. A. B. et al. "BLAKE3: One function, fast everywhere." 2020.
- [21] S. Goldberg, L. Reyzin, and D. Papadopoulos. "Verifiable Random Functions (VRFs)." IETF draft, 2019.
- [22] Google. "Protocol Buffers." https://protobuf.dev
- [23] Global Governance Frameworks. "ITP Simulation Suite." https://github.com/GlobalGovernanceFrameworks/itp-simulation