Independent Submission
Internet-Draft
Intended status: Informational
Expires: 13 August 2026

C. Ayerbe Posada
ULISSY s.r.l.

9 February 2026

TRIP: Trajectory-based Recognition of Identity Proof
draft-ayerbe-trip-protocol-02

Abstract

This document specifies the Trajectory-based Recognition of Identity Proof (TRIP) protocol, a decentralized mechanism for establishing claims of physical-world presence through cryptographically signed, spatially quantized location attestations called "breadcrumbs." Breadcrumbs are chained into an append-only log, bundled into verifiable epochs, and distilled into a Trajectory Identity Token (TIT) that serves as a persistent pseudonymous identifier.

The protocol employs a Criticality Engine grounded in statistical physics to distinguish biological movement from synthetic trajectories. Power Spectral Density analysis detects the 1/f signature of Self-Organized Criticality in human mobility. A six-component Hamiltonian energy function scores each breadcrumb against the identity’s learned behavioral profile in real time.

This revision formalizes the mapping to the RATS Architecture (RFC 9334), clarifies the Verifier trust model including support for multiple independent Verifiers, introduces an active challenge-response verification protocol that binds Attestation Results to specific Relying Party requests with cryptographic freshness, and corrects the privacy model to accurately describe the flow of Evidence between Attester and Verifier.

TRIP is designed to be transport-agnostic and operates independently of any particular naming system, blockchain, or application layer.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

Conventional approaches to proving that an online actor corresponds to a physical human being rely on biometric capture, government-issued documents, or knowledge-based challenges. Each technique introduces a centralized trust anchor, creates honeypots of personally identifiable information (PII), and is susceptible to replay or deepfake attacks.

TRIP takes a fundamentally different approach: it treats sustained physical movement through the real world as evidence of embodied existence. A TRIP-enabled device periodically records its position as a "breadcrumb" -- a compact, privacy- preserving, cryptographically signed attestation that the holder of a specific Ed25519 key pair was present in a particular spatial cell at a particular time. An adversary who controls only digital infrastructure cannot fabricate a plausible trajectory because doing so requires controlling radio-frequency environments (GPS, Wi-Fi, cellular, IMU) at many geographic locations over extended periods.

Version -01 introduced a Criticality Engine grounded in Giorgio Parisi’s Nobel Prize-winning work on scale-free correlations [PARISI-NOBEL] and Albert-Laszlo Barabasi’s research on the fundamental limits of human mobility [BARABASI-MOBILITY].

This revision (-02) addresses three areas identified through expert review: it formalizes the mapping between TRIP components and the RATS Architecture [RFC9334], explicitly stating the Verifier trust model and multi-Verifier deployment assumptions; it introduces an active challenge-response verification protocol that provides cryptographic freshness guarantees for Attestation Results; and it corrects the privacy model to accurately describe that H3-quantized Evidence is transmitted to the Verifier, with privacy deriving from the lossy quantization transform rather than data locality.

This document specifies the data structures, algorithms, and verification procedures that constitute the TRIP protocol. It intentionally omits transport bindings, naming-system integration, and blockchain anchoring, all of which are expected to be addressed in companion specifications.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

Terms defined in the RATS Architecture [RFC9334] (Attester, Evidence, Verifier, Attestation Result, Relying Party) are used throughout this document with their RFC 9334 meanings. Additional terms specific to TRIP:

Breadcrumb A single, signed attestation of spatiotemporal presence.

The atomic unit of TRIP Evidence.

Trajectory An ordered, append-only chain of breadcrumbs produced by a single identity key pair.

Epoch A bundle of breadcrumbs (default 100) sealed with a Merkle root, forming a verifiable checkpoint.

Trajectory Identity Token (TIT) A pseudonymous identifier derived from an Ed25519 public key paired with trajectory metadata.

Criticality Engine The analytical subsystem that evaluates trajectory statistics for signs of biological Self-Organized Criticality (SOC). In RATS terms, the Criticality Engine is a component of the Verifier.

Hamiltonian (H) A weighted energy function that quantifies how much a new breadcrumb deviates from the identity's learned behavioral profile.

Anchor Cell An H3 cell where an identity has historically spent significant time (e.g., home, workplace).

Flock The set of co-located TRIP entities whose aggregate movement provides a reference signal for alignment verification.

Proof-of-Humanity (PoH) Certificate A compact Attestation Result containing only statistical exponents derived from the trajectory, with no raw location data.

1.3. Changes from -01

This section summarizes the substantive changes from draft-ayerbe-trip-protocol-01:

* Expanded the RATS Architecture mapping (Section 11) with explicit Verifier trust model, multi-Verifier deployment language, and Evidence flow description.

* Replaced the two-sentence replay protection text with a full section (Section 12) covering chain-level replay, attestation-result replay, and the Active Verification Protocol with CDDL schemas.

* Corrected the privacy model (Section 14) to state that H3-quantized Evidence is transmitted to the Verifier; privacy derives from the quantization transform, not data locality.

* Added Deployment Considerations (Section 15) addressing multi-Verifier operation, trust delegation, and federation.

* Moved RFC 9334 from informative to normative references.

2. Breadcrumb Data Structure

A breadcrumb is encoded as a CBOR map [RFC8949] with the following fields:

Key	CBOR Type	Description
0	uint	Index (sequence number)
1	bstr (32)	Identity public key (Ed25519)
2	uint	Timestamp (Unix seconds)
3	uint	H3 cell index
4	uint	H3 resolution (7-10)
5	bstr (32)	Context digest (SHA-256)
6	bstr (32) / null	Previous block hash
7	map	Meta flags
8	bstr (64)	Ed25519 signature

Table 1: Breadcrumb CBOR Fields

2.1. Spatial Quantization

The H3 geospatial indexing system [H3] partitions the Earth’s surface into hexagonal cells at multiple resolutions. TRIP employs resolutions 7 through 10:

Resolution	Avg. Area	Edge Length	Use Case
7	~5.16 km^2	~1.22 km	Rural / low-density
8	~0.74 km^2	~0.46 km	Suburban / general
9	~0.11 km^2	~0.17 km	Urban / high-density
10	~0.015 km^2	~0.07 km	Default / standard verification

Table 2: H3 Resolution Parameters

A conforming implementation MUST quantize raw GPS coordinates to an H3 cell before any signing or storage operation. Raw coordinates MUST NOT appear in breadcrumbs or in any protocol message transmitted between TRIP entities.

H3 resolution is a configurable protocol parameter. Implementations SHOULD default to resolution 10. Deployments MAY select alternative resolutions based on jurisdictional requirements, population density, and use-case sensitivity. Lower resolutions (larger cells) provide stronger location privacy at the cost of reduced spatial discrimination for trust computation.

2.2. Context Digest Computation

The context digest binds ambient environmental signals to the breadcrumb without revealing them. The digest is computed as follows:

1. Construct a pipe-delimited string of tagged components in the following order:

   * "h3:" followed by the H3 cell hex string

   * "ts:" followed by the timestamp bucketed to 5-minute intervals (floor(Unix_minutes / 5) * 5)

   * "wifi:" followed by the first 16 hex characters of SHA-256(sorted comma-joined BSSIDs), if Wi-Fi scan data is available

   * "cell:" followed by the first 16 hex characters of SHA-256(sorted comma-joined tower IDs), if cellular data is available

* "imu:" followed by the first 16 hex characters of SHA-256(IMU vector string), if inertial sensor data is available

2. Compute SHA-256 over the UTF-8 encoding of the resulting string.

Absent components MUST be omitted entirely, not represented as empty strings.

2.3. Signature Production

The signable payload is the deterministic CBOR encoding (per Section 4.2 of [RFC8949]) of a CBOR map containing fields 0 through 7, with map keys sorted in ascending integer order. The Ed25519 signature [RFC8032] is computed over the raw bytes of this CBOR encoding and stored at key 8.

```
signable_payload = CBOR-Deterministic(fields[0..7])
signature        = Ed25519-Sign(private_key, signable_payload)
```

Deterministic CBOR encoding ensures that any conforming implementation produces identical byte sequences for the same logical content, which is essential for reproducible signature verification across heterogeneous platforms.

2.4. Block Hash and Chaining

The block hash is the SHA-256 digest of the complete deterministic CBOR encoding of the breadcrumb (fields 0 through 8 inclusive, i.e., including the signature):

```
BreadcrumbHash(B) = SHA-256(CBOR-Deterministic(B[0..8]))
B[N+1].field[6]   = BreadcrumbHash(B[N])
B[0].field[6]     = null
```

Each breadcrumb at index > 0 MUST carry the block hash of its immediate predecessor in field 6, forming an append-only hash chain. The genesis breadcrumb (index 0) MUST set field 6 to null (CBOR simple value 22).

3. Chain Management

3.1. Location Deduplication

Proof-of-Trajectory requires demonstrated movement. A conforming implementation MUST reject a breadcrumb if the H3 cell is identical to the immediately preceding breadcrumb. Implementations SHOULD also enforce a cap (default 10) on the number of breadcrumbs recordable at any single H3 cell to prevent stationary farming.

3.2. Minimum Collection Interval

Breadcrumbs SHOULD be collected at intervals of no less than 15 minutes. An implementation MAY allow shorter intervals during explicit "exploration" sessions but MUST NOT accept intervals shorter than 5 minutes.

3.3. Chain Verification

A Verifier MUST check:

1. Index values form a contiguous sequence starting at 0.
2. Timestamps are monotonically non-decreasing.
3. Each previousHash matches the block hash of the prior breadcrumb.
4. Each Ed25519 signature verifies against the identity public key and the canonical signed data.

4. Epochs

An epoch seals a batch of breadcrumbs (default 100) under a Merkle root. The epoch record is a CBOR map containing:

Key	Type	Description
0	uint	Epoch number
1	bstr (32)	Identity public key
2	uint	First breadcrumb index
3	uint	Last breadcrumb index
4	uint	Timestamp of first breadcrumb
5	uint	Timestamp of last breadcrumb
6	bstr (32)	Merkle root of breadcrumb hashes
7	uint	Count of unique H3 cells
8	bstr (64)	Ed25519 signature over fields 0-7

Table 3: Epoch CBOR Fields

The Merkle tree MUST use SHA-256 and a canonical left-right ordering of breadcrumb block hashes. An epoch is sealed when the breadcrumb count reaches the epoch size threshold.

5. Trajectory Identity Token (TIT)

A TIT is the externally presentable identity derived from a TRIP trajectory. It consists of:

* The Ed25519 public key (32 bytes).
* The current epoch count.
* The total breadcrumb count.
* The count of unique H3 cells visited.
* A trust score (see Section 10).

A TIT SHOULD be encoded as a CBOR map for machine consumption and MAY additionally be represented as a Base64url string for URI embedding.

6. The Criticality Engine

The Criticality Engine is the core analytical component of the TRIP Verifier. It evaluates whether a trajectory exhibits the statistical signature of biological Self-Organized Criticality (SOC) -- the phenomenon where living systems operate at the boundary between order and chaos, producing scale-free correlations that are mathematically distinct from synthetic or automated movement.

The theoretical foundation rests on three pillars:

First, Parisi’s demonstration [PARISI-NOBEL] that flocking organisms such as starling murmurations exhibit scale-free correlations [CAVAGNA-STARLINGS] where perturbations propagate across the entire group regardless of size. Crucially, Ballerini et al. showed that these interactions are topological (based on nearest k neighbors) rather than metric (based on distance) [BALLERINI-TOPOLOGICAL]. TRIP exploits this through Power Spectral Density analysis (Section 6.1): human movement produces characteristic 1/f pink noise that synthetic trajectories cannot replicate.

Second, Barabasi et al.’s discovery [BARABASI-MOBILITY] that human displacement follows truncated Levy flights with approximately 93% predictability [SONG-LIMITS]. TRIP learns each identity’s mobility profile -- displacement distribution, anchor transition patterns, and circadian rhythms -- and detects deviations from these learned baselines (Section 7).

Third, a six-component Hamiltonian energy function (Section 8) that combines spatial, temporal, kinetic, flock-alignment, contextual, and structural analysis into a single anomaly score for each incoming breadcrumb. The Hamiltonian provides real-time detection while the PSD and mobility statistics provide aggregate trajectory assessment.

6.1. Power Spectral Density Analysis

The primary diagnostic is the Power Spectral Density (PSD) of the displacement time series. Given a trajectory of N breadcrumbs with displacements d(i) between consecutive breadcrumbs, the PSD is computed via the Discrete Fourier Transform:

```
S(f) = |DFT(d)|^2

where d = [d(0), d(1), ..., d(N-1)]
and d(i) = haversine_distance(cell(i), cell(i-1))
```

The PSD is then fitted to a power-law model:

```
S(f) ~ 1 / f^alpha
```

The exponent alpha (the "Parisi Factor") is the critical diagnostic:

Alpha Range	Noise Type	Classification
0.00 - 0.15	White noise	Synthetic / automated script
0.15 - 0.30	Near-white	Suspicious (possible sophisticated bot)
0.30 - 0.80	Pink noise (1/f)	Biological / human
0.80 - 1.20	Near-brown	Suspicious (possible replay with drift)
1.20+	Brown noise	Drift anomaly / sensor failure

Table 4: PSD Alpha Exponent Classification

A conforming implementation MUST compute the PSD alpha exponent over a sliding window of the most recent 64 breadcrumbs (minimum) to 256 breadcrumbs (recommended). The alpha value MUST fall within [0.30, 0.80] for the trajectory to be classified as biological.

The key insight is that automated movement generators lack the long-range temporal correlations ("memory") inherent in a system operating at criticality. A random walk produces white noise (alpha near 0). A deterministic replay produces brown noise (alpha near 2). Only a biological system operating at the critical point produces pink noise in the characteristic [0.30, 0.80] range.

6.2. Criticality Confidence Score

The Criticality Confidence is a value in [0, 1] computed from the alpha exponent and the goodness-of-fit (R-squared) of the power-law regression:

```
alpha_score = 1.0 - |alpha - 0.55| / 0.25

criticality_confidence = alpha_score * R_squared

where:
  0.55 is the center of the biological range
  0.25 is the half-width of the biological range
  R_squared is the coefficient of determination of the
    log-log linear regression
```

A criticality_confidence below 0.5 SHOULD trigger elevated monitoring. A value below 0.3 SHOULD flag the trajectory for manual review or additional verification challenges.

7. Mobility Statistics

This section defines the mobility model that enforces known constraints of human movement, as established by Barabasi et al. [BARABASI-MOBILITY].

7.1. Truncated Levy Flights

Human displacement between consecutive recorded locations follows a truncated power-law distribution:

```
P(delta_r) ~ delta_r^(-beta) * exp(-delta_r / kappa)
```

where:

- delta_r = displacement distance (km)
- beta = power-law exponent (typically 1.50 - 1.90)
- kappa = exponential cutoff distance (km)

The exponent beta captures the heavy-tailed nature of human movement: most displacements are short (home to office) but occasional long jumps (travel) follow a predictable distribution. The cutoff kappa is learned per identity and represents the characteristic maximum range.

A conforming implementation MUST maintain a running estimate of beta and kappa for each identity by fitting the displacement histogram using maximum likelihood estimation over the most recent epoch (100 breadcrumbs).

A new displacement that falls outside the 99.9th percentile of the fitted distribution MUST increment the spatial anomaly counter.

7.2. Trajectory Predictability

Research has demonstrated that approximately 93% of human movement is predictable based on historical patterns [SONG-LIMITS]. TRIP exploits this by maintaining a Markov Transition Matrix over anchor cells:

```
T[a_i][a_j] = count(transitions from a_i to a_j)
                / count(all departures from a_i)
```

where a_i, a_j are anchor cells.

An anchor cell is defined as any H3 cell where the identity has recorded 5 or more breadcrumbs. The transition matrix is rebuilt at each epoch boundary.

The predictability score Pi for an identity is the fraction of observed transitions that match the highest-probability successor in the Markov matrix. Human identities converge toward Pi values in the range [0.80, 0.95] after approximately 200 breadcrumbs. Deviations below 0.60 are anomalous.

7.3. Circadian and Weekly Profiles

The implementation SHOULD maintain two histogram profiles:

* A circadian profile C[hour] recording the probability of activity in each hour of the day (24 bins).

* A weekly profile W[day] recording the probability of activity on each day of the week (7 bins).

These profiles provide the temporal baseline for the Hamiltonian temporal energy component (Section 8.2).

8. The Six-Component Hamiltonian

To assess each incoming breadcrumb, the Criticality Engine computes a weighted energy score H that quantifies how much the breadcrumb deviates from the identity’s learned behavioral profile. High energy indicates anomalous behavior; low energy indicates normalcy.

```
H = w_1 * H_spatial
  + w_2 * H_temporal
  + w_3 * H_kinetic
  + w_4 * H_flock
  + w_5 * H_contextual
  + w_6 * H_structure
```

Default weights:

Component	Weight	Diagnostic Target
H_spatial	0.25	Displacement anomalies (teleportation)
H_temporal	0.20	Circadian rhythm violations
H_kinetic	0.20	Anchor transition improbability
H_flock	0.15	Misalignment with local human flow
H_contextual	0.10	Sensor cross-correlation failure
H_structure	0.10	Chain integrity and timing regularity

Table 5: Hamiltonian Component Weights

Weights are modulated by the profile maturity m, defined as min(breadcrumb_count / 200, 1.0). During the bootstrap phase (m < 1.0), all weights are scaled by m, widening the acceptance threshold for new identities.

8.1. H_spatial: Displacement Anomaly

Given the identity's fitted truncated Levy distribution P(delta_r), the spatial energy for a displacement delta_r is the negative log-likelihood (surprise):

```
H_spatial = -log(P(delta_r))
```

where P(delta_r) = C * delta_r^(-beta) * exp(-delta_r / kappa) and C is the normalization constant.

Typical displacements yield H_spatial near the identity's historical baseline. A displacement that exceeds the identity's learned kappa cutoff by more than a factor of 3 produces an H_spatial value in the CRITICAL range.

8.2. H_temporal: Rhythm Anomaly

Using the circadian profile C[hour] and weekly profile W[day]:

```
H_temporal = -log(C[current_hour]) - log(W[current_day])
```

Activity at 3:00 AM for an identity with a 9-to-5 circadian profile yields high H_temporal. Activity at 8:00 AM on a Tuesday for the same identity yields low H_temporal.

8.3. H_kinetic: Transition Anomaly

Using the Markov Transition Matrix T:

```
from_anchor = nearest anchor to previous breadcrumb
to_anchor   = nearest anchor to current breadcrumb
H_kinetic   = -log(max(T[from_anchor][to_anchor], epsilon))

where epsilon = 0.001 (floor to prevent log(0))
```

A home-to-office transition at 8:00 AM yields low H_kinetic. An office-to-unknown-city transition yields high H_kinetic.

8.4. H_flock: Topological Alignment

Inspired by Parisi's finding that starlings track their k nearest topological neighbors (k approximately 6-7) rather than all birds within a metric radius [PARISI-NOBEL], the flock energy measures alignment between the identity's velocity vector and the aggregate velocity of co-located TRIP entities.

```
v_self  = displacement vector of current identity
v_flock = mean displacement vector of k nearest
          co-located identities (k = 7)

alignment = dot(v_self, v_flock)
            / (|v_self| * |v_flock|)

H_flock = 1.0 - max(alignment, 0)
```

When flock data is unavailable (sparse network or privacy constraints), the implementation SHOULD fall back to comparing the current velocity against the identity's own historical velocity distribution at the same location and time-of-day.

H_flock defeats GPS replay attacks: an adversary replaying a previously recorded trajectory will find that the ambient flock has changed since the recording, producing a misalignment signal.

8.5. H_contextual: Sensor Cross-Correlation

This component compares the IMU (accelerometer, gyroscope) signature against the claimed GPS displacement. A genuine device in motion produces correlated IMU and GPS readings. GPS injection on a stationary device is detected by the absence of corresponding IMU activity:

```
H_contextual = divergence(
  observed_imu_magnitude,
  expected_imu_magnitude_for(gps_displacement)
)
```

Implementations that lack IMU access MUST set H_contextual = 0 and SHOULD increase the weights of other components proportionally.

8.6. H_structure: Chain Structural Integrity

This component evaluates the structural properties of the breadcrumb chain itself:

* Inter-breadcrumb timing regularity: excessively uniform intervals suggest automation.

* Hash chain continuity: any break in the chain produces maximum H_structure.

* Phase-space smoothness: the velocity-acceleration phase portrait of a human trajectory traces smooth loops, while bots produce either chaotic blobs or tight limit cycles.

8.7. Alert Classification

The total Hamiltonian H maps to an alert level. The baseline H_baseline is the rolling median of the identity's own recent energy values, making the threshold self-calibrating per identity:

H Range	Level	Action
[0, H_baseline * 1.5)	NOMINAL	Normal operation
[H_baseline * 1.5, 3.0)	ELEVATED	Increase sampling frequency, log
[3.0, 5.0)	SUSPICIOUS	Flag for review, require reconfirmation
[5.0, infinity)	CRITICAL	Freeze trust score, trigger challenge

Table 6: Hamiltonian Alert Levels

9. Proof-of-Humanity Certificate

A PoH Certificate is a compact, privacy-preserving Attestation Result (in the RATS sense) asserting that an identity has demonstrated biological movement characteristics. It contains ONLY statistical exponents derived from the trajectory -- no raw location data, no GPS coordinates, no cell identifiers.

The certificate is encoded as a CBOR map:

Key	Type	Description
0	bstr (32)	Identity public key
1	uint	Issuance timestamp
2	uint	Epoch count at issuance
3	float	PSD alpha exponent
4	float	Levy beta exponent
5	float	Levy kappa cutoff (km)
6	float	Predictability score Pi
7	float	Criticality confidence
8	float	Trust score T
9	uint	Unique cell count
10	uint	Total breadcrumb count
11	uint	Validity duration (seconds)
12	bstr (16) / null	Relying Party nonce (if active verification)
13	bstr (32) / null	Chain head hash at issuance
14	bstr (64)	Verifier Ed25519 signature

Table 7: PoH Certificate CBOR Fields

Fields 12 and 13 are populated only when the certificate is issued in response to an Active Verification request (Section 12.3). In Passive mode, these fields MUST be null.

A Relying Party receiving a PoH Certificate can verify:

1. The Verifier signature (field 14) is valid against a trusted Verifier public key.

2. The alpha exponent (field 3) falls within [0.30, 0.80].

3. The criticality confidence (field 7) exceeds the Relying Party's policy threshold.

4. The trust score (field 8) meets application requirements.

5. The certificate has not expired (field 1 + field 11 > current time).

6. If Active mode: the nonce (field 12) matches the Relying Party's original challenge, and the chain head hash (field 13) provides freshness binding.

The certificate reveals NOTHING about where the identity has been -- only that it has moved through the world in a manner statistically consistent with a biological organism.

10. Trust Scoring

The trust score T is computed as a weighted combination of four factors:

```
T = 0.40 * min(breadcrumb_count / 200, 1.0)
  + 0.30 * min(unique_cells / 50, 1.0)
  + 0.20 * min(days_since_first / 365, 1.0)
  + 0.10 * chain_integrity

chain_integrity = 1.0 if chain verification passes, else 0.0
T is expressed as a percentage in [0, 100].
```

The threshold for claiming a handle (binding a human-readable name to a TIT) requires breadcrumb_count >= 100 and T >= 20.

In the Parisi percolation model, the trust score also incorporates the criticality confidence from the PSD analysis. A trajectory that fails the criticality test (alpha outside [0.30, 0.80]) MUST have its trust score capped at 50, regardless of other factors.

11. RATS Architecture Mapping

TRIP implements the Remote ATtestation procedureS (RATS) architecture defined in [RFC9334]. This section provides the normative mapping between TRIP components and RATS roles.

11.1. Role Mapping

RATS Role	TRIP Component	Description
Attester	TRIP-enabled mobile device	Collects breadcrumbs, signs them with the identity Ed25519 private key, chains them into the append-only trajectory log, and transmits H3-quantized Evidence to the Verifier.
Evidence	Breadcrumbs and epoch records	H3-quantized spatiotemporal claims including cell identifiers, timestamps, context digests, chain hashes, and Ed25519 signatures. Evidence is transmitted from Attester to Verifier.
Verifier	Criticality Engine	Receives Evidence, performs chain verification, computes PSD alpha exponents, fits Levy flight parameters, evaluates the six-component Hamiltonian, and produces Attestation Results.
Attestation Result	PoH Certificate and trust score	Contains only statistical exponents (alpha, beta, kappa) and aggregate scores. No raw Evidence (cell IDs, timestamps, chain hashes) is included in the Attestation Result.
Relying Party	Any service consuming PoH Certificates	Evaluates the Attestation Result against its own policy. Does not receive or process raw Evidence.

Table 8: TRIP-to-RATS Role Mapping

11.2. Evidence Flow

H3-quantized Evidence is transmitted from the Attester to the Verifier. This is an explicit design choice: the Verifier requires access to the full breadcrumb chain to compute PSD exponents, fit Levy flight parameters, and evaluate the Hamiltonian.

Privacy preservation derives from the H3 quantization transform applied by the Attester before any data leaves the device, NOT from data locality. Raw GPS coordinates MUST NOT be transmitted. The quantization transform is lossy and irreversible: given an H3 cell identifier, an observer can determine only that the Attester was within the cell’s area, not the precise coordinates.

The Verifier receives cell identifiers, timestamps, context digests, and chain hashes. The Verifier MUST NOT forward raw Evidence to Relying Parties. Only the Attestation Result (PoH Certificate) is disclosed to Relying Parties.

11.3. Verifier Trust Model

The Relying Party MUST trust the Verifier that produced the Attestation Result. This trust relationship is analogous to the trust a TLS client places in a Certificate Authority: the protocol defines the verification procedures, but the selection of trusted Verifiers is a deployment policy decision.

The TRIP protocol supports multiple independent Verifiers. An Attester MAY submit Evidence to more than one Verifier. A Relying Party MAY accept Attestation Results from any Verifier it trusts. The set of trusted Verifiers is configured by the Relying Party and is outside the scope of this specification.

Each Verifier MUST have its own Ed25519 key pair. The Verifier signs PoH Certificates with its private key (field 14 of the PoH Certificate). Relying Parties verify this signature against the Verifier’s published public key.

12. Replay Protection

TRIP provides replay protection at two distinct layers: protection of the Evidence chain against tampering, and protection of Attestation Results against replay to Relying Parties.

12.1. Chain-Level Replay Protection

The monotonically increasing index and the chaining via the previous block hash field provide replay protection within a single trajectory. A replayed breadcrumb will fail the chain integrity check. Cross-trajectory replay (injecting breadcrumbs signed by identity A into a chain belonging to identity B) will fail Ed25519 signature verification.

An attacker who obtains a copy of Evidence previously submitted to a Verifier cannot inject it into a different identity's chain because every breadcrumb is signed by the identity's private key and chained to the preceding breadcrumb via hash linkage.

12.2. Attestation Result Replay Protection

Chain-level protection ensures Evidence integrity but does not prevent an attacker from replaying a previously issued PoH Certificate to a Relying Party. An attacker who intercepts a valid PoH Certificate could present it to a different Relying Party, or present it after the identity's trust state has changed.

TRIP defines two verification modes to address this:

Passive Verification The Relying Party queries the Verifier for the current PoH Certificate associated with an identity. The Verifier returns its most recently computed Attestation Result. No freshness binding is provided. The Relying Party accepts the staleness risk inherent in cached results. This mode is suitable for low-stakes decisions where the cost of a replayed certificate is bounded (e.g., read access, rate limiting).

Active Verification The Relying Party provides an unpredictable nonce to the Verifier as part of a challenge-response exchange. The resulting PoH Certificate is cryptographically bound to the specific request, the current trajectory state, and the current moment. This mode is REQUIRED for high-stakes decisions (e.g., financial operations, handle claiming, publishing). See Section 12.3 for the full protocol.

12.3. Active Verification Protocol

The Active Verification Protocol provides cryptographic freshness guarantees by binding the Attestation Result to a Relying Party-supplied nonce, the current chain head, and the current time. The protocol proceeds as follows:

1. The Relying Party generates an unpredictable nonce (RECOMMENDED: 16 bytes from a cryptographically secure random number generator) and sends a Verification Request to the Verifier:

```
VerificationRequest = {
  0 => bstr .size 32,   ; identity public key
  1 => bstr .size 16,   ; nonce
  2 => uint,            ; request timestamp
  3 => uint,            ; requested freshness window (seconds)
}
```

2. The Verifier delivers a Liveness Challenge to the Attester via a real-time channel (e.g., WebSocket push, push notification):

```
LivenessChallenge = {
  0 => bstr .size 16,   ; nonce (from Relying Party)
  1 => bstr .size 32,   ; verifier identity (public key)
  2 => uint,            ; challenge timestamp
  3 => uint,            ; response deadline (seconds)
}
```

3. The Attester constructs and signs a Liveness Response binding the nonce to the current chain state:

```
LivenessResponse = {
  0 => bstr .size 16,   ; nonce echo
  1 => bstr .size 32,   ; chain_head_hash (hash of most
                        ;   recent breadcrumb)
  2 => uint,            ; response timestamp
  3 => uint,            ; current breadcrumb index
  4 => bstr .size 64,   ; Ed25519 signature over fields 0-3
                        ;   using identity private key
}
```

4. The Verifier validates the Liveness Response by checking all of the following:

   * The Ed25519 signature (field 4) is valid against the identity's public key over fields 0-3.

   * The nonce echo (field 0) matches the nonce from the original Verification Request.

   * The chain_head_hash (field 1) is consistent with the Verifier's stored trajectory state for this identity.

   * The response timestamp (field 2) is within the response deadline specified in the Liveness Challenge.

* The breadcrumb index (field 3) matches or exceeds the Verifier's last known index for this identity.

5. Upon successful validation, the Verifier produces a fresh PoH Certificate with field 12 set to the Relying Party's nonce and field 13 set to the chain_head_hash from the Liveness Response. The Verifier signs this certificate with its own Ed25519 key and returns it to the Relying Party.

6. The Relying Party verifies the PoH Certificate per Section 9, additionally confirming that field 12 matches its original nonce.

If the Attester does not respond within the response deadline, the Verifier MUST return an error to the Relying Party indicating that liveness could not be confirmed. The Verifier MUST NOT fall back to Passive mode when Active mode was explicitly requested.

12.4. Active Verification CDDL

The following CDDL [RFC8610] schema defines the Active Verification messages:

```
; Active Verification Protocol CDDL Schema

verification-request = {
  0 => bstr .size 32,        ; identity_key
  1 => bstr .size 16,        ; nonce
  2 => uint,                 ; request_timestamp
  3 => uint,                 ; freshness_window_seconds
}

liveness-challenge = {
  0 => bstr .size 16,        ; nonce
  1 => bstr .size 32,        ; verifier_key
  2 => uint,                 ; challenge_timestamp
  3 => uint,                 ; response_deadline_seconds
}

liveness-response = {
  0 => bstr .size 16,        ; nonce_echo
  1 => bstr .size 32,        ; chain_head_hash
  2 => uint,                 ; response_timestamp
  3 => uint,                 ; current_breadcrumb_index
  4 => bstr .size 64,        ; ed25519_signature
}

; The PoH Certificate (Section 9) with fields 12-13
; populated serves as the Attestation Result for
; Active Verification.
```

13. Security Considerations

13.1. GPS Replay Attacks

An adversary records a legitimate trajectory and replays the GPS coordinates on a different device. TRIP detects this through multiple channels:

* H_flock: the ambient flock of co-located entities has changed since the recording. The replayed trajectory will show misalignment with current human flow.

* H_contextual: unless the adversary also replays Wi-Fi BSSIDs, cellular tower IDs, and IMU data, the context digest will not match.

* H_structure: the timing regularity of a replay is typically either too perfect (exact timestamps) or shifted in a detectable pattern.

13.2. Synthetic Walk Generators

An adversary uses software to generate plausible-looking GPS coordinates. The Criticality Engine defeats this:

* PSD alpha test: random walk generators produce white noise (alpha approximately 0). Brownian motion generators produce alpha approximately 2. Neither falls in the biological [0.30, 0.80] range.

* Levy flight fitting: synthetic displacements rarely match the truncated power-law distribution with biologically plausible beta and kappa values.

* Predictability test: synthetic trajectories either show near-zero predictability (random) or near-perfect predictability (scripted), both outside the human [0.80, 0.95] range.

13.3. Emulator Injection

An adversary runs the TRIP client on an Android/iOS emulator with spoofed GPS. Detection relies on:

* H_contextual: emulators typically provide zero or synthetic IMU data that does not correlate with claimed GPS displacement.

* Context digest: emulators lack real Wi-Fi scan data and cellular tower IDs, producing empty or static context digests.

13.4. Device Strapping (Robot Dog Attack)

An adversary straps a phone to a mobile robot or drone. This is the most sophisticated attack because it produces real GPS, Wi-Fi, cellular, and IMU data from actual physical movement. Mitigation relies on:

* PSD alpha test: robotic movement typically lacks the characteristic 1/f noise of biological systems. Robots move with mechanical regularity (brown noise) or programmatic randomness (white noise).

* Phase-space smoothness (H_structure): a robot’s velocity-acceleration phase portrait differs characteristically from human movement.

* Circadian and weekly profiles: a robot generating breadcrumbs 24/7 will diverge from human activity patterns.

This attack remains an active area of research. The protocol's defense-in-depth approach through multiple independent Hamiltonian components makes it progressively more expensive to defeat all channels simultaneously.

13.5. Verifier Compromise

A compromised Verifier could issue fraudulent PoH Certificates. Mitigations include:

* Relying Parties SHOULD accept certificates from multiple independent Verifiers and cross-check results for high-stakes operations.

* Verifier key rotation and revocation procedures SHOULD be established as part of deployment policy.

* The Active Verification Protocol ensures that even a compromised Verifier cannot produce a valid certificate without the Attester's cooperation (the Liveness Response requires the identity's private key).

13.6. Denial of Service

An attacker can generate large numbers of Verification Requests or breadcrumb submissions to consume Verifier resources. Verifiers SHOULD rate-limit requests per identity and per Relying Party. The Active Verification Protocol's real-time requirement (Attester must respond within the deadline) provides an inherent rate limit on valid completions.

14. Privacy Considerations

14.1. Quantization-Based Privacy

TRIP's privacy model is based on lossy spatial quantization, not on data locality. H3-quantized Evidence (cell identifiers, timestamps, context digests, and chain hashes) is transmitted from the Attester to the Verifier. Raw GPS coordinates MUST NOT be transmitted and MUST NOT be stored by the Attester after quantization.

The H3 quantization transform is lossy and irreversible. Given an H3 cell identifier, an observer learns only that the Attester was within the cell's area at the stated time. At the default resolution 10, each cell covers approximately 15,000 m^2 (~0.015 km^2), providing meaningful ambiguity in populated areas.

H3 resolution is a configurable protocol parameter that controls the privacy-precision tradeoff. Deployments operating under strict privacy regulations (e.g., GDPR) MAY mandate lower resolutions. Users SHOULD be informed of the selected resolution and its privacy implications.

14.2. Verifier Data Handling

The Verifier receives and processes H3-quantized Evidence to compute trust scores and PoH Certificates. The Verifier:

* MUST NOT forward raw Evidence (breadcrumbs, cell identifiers, timestamps) to Relying Parties.

* MUST disclose its data retention policy to Attesters.

* SHOULD retain only the statistical aggregates (alpha, beta, kappa, transition matrices, circadian profiles) necessary for ongoing trust computation, and MAY discard individual breadcrumbs after incorporation into the aggregate model.

* MUST support data deletion requests where required by applicable law.

14.3. Relying Party Data Minimization

The PoH Certificate is designed for maximum data minimization. A Relying Party learns:

* The identity’s public key.

* Statistical exponents characterizing movement patterns (alpha, beta, kappa).

* Aggregate counts (epochs, breadcrumbs, unique cells).

* A trust score and criticality confidence.

A Relying Party does NOT learn: which cities, countries, or specific locations the identity has visited; the identity’s home or workplace locations; the identity’s daily schedule; or any raw trajectory data.

14.4. Trajectory Correlation and Sybil Resistance

A trajectory is intrinsically linkable: all breadcrumbs share the same identity key. This is by design, as trust accumulation requires identity continuity, and Proof-of-Humanity requires a durable binding between one physical entity and one cryptographic identity.

A single physical entity operating multiple TRIP identities simultaneously constitutes a Sybil attack. TRIP raises the cost of such attacks through multiple mechanisms: each identity requires a separate physical device, weeks of sustained movement, and independent trajectory accumulation. The H_flock component (Section 8.4) provides a detection mechanism: co-located trajectories with identical displacement vectors produce a correlated-movement signal that Verifiers MAY use to flag suspected Sybil identities.

Cross-trajectory correlation analysis (detecting that two identities may be operated by the same physical entity) is a Verifier-side heuristic and is outside the scope of this specification. However, Verifiers SHOULD implement such heuristics and MAY reduce trust scores for trajectories that exhibit sustained co-movement with other identities.

14.5. Population Density Considerations

In sparsely populated areas, even cell-level granularity may narrow identification to very few individuals. Implementations SHOULD use lower resolution (larger cells) in rural areas and MAY allow users to override to a lower resolution at any time.

15. Deployment Considerations

15.1. Multiple Verifier Deployments

The TRIP architecture supports multiple independent Verifier deployments. Any entity that implements the verification procedures defined in this specification (chain verification, PSD analysis, Levy flight fitting, Hamiltonian evaluation) MAY operate as a TRIP Verifier.

An Attester MAY submit Evidence to more than one Verifier simultaneously or sequentially. Each Verifier maintains its own independent view of the trajectory and produces its own Attestation Results.

A Relying Party MAY accept Attestation Results from any Verifier it trusts. The mechanism for establishing and managing Verifier trust (e.g., trust lists, certificate pinning, reputation systems) is a deployment policy decision outside the scope of this specification.

15.2. Verifier Interoperability

All conforming Verifiers MUST implement the base verification procedures: chain integrity (Section 3.3), PSD alpha classification (Section 6.1), and the PoH Certificate format (Section 9).

Verifiers MAY implement additional analysis beyond the base specification (e.g., machine-learning-based anomaly detection, additional Hamiltonian components). Such extensions MUST NOT alter the PoH Certificate format but MAY influence the trust score and criticality confidence values.

Two Verifiers processing the same Evidence SHOULD produce consistent alpha, beta, and kappa values (within numerical precision bounds) because these are derived from deterministic mathematical operations on the same input data. Trust scores MAY differ if Verifiers apply different weighting policies.

15.3. Transport Binding

This specification does not mandate a specific transport for Evidence submission, Liveness Challenges, or Verification Requests. Implementations MAY use HTTPS, WebSocket, CoAP, or any transport that provides confidentiality and integrity protection.

The Active Verification Protocol (Section 12.3) requires a real-time channel between the Verifier and Attester for Liveness Challenge delivery. Implementations SHOULD use persistent connections (e.g., WebSocket) or mobile push notification services to minimize latency.

15.4. Naming System Integration

TRIP operates on Ed25519 public keys and TITs. The binding of human-readable names (handles) to TRIP identities is explicitly outside the scope of this specification and is expected to be addressed in a companion document. TRIP provides the trust foundation (PoH Certificates) upon which naming systems can make registration and access-control decisions.

15.5. Accessibility and Low-Mobility Users

TRIP does not require geographic travel. It requires sustained physical existence over time, which is a fundamentally different property. This section clarifies how the protocol accommodates users with limited or no physical mobility, addressing concerns raised during review [ZHANG-REVIEW].

A person who remains in a single H3 cell for an extended period still generates a valid trajectory. The trust scoring formula (Section 10) assigns 20% weight to temporal continuity (days_since_first) and 40% weight to breadcrumb count, both of which accumulate regardless of spatial diversity. A homebound user who collects breadcrumbs over 365 days achieves 60% of the maximum trust score from time and volume alone.

The context digest (Section 2.2) provides environmental diversity even without movement. Wi-Fi access points in a neighborhood change over time as devices are added, removed, or replaced. Cellular tower assignments shift with network load. IMU sensors detect micro-movements (hand tremor, breathing, device repositioning) that produce non-trivial context variation. These ambient changes ensure that breadcrumbs collected from a fixed location are not identical, defeating simple replay attacks.

The Hamiltonian (Section 8) is self-calibrating per identity: the baseline H_baseline is the rolling median of the identity's own energy values. A user with limited mobility develops a low-energy baseline that reflects their actual movement pattern. Anomalies are detected relative to the individual's own profile, not against a global expectation of travel.

The PSD alpha exponent (Section 6.1) may require adaptation for stationary users. When the displacement time series has very low variance, the PSD analysis degrades. Implementations SHOULD supplement spatial PSD with temporal PSD (analyzing inter-breadcrumb timing patterns) for identities whose unique cell count is below a configurable threshold (default: 5 cells over the most recent 200 breadcrumbs). Circadian rhythms in collection timing exhibit the same 1/f characteristics as spatial displacement, providing an alternative criticality signal.

Deployments MUST NOT impose minimum spatial diversity requirements that would exclude users with mobility limitations. The trust scoring formula MAY be adjusted by deployment policy to increase the weight of temporal factors for identities that self-declare reduced mobility, provided that the overall security model remains sound.

16. IANA Considerations

This document has no IANA actions at this time. Future revisions may request:

* CBOR tag assignments for breadcrumb, epoch, PoH Certificate, and Active Verification message structures.

* A media type registration for application/trip+cbor.

* An entry in a TRIP Verification Mode registry (passive, active).

17. References

17.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/info/rfc9334>.

17.2. Informative References

[H3] Uber Technologies, "H3: Uber’s Hexagonal Hierarchical Spatial Index", 2023, <https://h3geo.org/>.

[PARISI-NOBEL] The Nobel Foundation, "Nobel Prize in Physics 2021: Giorgio Parisi", 2021, <https://www.nobelprize.org/prizes/physics/2021/parisi/facts/>.

[CAVAGNA-STARLINGS] Cavagna, A., Cimarelli, A., Giardina, I., Parisi, G., Santagati, R., Stefanini, F., and M. Viale, "Scale-free correlations in starling flocks", Proceedings of the National Academy of Sciences, 107(26), 11865-11870, DOI 10.1073/pnas.1005766107, 2010, <https://doi.org/10.1073/pnas.1005766107>.

[BALLERINI-TOPOLOGICAL] Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., Lecomte, V., Orlandi, A., Parisi, G., Procaccini, A., Viale, M., and V. Zdravkovic, "Interaction ruling animal collective behavior depends on

topological rather than metric distance", Proceedings of the National Academy of Sciences, 105(4), 1232-1237, DOI 10.1073/pnas.0711437105, 2008, <https://doi.org/10.1073/pnas.0711437105>.

[BARABASI-MOBILITY]
Gonzalez, M.C., Hidalgo, C.A., and A.-L. Barabasi, "Understanding individual human mobility patterns", Nature, 453, 779-782, DOI 10.1038/nature06958, 2008, <https://doi.org/10.1038/nature06958>.

[SONG-LIMITS]
Song, C., Qu, Z., Blumm, N., and A.-L. Barabasi, "Limits of Predictability in Human Mobility", Science, 327(5968), 1018-1021, DOI 10.1126/science.1177170, 2010, <https://doi.org/10.1126/science.1177170>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[ZHANG-REVIEW]
Zhang, J., "Review comments on draft-ayerbe-trip-protocol-01", IETF mailing list post, February 2026.

Acknowledgements

The TRIP protocol builds upon foundational work in cryptographic identity systems, geospatial indexing, statistical physics, and network science. The author thanks the contributors to the H3 geospatial system, the Ed25519 specification authors, and the broader IETF community for establishing the standards that TRIP builds upon. The Criticality Engine framework is inspired by the work of Giorgio Parisi on scale-free correlations in biological systems and Albert-Laszlo Barabasi on the fundamental limits of human mobility.

The author thanks Muhammad Usama Sardar for detailed review of -01 that identified the need for explicit RATS role mapping, attestation-result replay protection, and privacy model corrections. These contributions substantially improved the rigor of this specification. The author also thanks Jun Zhang for raising critical questions about accessibility and the applicability of mobility models to users with limited physical mobility, leading to the Accessibility and Low-Mobility Users section.

Author’s Address

Camilo Ayerbe Posada
ULISSY s.r.l.
Via Gaetano Sacchi 16
00153 Roma RM
Italy
Email: cayerbe@gmail.com