

Individual Submission
Internet-Draft
Intended status: Experimental
Expires: 22 August 2026

D. Condrey
WritersLogic
18 February 2026

Proof of Process (PoP): Architecture and Evidence Format
draft-condrey-rats-pop-protocol-06

Abstract

This document specifies the Proof of Process (PoP) Evidence Framework, a specialized profile of Remote Attestation Procedures (RATS) designed to validate the provenance of effort in digital authorship. Unlike traditional provenance, which tracks file custody, PoP attests to the continuous physical process of creation.

The protocol defines a cryptographic mechanism for generating Evidence Packets utilizing a composite Sequential Work Function (SWF) based on Proof of Biological Space-Time (PoBST) to enforce temporal monotonicity and Cross-Domain Constraint Entanglement (CDCE) to bind behavioral entropy (human jitter) and physical state to the document. Technical specifications for wire formats, sequential work functions, and hardware-anchored trust are provided.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at <https://github.com/writerslogic/draft-condrey-rats-pop>.

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/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 August 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	3
2. Requirements Language	4
3. System Model	4
3.1. RATS Entity Roles	4
3.2. Compatibility with RATS Architecture	5
3.3. Applicability to RATS Architecture	6
4. Protocol Overview	6
4.1. Passport Model Message Flow	6
4.2. Evidence Lifecycle	8
5. Threat Model	10
5.1. Adversarial Attester Model	10
5.2. Security Goals	11
5.3. Attack Taxonomy	11
5.3.1. Retype Attack	11
5.3.2. Replay Attack	12
5.3.3. Relay Attack	12
5.3.4. SWF Acceleration Attack	12
5.3.5. AE Spoofing	12
5.3.6. Diversion Attack	13
5.4. Out-of-Scope Threats	14
6. Core Principles and Claims	14
7. Protocol Rationale and Terminology	14
8. Attester State Machine	15
9. Evidence Content Tiers	15
10. Attestation Assurance Levels	16
10.1. Tier T1: Software-Only	16
10.2. Tier T2: Attested Software	17
10.3. Tier T3: Hardware-Bound	17
10.4. Tier T4: Hardware-Hardened	17
11. Profile Architecture	17
11.1. Conformance Requirements	18
12. Evidence Format and CDDL	18
12.1. Checkpoint Hash Computation	23
12.2. Checkpoint Computation Order	24
12.3. Evidence Protection	24

13. Sequential Work Function	25
13.1. Construction	25
13.2. Verification Protocol	26
13.3. Fiat-Shamir Sample Derivation	26
13.4. SWF Seed Derivation	27
13.5. Merkle Tree Construction	27
13.6. Mandatory SWF Parameters	28
13.7. Entangled MAC Computation	28
13.8. Jitter Seal Computation	29
13.9. Security Bound	29
13.10. Hardware-Anchored Time (HAT)	30
14. IANA Considerations	30
14.1. CBOR Tags	30
14.2. SMI Private Enterprise Number	31
14.3. EAT Profile	31
14.4. Media Types	31
14.5. TLS Exporter Label	32
15. Security Considerations	32
15.1. Primary Threat: Adversarial Attester	32
15.2. Retype Attack Defenses	33
15.3. Relay and Replay Attack Defenses	33
15.4. SWF Acceleration Defenses	34
15.5. Trust Gradation by Tier	34
15.6. Forgery Cost Bounds	34
15.7. Denial of Service	35
15.8. MAC Field Security Limitations	35
15.9. Physical Freshness by Tier	36
15.10. Implementation Security Requirements	36
16. Privacy Considerations	36
16.1. Data Minimization	36
16.2. Behavioral Fingerprinting	37
16.3. Physical State Leakage	37
16.4. Unlinkability	37
17. References	37
17.1. Normative References	37
17.2. Informative References	39
SWF Test Vectors	40
Acknowledgements	41
Author's Address	41

1. Introduction

The rapid proliferation of generative artificial intelligence has created an authenticity crisis in digital discourse. While traditional provenance tracks the "custody of pixels," it fails to attest to the human-driven process of creation. This document specifies the Proof of Process (PoP) protocol, which extends the RATS architecture [RFC9334] to validate the "provenance of effort."

Unlike traditional attestation which captures static system state, PoP attests to a continuous physical process. It introduces Proof of Biological Space-Time (PoBST) to enforce temporal monotonicity and Cross-Domain Constraint Entanglement (CDCE) to bind behavioral entropy (human jitter) and physical state (thermodynamics) to the document's evolution.

By entangling content hashes with these physical constraints, this protocol enables an Attester to generate an Evidence Packet (.pop) that imposes quantifiable cost on forgery of authorship claims, preserving privacy by design without disclosing document content.

2. 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.

3. System Model

This section defines the PoP system model in terms of the RATS architecture [RFC9334] and identifies where PoP diverges from standard remote attestation assumptions.

3.1. RATS Entity Roles

PoP maps to RATS entity roles as follows:

Attester: The authoring application and its host platform. The Attester generates Evidence Packets (.pop) containing behavioral entropy, physical state markers, and SWF proofs. Unlike traditional RATS deployments, the Attester in PoP is operated by the entity whose claims are being verified (the author).

Attesting Environment (AE): The software and hardware components that collect telemetry and generate cryptographic bindings. This includes the authoring application, operating system interfaces for entropy collection, and hardware Secure Elements (TPM/SE) when available.

Verifier: An entity that appraises Evidence Packets and produces Attestation Results. Verifiers may be operated by publishers, platforms, or independent third parties. Verifier logic is specified in [PoP-Appraisal].

Relying Party: Consumers of Attestation Results who make trust

decisions based on the appraisal. This includes publishers, readers, or automated systems that need authenticity assurance.

Endorser: Entities that vouch for the Attesting Environment's integrity by issuing Endorsements. In PoP, Endorsers include hardware manufacturers that issue TPM endorsement certificates and platform attestation credentials for T3/T4 tiers.

Reference Value Provider: Entities that supply Reference Values for appraisal. In PoP, this includes the PoP specification itself (defining expected behavioral patterns and SWF parameters) and calibration services that provide updated forensic baselines.

3.2. Compatibility with RATS Architecture

PoP implements a specialized RATS profile with a critical trust inversion: in traditional remote attestation, the Attester is a device whose owner (Relying Party) wants assurance about its state. The adversary is typically external -- malware, network attackers, or supply chain threats.

In PoP, the Attester is operated by the author, and the Relying Party (publisher, reader) has no privileged access to the authoring environment. The primary adversary is the Attester operator themselves. This fundamental inversion shapes the entire security model:

- * Evidence must be unforgeable by the entity generating it
- * Temporal claims must be bound to physical constraints the Attester cannot circumvent
- * Behavioral entropy must be computationally expensive to simulate
- * Hardware attestation provides value only when the hardware root of trust is genuinely inaccessible to the Attester operator

Despite this inversion, PoP maintains compatibility with RATS message flows and data formats, enabling integration with existing RATS infrastructure where appropriate.

3.3. Applicability to RATS Architecture

PoP extends the RATS framework beyond traditional device state attestation to process attestation — verifying that a physical process (human authorship) occurred as claimed. This extension is justified because the fundamental problem structure is identical: an Attester generates Evidence, conveys it to a Verifier, and the Verifier produces Attestation Results for Relying Parties. The RATS entity roles, message flows, and data format conventions apply directly.

The adversarial Attester model (see Section 5.1) inverts the standard RATS trust assumption. The RATS architecture accommodates this through its layered trust model and configurable Appraisal Policies ([RFC9334], Section 8). The Experimental category is appropriate for exploring this novel application of RATS.

4. Protocol Overview

This section provides an end-to-end overview of the PoP protocol, mapping the message flow to the RATS passport model and illustrating the lifecycle of an Evidence Packet from creation through appraisal.

4.1. Passport Model Message Flow

PoP follows the RATS passport model ([RFC9334], Section 8.1; [RATS-Models]) in which Evidence flows directly from the Attester to the Verifier, and Attestation Results flow from the Verifier to the Relying Party:

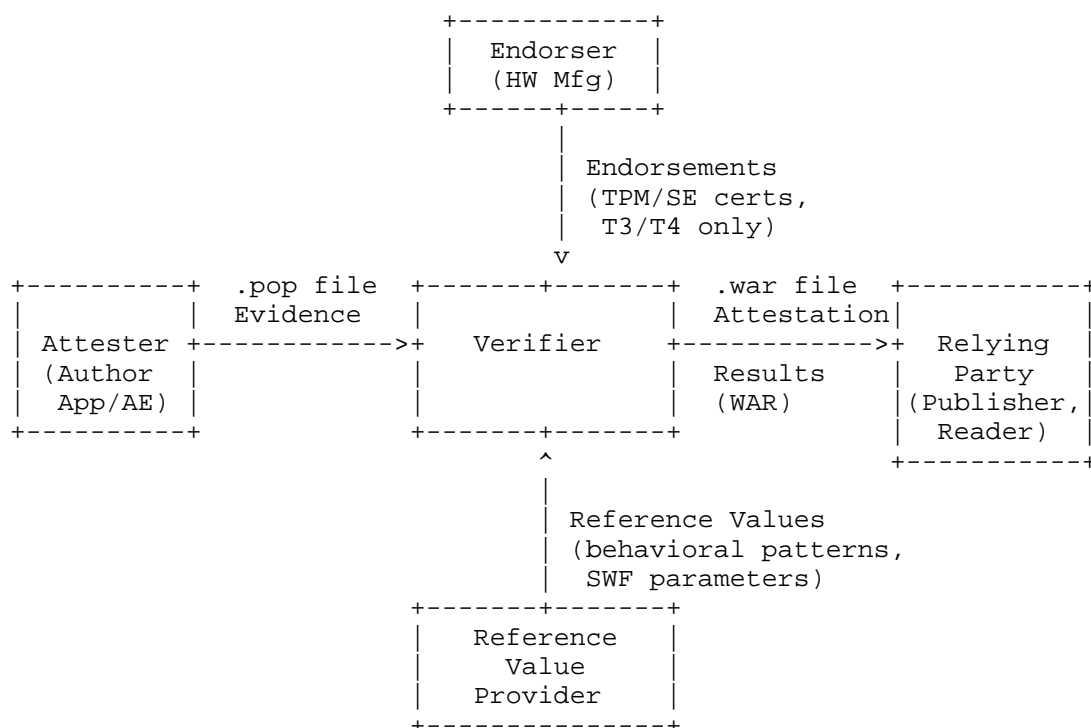


Figure 1: PoP Passport Model Message Flow

In this model:

1. The Attester (authoring application running in the Attesting Environment) collects behavioral telemetry during content creation and generates an Evidence Packet (.pop) containing SWF proofs, jitter bindings, and physical state markers.
2. The Evidence Packet is conveyed to a Verifier, which appraises chain integrity, temporal ordering, behavioral entropy, and content binding per the procedures defined in [PoP-Appraisal].
3. The Verifier produces a Writers Authenticity Report (.war) containing EAT claims, forensic assessment scores, and forgery cost estimates.
4. The Relying Party (publisher, reader, or automated platform) consumes the WAR to make trust decisions about the claimed authorship provenance.

Endorsers (hardware manufacturers) supply TPM endorsement certificates and Secure Element attestations that Verifiers use to validate hardware-bound claims in T3/T4 Evidence. Reference Value Providers supply the expected behavioral patterns, SWF difficulty parameters, and profile specifications that Verifiers use as appraisal baselines.

4.2. Evidence Lifecycle

The following sequence illustrates the end-to-end lifecycle of a PoP attestation session:

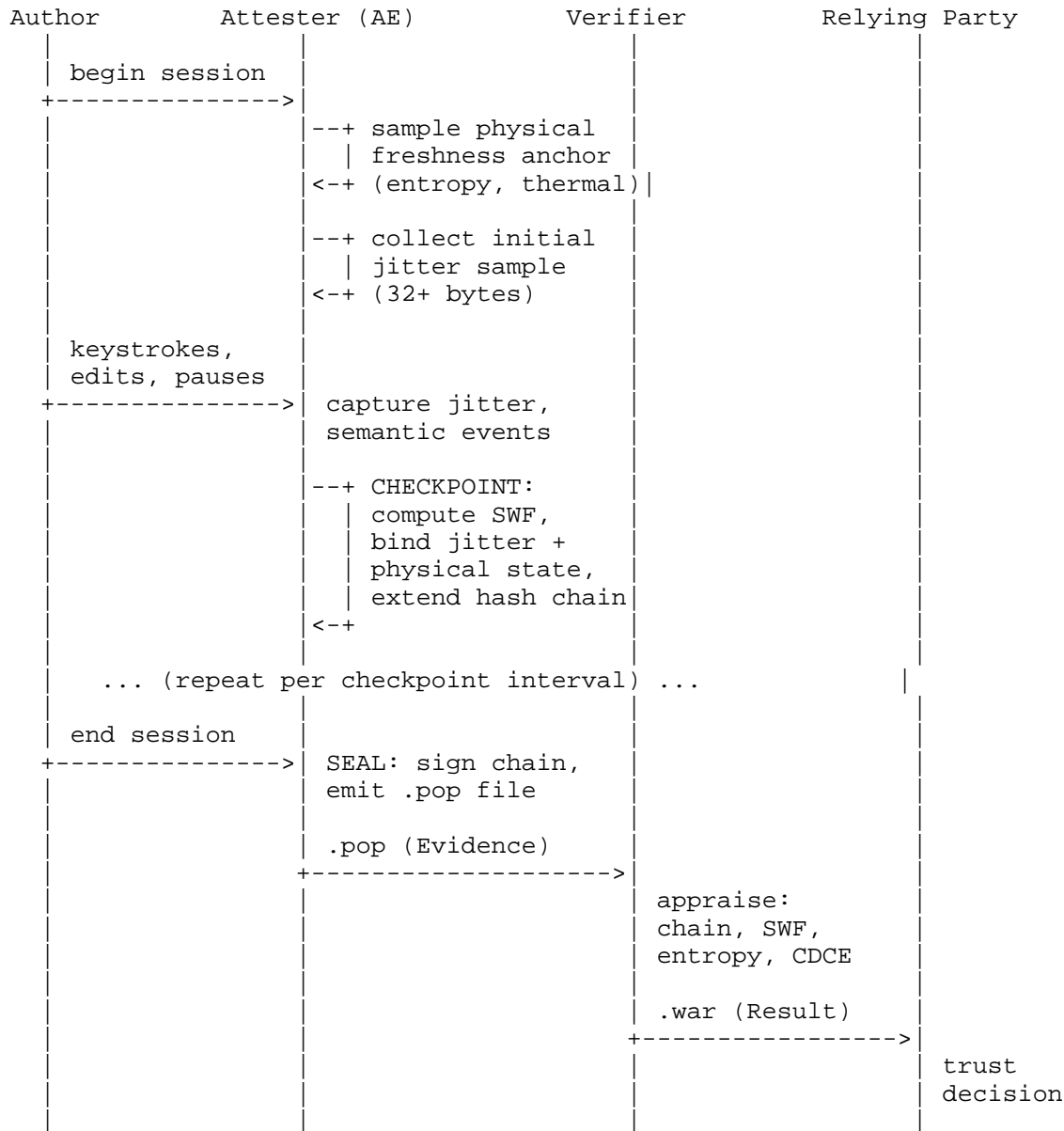


Figure 2: Evidence Packet Lifecycle

Each checkpoint interval (default 30 seconds, MUST be between 10 and 120 seconds) produces one link in the hash chain. The SWF computation runs continuously during the interval, binding the author's behavioral entropy and the platform's physical state to the

elapsed wall-clock time. At session end, the Attester seals the complete chain into a .pop Evidence Packet for conveyance to the Verifier.

5. Threat Model

This section defines the adversary model following the methodology of [RFC3552] and incorporating insights from RATS security analysis [Sardar-RATS]. The threat model assumes a Dolev-Yao style adversary [Dolev-Yao] with domain-specific constraints.

5.1. Adversarial Attester Model

The PRIMARY threat in PoP is an adversarial Attester -- an author who controls the Attesting Environment and seeks to generate Evidence for content they did not authentically author. This inverts the standard RATS trust assumption where the Attester is trusted to report honestly.

The adversarial Attester has the following capabilities:

- * `_Full software control:` Can modify, instrument, or replace any software component including the authoring application and operating system
- * `_Timing manipulation:` Can adjust system clocks, virtualize execution environments, and attempt to compress or expand apparent time
- * `_Entropy injection:` Can inject synthetic behavioral data (keystroke timing, jitter sequences) from pre-recorded or generated sources
- * `_Content pre-generation:` Can generate document content using AI tools or other assistance before initiating the attestation session
- * `_Parallel execution:` Can run multiple attestation sessions simultaneously or use distributed resources

The adversary is constrained by:

- * `_Physics:` Cannot violate thermodynamic laws or accelerate hardware beyond physical limits
- * `_Memory bandwidth:` MHSF computations are bounded by available memory bandwidth

- * `_Hardware isolation:_` In T3/T4 tiers, cannot extract keys from Secure Elements without physical tampering
- * `_Economic rationality:_` Will not expend resources exceeding the value of successful forgery

5.2. Security Goals

PoP provides the following authentication properties, defined in terms of adversary advantage:

Temporal Authenticity: Given Evidence claiming authorship duration D , an adversary cannot produce valid Evidence in time significantly less than D . Formally: $\text{Adv_temporal} = \Pr[\text{Verify}(E) = \text{accept AND Time}(\text{Generate}(E)) < D - \text{epsilon}]$ is negligible for meaningful epsilon.

Behavioral Authenticity: Given Evidence containing behavioral entropy B , an adversary cannot efficiently generate synthetic entropy that is indistinguishable from biological origin. The cost of generating synthetic behavioral data satisfying all forensic constraints MUST exceed a defined threshold.

Content Binding: Evidence E is cryptographically bound to document D such that E cannot be repurposed to attest a different document D' . This property is unconditional given collision resistance of SHA-256.

Non-repudiation (T3/T4): In hardware-bound tiers, Evidence is signed with keys that the Attester cannot extract or duplicate, providing non-repudiation of the attestation act.

5.3. Attack Taxonomy

The following attacks are in scope for PoP defenses:

5.3.1. Retype Attack

The canonical forgery attack against PoP: an adversary generates content using AI or other assistance, then retypes the pre-existing content while collecting "authentic" behavioral telemetry. This attack exploits the gap between typing existing text and composing original text.

PoP defends against retype attacks through:

- * Cognitive Load Correlation: Authentic composition exhibits increased inter-keystroke intervals during high-complexity passages. Retyping known text shows uniform timing regardless of content complexity. Evidence with semantic-timing correlation $r < 0.2$ is flagged for additional scrutiny (see Section 15.2).
- * Error Topology: Authentic authoring exhibits characteristic error patterns (hesitations, deletions near recent insertions, self-corrections). Retyping from reference exhibits either unnaturally low error rates or artificially injected errors lacking positional correlation.
- * Semantic-Temporal Binding: The SWF proof binds the document's semantic evolution to wall-clock time. Retyping requires real-time effort proportional to document length, even if content was pre-generated.

Retype attacks remain economically viable for short documents. The forgery cost scales with document length and checkpoint frequency, providing graduated assurance rather than binary security.

5.3.2. Replay Attack

Attempting to reuse previously valid Evidence for new claims. Defeated by Physical Freshness anchors that bind Evidence to non-reproducible physical state (thermal trajectories, kernel entropy samples).

5.3.3. Relay Attack

Forwarding challenges or Evidence between a legitimate author and an adversary's session. In PoP, this manifests as claiming credit for another author's work. Defeated by hardware-bound signing (T3/T4) and out-of-band presence challenges that verify physical proximity.

5.3.4. SWF Acceleration Attack

Using specialized hardware to compute SWF proofs faster than consumer hardware. Mitigated by Argon2id's memory-hardness (computation bounded by memory bandwidth, not ALU throughput) and Hardware-Anchored Time in T3/T4 tiers.

5.3.5. AE Spoofing

Presenting a virtualized or modified Attesting Environment as genuine. In T1/T2 tiers, this is possible and Evidence should be weighted accordingly. T3/T4 tiers require hardware attestation that is difficult to spoof without physical access to the Secure Element.

5.3.6. Diversion Attack

An adversary redirects Evidence intended for one Verifier to a different Verifier or Relying Party context. PoP Evidence Packets do not inherently bind to a specific Verifier identity. To mitigate this, implementations SHOULD use the TLS Exported Keying Material (EKM) mechanism defined in [RFC9266] to bind Evidence to the transport session.

When the Attester conveys Evidence over TLS, it SHOULD populate the optional channel-binding field (key 11) in the evidence-packet structure as follows:

1. The Attester calls the TLS exporter function with label "EXPORTER-PoP-channel-binding" and an empty context value, requesting 32 bytes of output.
2. The Attester sets binding-type to 1 (tls-exporter) and binding-value to the 32-byte EKM output.
3. The Verifier, upon receiving the Evidence Packet, calls the same TLS exporter function on its side of the connection and compares the result to the binding-value in the channel-binding field.
4. If the values do not match, the Verifier MUST reject the Evidence Packet as potentially diverted.

The EKM label "EXPORTER-PoP-channel-binding" is specific to this protocol. The empty context ensures the binding depends solely on the TLS session keys, which are unique per connection.

For offline verification (where no TLS session exists between Attester and Verifier), the channel-binding field is absent and Relying Parties MUST evaluate Evidence provenance through out-of-band channels.

When PoP Evidence is conveyed over an attested TLS channel, implementations MAY leverage the SEAT exported authenticator mechanism [SEAT-EXPAT] to combine platform attestation (proving the Attesting Environment's integrity) with PoP process attestation (proving the authorship process). The TLS channel binding described above is compatible with the SEAT evidence binding approach, which derives binding values from TLS exporters. At T3/T4 tiers, SEAT platform attestation provides the hardware trust anchor that corroborates PoP's Attesting Environment claims.

5.4. Out-of-Scope Threats

The following threats are explicitly out of scope:

- * Nation-state HSM compromise: Adversaries capable of extracting keys from certified HSMs via invasive physical attacks
- * Physics-level laboratory spoofing: Adversaries capable of simulating thermal trajectories and entropy sources at sub-microsecond precision
- * Quantum computation: Attacks requiring large-scale quantum computers (SHA-256 collision, Argon2id inversion)

6. Core Principles and Claims

Building on the threat model defined above, PoP operates on five primary constraints:

- * **Physics-based Cost:** Memory-Hard Sequential Functions (MHSF) establish an economic lower bound on forgery, ensuring consumer hardware remains competitive with specialized ASICs.
- * **Physical Freshness:** Replay and simulation attacks are defeated by anchoring sessions to irreversible physical markers (Thermal Trajectories and Kernel Entropy pools). Every session incorporates Non-deterministic Physical Freshness sampled within the AE at the start of the sequential work function execution.
- * **Biological Binding:** Captured human motor-signal randomness (jitter) serves as the non-deterministic seed for the spacetime proof.
- * **Out-of-Band Presence:** Utilizing secondary physical devices (e.g., smartphone QR scans) to bridge the digital-physical gap and ensure a human is in the loop.
- * **Asymmetric Verification:** The sequential work function allows proofs to be verified probabilistically via Merkle-sampled audit proofs, ensuring scalability and DoS resistance.

7. Protocol Rationale and Terminology

The Proof of Process (PoP) framework follows the RATS architecture while introducing domain-specific extensions for physical process attestation.

PoP Evidence Packet (.pop): An Attester artifact containing Merkle

trees, PoBST traces, and physical liveness markers (CBOR tag 1347571280, encoding ASCII "POP ").

WAR Result (.war): A Verifier Attestation Result containing signed EAT claims and forensic assessments (CBOR tag 1463894560). The WAR format is specified in [PoP-Appraisal].

PoBST: Proof of Biological Space-Time. A memory-hard sequential function with probabilistic verification, entangled with human jitter.

CDCE: Cross-Domain Constraint Entanglement. The method of weaving jitter and thermodynamics into the cryptographic chain.

SWF: Sequential Work Function. The composite construction combining Argon2id and iterated SHA-256 (see Section 13).

8. Attester State Machine

The Attesting Environment (AE) MUST implement the following formal state machine:

- * RECORDING: AE captures semantic events and physical telemetry into a hash-linked buffer. Events are appended and the block hash is updated.
- * PENDING_CHECK: The current event block is frozen to prepare for a checkpoint. No new events are accepted into this block.
- * CHECKPOINT: AE computes the SWF over the entangled seed (previous hash + current jitter + physical markers).
- * SEALING: The Attester generates a final snapshot, signs the transcript root with the Attester's signing key (hardware-bound for T3/T4; software-managed for T1/T2), and prepares the transport container (.pop).

9. Evidence Content Tiers

PoP Evidence Packets are classified by the depth of behavioral and forensic data collected:

CORE (Tier Value 1): Checkpoint chain with PoBST proofs, SHA-256 content binding, and physical freshness anchors. Proves temporal ordering and content integrity.

ENHANCED (Tier Value 2): All CORE components plus behavioral entropy

capture (Jitter Seals) and intra-checkpoint correlation. Adds evidence of interactive authoring behavior.

MAXIMUM (Tier Value 3): All ENHANCED components plus CDCE, error topology analysis, and forgery cost bounds. Provides the strongest available evidence.

PoP Evidence is classified along two orthogonal axes. Evidence Content Tier (CORE/ENHANCED/MAXIMUM) determines the depth of behavioral and forensic data collected. Attestation Assurance Level (T1-T4) determines the strength of hardware trust anchoring. These axes are independent: a T3 CORE packet provides hardware-bound signing with minimal behavioral data, while a T1 MAXIMUM packet provides rich behavioral data with software-only signing.

10. Attestation Assurance Levels

The attestation tier system maps to established assurance frameworks including NIST SP 800-63B Authenticator Assurance Levels (AAL), ISO/IEC 29115 Levels of Assurance (LoA), and Entity Attestation Token (EAT) security levels as defined in [RFC9711].

PoP Tier	NIST AAL	EAT Security Level (RFC 9711)
T1: Software-Only	AAL1	unrestricted (1)
T2: Attested Software	AAL2	restricted (2)
T3: Hardware-Bound	AAL3	hardware (4)
T4: Hardware-Hardened	LoA4	hardware (4)

Table 1

T3 and T4 both map to EAT security level "hardware" (4) because the EAT specification does not distinguish PUF-level binding from standard TPM key binding.

10.1. Tier T1: Software-Only

Binding Strength: none or hmac_local

NIST AAL Mapping: AAL1

Security Properties:

- * SWF timing provides temporal ordering

- * Hash chains provide tamper evidence
- * Jitter entropy provides behavioral binding
- * No hardware root of trust; keys stored in software

10.2. Tier T2: Attested Software

T2 extends T1 with optional hardware attestation hooks. The AE attempts to use platform security features (Keychain, DeviceCheck) but degrades gracefully. Maps to AAL2.

10.3. Tier T3: Hardware-Bound

Requires TPM 2.0 or platform Secure Enclave key binding. Evidence generation MUST fail if hardware is unavailable. Maps to AAL3.

10.4. Tier T4: Hardware-Hardened

Discrete TPM + PUF binding + Enclave execution. Anti-tamper evidence required. Exceeds AAL3 requirements; maps to ISO/IEC 29115 LoA4.

11. Profile Architecture

The PoP specification defines three implementation profiles that establish Mandatory-to-Implement (MTI) requirements for interoperability.

Feature ID	Feature Name	CORE	ENHANCED	MAXIMUM
1	swf-argon2id-sha256	M	M	M
2	content-binding	M	M	M
4	checkpoint-chain	M	M	M
50	behavioral-entropy	O	M	M
60	assistive-mode	O	O	O
105	hardware-attestation	O	O	M

Table 2

Feature IDs 1-9 are reserved for core protocol features. IDs 50-99 are reserved for behavioral features. IDs 100-199 are reserved for hardware features. Future revisions may define additional features within these ranges.

11.1. Conformance Requirements

A conforming Attester MUST implement at least the CORE profile. A conforming Verifier MUST be capable of validating all three profiles. Verifiers encountering unknown fields MUST ignore them and proceed with validation of known fields. Verifiers receiving an Evidence Packet with version greater than 1 MUST reject the packet unless they implement the corresponding protocol version.

The profile-uri field in an Evidence Packet MUST be set to "urn:ietf:params:rats:eat:profile:pop:1.0" for Evidence conforming to this specification.

In the document-ref structure, byte-length is the length in bytes of the UTF-8 encoded document, and char-count is the number of Unicode scalar values (code points).

If the Evidence Packet omits the attestation-tier field, the Verifier MUST assess the tier from the evidence content: T1 if no hardware attestation is present, T2 if platform attestation hooks are detected, T3 if TPM key binding is verified, T4 if anti-tamper evidence and PUF binding are confirmed.

12. Evidence Format and CDDL

Evidence Packets are CBOR-encoded [RFC8949] and identified by semantic tag 1347571280. The CDDL notation [RFC8610] is used to define the wire format.

```
; CBOR tag wrappers
pop-evidence = #6.1347571280(evidence-packet)
pop-war = #6.1463894560(attestation-result)

; Primary structures
evidence-packet = {
  1 => uint,           ; version (MUST be 1)
  2 => tstr,           ; profile-uri
  3 => uuid,           ; packet-id
  4 => pop-timestamp,  ; created
  5 => document-ref,   ; document
  6 => [3* checkpoint], ; checkpoints (min 3)
  ? 7 => attestation-tier, ; T1-T4
  ? 8 => [* tstr],     ; limitations
```

```

    ? 9 => profile-declaration,      ; profile
    ? 10 => [+ presence-challenge], ; QR/OOB proofs
    ? 11 => channel-binding,          ; TLS EKM binding
    ; keys 14-17 reserved for future use
    ? 13 => content-tier,              ; Evidence Content Tier
    ? 18 => physical-liveness,         ; physical-liveness markers
    * int => any,                      ; extension fields
}

checkpoint = {
    1 => uint,                        ; sequence (monotonic)
    2 => uuid,                        ; checkpoint-id
    3 => pop-timestamp,               ; timestamp (local)
    4 => hash-value,                  ; content-hash
    5 => uint,                        ; char-count
    6 => edit-delta,                  ; delta
    7 => hash-value,                  ; prev-hash
    8 => hash-value,                  ; checkpoint-hash
    9 => process-proof,               ; SWF proof
    ? 10 => jitter-binding,            ; behavioral-entropy (ENHANCED+)
    ? 11 => physical-state,            ; CDCE Weave (ENHANCED+)
    ? 12 => bstr .size 32,            ; entangled-mac (ENHANCED+)
    * int => any,                      ; extension fields
}

document-ref = {
    1 => hash-value,                  ; content-hash
    ? 2 => tstr,                      ; filename
    3 => uint,                        ; byte-length
    4 => uint,                        ; char-count
    ? 5 => hash-salt-mode,             ; salting mode
    ? 6 => bstr .size 32,              ; salt-commitment
}

process-proof = {
    1 => proof-algorithm,              ; algorithm id
    2 => proof-params,                 ; SWF params
    3 => bstr .size 32,                ; input (seed)
    4 => bstr .size 32,                ; merkle-root
    5 => [+ merkle-proof],             ; sampled proofs
    6 => float32,                     ; claimed-duration (seconds)
}

; Subsidiary type definitions
attestation-tier = &(
    software-only: 1,                 ; T1: AAL1
    attested-software: 2,             ; T2: AAL2
    hardware-bound: 3,                ; T3: AAL3

```

```
    hardware-hardened: 4,           ; T4: LoA4
)

content-tier = &(
    core: 1,
    enhanced: 2,
    maximum: 3,
)

proof-algorithm = &(
    ; 1 is reserved for future use
    pobst-argon2id: 20,
)

hash-salt-mode = &(
    unsalted: 0,
    author-salted: 1,
)

proof-params = {
    1 => uint,           ; time-cost (t)
    2 => uint,           ; memory-cost (m, KiB)
    3 => uint,           ; parallelism (p)
    4 => uint,           ; iterations
}

jitter-binding = {
    1 => [+ float32],    ; intervals (ms)
    2 => float32,        ; entropy-estimate (bits)
    3 => bstr .size 32,  ; jitter-seal (HMAC)
}

merkle-proof = {
    1 => uint,           ; leaf-index
    2 => [+ bstr .size 32], ; sibling-path
    3 => bstr .size 32,  ; leaf-value
}

edit-delta = {
    1 => uint,           ; chars-added
    2 => uint,           ; chars-deleted
    3 => uint,           ; op-count
    ? 4 => [* edit-position], ; positions
}

edit-position = [
    uint,               ; offset
    int,                ; change (+/-), MUST be non-zero
]
```

```
]

physical-state = {
    1 => [+ float32],           ; thermal (relative)
    2 => int,                   ; entropy-delta (signed)
    ? 3 => bstr .size 32,       ; kernel-commitment
}

physical-liveness = {
    1 => [+ thermal-sample],    ; thermal trajectory
    2 => bstr .size 32,         ; entropy-anchor
}

thermal-sample = [
    pop-timestamp,             ; sample time
    float32,                   ; temperature delta
]

presence-challenge = {
    1 => bstr .size (16..256),   ; challenge-nonce (128+ bits)
    2 => bstr,                   ; device-signature (MUST be COSE_Sign1)
    3 => pop-timestamp,         ; response-time
}

profile-declaration = {
    1 => tstr,                   ; profile-id
    2 => [+ uint],              ; feature-flags
}

binding-type = &(
    tls-exporter: 1,
)

channel-binding = {
    1 => binding-type,          ; binding-type
    2 => bstr .size 32,         ; binding-value (EKM output)
}

; Base types
uuid = bstr .size 16
pop-timestamp = #6.1(float32)   ; CBOR tag 1 (epoch-based, float32)
hash-value = {
    1 => hash-algorithm,
    2 => bstr,
}
hash-algorithm = &(
    sha256: 1,
    sha384: 2,
```

```
    sha512: 3,  
)
```

The attestation-result type used in the pop-war tag wrapper is defined in [PoP-Appraisal].

All floating-point fields in this specification MUST be encoded using 32-bit IEEE 754 binary32 format, regardless of whether a smaller encoding would suffice. This ensures deterministic encoding.

pop-timestamp values MUST use floating-point encoding with at least millisecond precision. Integer encoding (second granularity) MUST NOT be used. pop-timestamp values MUST be positive (greater than zero). Verifiers MUST reject Evidence containing negative or zero timestamps.

When hash-salt-mode is author-salted (1), the author generates a random salt of at least 16 bytes. The salt-commitment field MUST contain SHA-256(salt). To verify content binding, the author discloses the salt to the Verifier, which checks that SHA-256(disclosed_salt) matches the salt-commitment. The salt-commitment field MUST be constrained to 32 bytes (.size 32).

SHA-256 (value 1) is mandatory-to-implement. Conforming Attesters and Verifiers MUST support SHA-256. Support for SHA-384 and SHA-512 is OPTIONAL.

The hash digest length MUST match the algorithm output length: 32 bytes for SHA-256, 48 bytes for SHA-384, and 64 bytes for SHA-512.

All hash-value fields within a single Evidence Packet MUST use the same hash algorithm. Verifiers MUST reject Evidence Packets containing mixed hash algorithms.

Encoders MUST NOT use CBOR preferred float serialization (which may encode values like 0.0 as float16) for PoP fields. All floating-point values MUST be encoded as 4-byte IEEE 754 binary32 (CBOR major type 7, additional info 26) regardless of the value.

Extension keys in evidence-packet and checkpoint structures MUST use integer values 100 or greater. Keys 0-99 are reserved for use by this specification and future revisions.

The op-count field in edit-delta counts the number of discrete editing operations (insertions, deletions, and replacements) during the checkpoint interval. A single paste operation counts as one operation regardless of character count.

In edit-position entries, the change value MUST be non-zero. Positive values indicate insertion of characters at the offset; negative values indicate deletion. A zero change value is semantically meaningless and MUST NOT appear.

The device-signature in a presence-challenge MUST be a COSE_Sign1 structure [RFC9052] covering the challenge-nonce. The signing key MUST be hardware-bound on the secondary device. The Verifier obtains the corresponding public key through prior device registration (the registration mechanism is out of scope for this document).

Per-checkpoint physical-state (checkpoint key 11) captures instantaneous thermal and entropy measurements. Packet-level physical-liveness (evidence-packet key 18) provides a session-wide thermal trajectory for replay detection. physical-liveness SHOULD be included in ENHANCED and MAXIMUM profiles. When both are present, Verifiers MUST verify that per-checkpoint thermal values are consistent with the session-wide trajectory.

12.1. Checkpoint Hash Computation

The checkpoint-hash field MUST be computed as follows:

```
checkpoint-hash = SHA-256(  
    prev-hash ||  
    content-hash ||  
    CBOR-encode(edit-delta) ||  
    CBOR-encode(jitter-binding) ||  
    CBOR-encode(physical-state) ||  
    process-proof.merkle-root  
)
```

Where || denotes concatenation and CBOR-encode produces deterministic CBOR per Section 4.2.1 of [RFC8949].

For the first checkpoint in a chain (sequence = 1), prev-hash MUST be set to SHA-256(CBOR-encode(document-ref)). This anchors the chain to the document identity.

When jitter-binding and physical-state fields are absent (CORE profile), the checkpoint-hash MUST be computed without those terms:
checkpoint-hash = SHA-256(prev-hash || content-hash || CBOR-encode(edit-delta) || process-proof.merkle-root).

All components except process-proof.merkle-root are either fixed-length hashes (32/48/64 bytes per algorithm) or CBOR-encoded (self-delimiting). The merkle-root (32 bytes, fixed length) is appended last. This concatenation is unambiguous and does not require additional domain separation.

12.2. Checkpoint Computation Order

The fields within a checkpoint MUST be computed in the following order:

1. Compute the SWF: run Argon2id with the derived seed, then iterate SHA-256 to produce all intermediate states. Construct the Merkle tree to obtain the merkle-root (process-proof key 4).
2. Compute the jitter-seal using the merkle-root as HKDF-Expand PRK input and jitter-binding.intervals as HMAC input.
3. Assemble the jitter-binding structure (intervals, entropy-estimate, jitter-seal).
4. Compute the entangled-mac using the merkle-root as HKDF-Expand PRK input and prev-hash, content-hash, jitter-binding, and physical-state as HMAC input.
5. Compute the checkpoint-hash over prev-hash, content-hash, edit-delta, jitter-binding, physical-state, and merkle-root.

This ordering ensures that each subsequent computation can reference the outputs of prior steps. Implementations MUST follow this order to produce interoperable checkpoints.

12.3. Evidence Protection

For T3 and T4 Attestation Tiers, Evidence Packets MUST be wrapped in a COSE_Sign1 envelope. For T1 and T2 tiers, COSE_Sign1 wrapping is RECOMMENDED. The COSE_Sign1 envelope [RFC9052] provides cryptographic protection during transport. The COSE_Sign1 structure provides:

- * Payload: the CBOR-encoded evidence-packet (including CBOR tag 1347571280)
- * Protected headers: algorithm identifier (ES256 or EdDSA RECOMMENDED)
- * Signature: computed using the Attester's signing key

For T3/T4 tiers, the signing key MUST be bound to a hardware Secure Element (TPM or platform SE). For T1/T2 tiers, a software-managed key is acceptable.

When COSE_Sign1 wrapping is not used (e.g., offline file-based conveyance), the Evidence Packet's integrity relies solely on the internal hash chain. Relying Parties MUST evaluate the trust implications of unwrapped Evidence.

For online conveyance, COSE_Sign1-wrapped Evidence Packets can be encapsulated within a Conceptual Message Wrapper (CMW) for transport via the SEAT cmw_attestation TLS extension [SEAT-EXPAT]. This enables PoP Evidence to be delivered alongside platform attestation evidence in a single post-handshake authentication exchange, which is the preferred attestation timing model [SEAT-Timing]. The SEAT use cases [SEAT-UseCases] identify runtime attestation and operation-triggered re-attestation as key requirements, both of which PoP's continuous checkpoint model satisfies.

13. Sequential Work Function

PoP uses a composite Sequential Work Function (SWF) combining Argon2id [RFC9106] for memory-hardness with iterated SHA-256 for sequential ordering. This construction is NOT a Verifiable Delay Function in the formal sense [Boneh2018]; it does not provide efficient public verification of the delay claim from the output alone.

Instead, verification relies on Merkle-sampled audit proofs: the Attester commits to a Merkle tree over intermediate states, and the Verifier checks a random subset of state transitions. This provides probabilistic verification in $O(k * \log n)$ time where k is the sample count and n is the iteration count.

13.1. Construction

The SWF is computed as follows:

```
state_0 = Argon2id(seed, salt=SHA-256("PoP-salt" || seed), t=1, m=65536, p=1, len=32)
for i in 1..iterations:
    state_i = SHA-256(state_{i-1})
merkle_root = MerkleTree(state_0, state_1, ..., state_iterations).root
```

The salt for Argon2id MUST be derived from the seed: salt = SHA-256("PoP-salt" || seed). This ensures domain separation between the password and salt inputs per RFC 9106 best practices.

The merkle-root field (process-proof key 4) MUST contain the Merkle tree root computed over all intermediate states. The final iteration state (state_iterations) is verified as the leaf at index "iterations" in the Merkle tree.

13.2. Verification Protocol

The Verifier MUST:

1. Recompute Argon2id from the declared seed to obtain state_0
2. For each sampled proof in the Merkle tree, verify the sibling path against the committed root and recompute SHA-256(state_i) to compare against state_{i+1}
3. Verify the final iteration state (state_iterations) by checking its Merkle proof against the committed root (process-proof key 4, merkle-root). If the final-leaf index is not included in the Fiat-Shamir sample set, the Verifier SHOULD additionally derive or request a proof for it.

A minimum of 20 sampled proofs is REQUIRED for CORE profile. ENHANCED profile requires 50 proofs. MAXIMUM profile requires 100 proofs.

13.3. Fiat-Shamir Sample Derivation

Merkle proof sample positions MUST be derived deterministically using a Fiat-Shamir transform to prevent the Attester from selectively including only honestly-computed leaves:

```
sample_seed = SHA-256(merkle_root || process-proof.input)
for j in 0..k-1:
    okm_j = HKDF-Expand(sample_seed, I2OSP(j, 4), 4)
    index_j = OS2IP(okm_j) mod (iterations + 1)
```

Where k is the number of required samples (20 for CORE, 50 for ENHANCED, 100 for MAXIMUM). HKDF-Expand is used with SHA-256 as the underlying hash function per [RFC5869]. I2OSP and OS2IP are the Integer-to-Octet-String and Octet-String-to-Integer primitives as defined in [RFC8017]. The Attester MUST include Merkle proofs for exactly these indices. The Verifier recomputes the sample positions from the committed root and seed, then verifies only those proofs.

If the derivation produces duplicate indices (index_j equal to a previously derived index), the Attester MUST continue generating additional indices by incrementing j beyond k-1 until k distinct indices are obtained. The Verifier MUST verify that all k sample indices are distinct.

Sample indices are in the range [0, iterations] inclusive. Padded Merkle tree leaves (indices greater than iterations) are never sampled by this derivation.

13.4. SWF Seed Derivation

The SWF seed for each checkpoint MUST be derived as:

```
seed = SHA-256(  
    prev-hash ||  
    CBOR-encode(jitter-binding.intervals) ||  
    CBOR-encode(physical-state)  
)
```

For the first checkpoint (sequence = 1):

```
seed = SHA-256(  
    CBOR-encode(document-ref) ||  
    initial-jitter-sample  
)
```

Where initial-jitter-sample is a minimum 32-byte sample of behavioral entropy collected before the first checkpoint. When jitter-binding and physical-state are absent (CORE profile without behavioral data), the seed MUST incorporate at least the prev-hash and a locally-generated 32-byte random nonce: seed = SHA-256(prev-hash || local-nonce). For the first checkpoint, the nonce provides non-determinism when initial-jitter-sample is unavailable. Implementations MUST NOT use a fully deterministic seed derivation.

NOTE: The test vectors in Appendix "SWF Test Vectors" use a simplified fixed seed for implementation validation. Production implementations MUST use the derivation specified above.

13.5. Merkle Tree Construction

The SWF Merkle tree is constructed over all intermediate states as follows:

- * Leaves: state_i for i in 0..iterations, where leaf-index = i and leaf-value = state_i. The total number of leaves is (iterations + 1).

- * Internal nodes: `SHA-256(left_child || right_child)`
- * Tree structure: binary Merkle tree. If the number of leaves is not a power of 2, the tree is padded by duplicating the last leaf until the count reaches the next power of 2.
- * The Merkle root is stored in `process-proof.merkle-root` (key 4).

The final iteration state (`state_iterations`) is the leaf at index "iterations" and is verified by checking its Merkle proof against the committed root.

13.6. Mandatory SWF Parameters

Conforming Attesters MUST use the following minimum SWF parameters for each Evidence Content Tier:

Parameter	CORE	ENHANCED	MAXIMUM
time-cost (t)	1	1	1
memory-cost (m, KiB)	65536	65536	131072
parallelism (p)	1	1	1
iterations	10000	50000	100000
Merkle samples (k)	20	50	100

Table 3

Verifiers MUST reject Evidence where declared proof-params are below the mandatory minimums for the claimed content tier. Expected wall-clock times for the Argon2id phase on reference hardware (DDR4, approximately 25 GB/s memory bandwidth): CORE approximately 50-100ms, ENHANCED approximately 50-100ms, MAXIMUM approximately 100-200ms. The subsequent SHA-256 iterations add approximately 0.1ms per 1000 iterations.

13.7. Entangled MAC Computation

When present, the `entangled-mac` field (checkpoint key 12) MUST be computed as HMAC-SHA-256 [RFC2104] with the following inputs:

```
mac-key   = HKDF-Expand(process-proof.merkle-root,  
                        "PoP-entangled-mac", 32)  
mac-input = prev-hash || content-hash ||  
            CBOR-encode(jitter-binding) ||  
            CBOR-encode(physical-state)  
entangled-mac = HMAC-SHA-256(mac-key, mac-input)
```

Where HKDF-Expand is defined in [RFC5869], || denotes concatenation, and CBOR-encode produces deterministic CBOR per Section 4.2.1 of [RFC8949].

NOTE: In the adversarial Attester model, the Attester generates the SWF output and therefore knows the MAC key. The entangled-mac provides internal consistency binding but does NOT prevent forgery by a malicious Attester (see Section 15.8).

13.8. Jitter Seal Computation

When present, the jitter-seal field (jitter-binding key 3) MUST be computed as HMAC-SHA-256 with the following inputs:

```
seal-key   = HKDF-Expand(process-proof.merkle-root,  
                        "PoP-jitter-seal", 32)  
seal-input = CBOR-encode(jitter-binding.intervals)  
jitter-seal = HMAC-SHA-256(seal-key, seal-input)
```

The jitter-seal binds the timing measurements to the checkpoint's SWF computation, preventing transplantation of jitter data from a different session. It is subject to the same adversarial Attester limitation as the entangled-mac (Section 15.8).

NOTE: In the adversarial Attester model, the Attester generates both the SWF output (from which the MAC key is derived) and the MAC input data. The entangled-mac and jitter-seal therefore provide data integrity binding but do not prevent an adversarial Attester from computing MACs over fabricated data. Their security value is limited to ensuring internal consistency within an honestly-generated checkpoint. See Section 15.

13.9. Security Bound

An adversary who skips fraction f of iterations will be detected with probability $1-(1-f)^k$ where k is the number of sampled proofs. With $k=20$ and $f=0.1$, detection probability exceeds 0.878. With $k=100$ and $f=0.05$, detection probability exceeds 0.994.

This bound holds under the random oracle model for SHA-256. The Attester commits to the Merkle root before sample positions are derived via the Fiat-Shamir transform. Finding a root that biases all k samples away from skipped iterations requires inverting SHA-256, which is computationally infeasible under standard assumptions.

13.10. Hardware-Anchored Time (HAT)

In T3/T4 tiers, the AE MUST anchor the SWF seed to the TPM Monotonic Counter. This prevents "SWF Speed-up" attacks by ensuring that the temporal proof is bound to the hardware's internal perception of time.

14. IANA Considerations

This document requests the following IANA registrations:

14.1. CBOR Tags

This document requests registration of two CBOR tags in the "CBOR Tags" registry per RFC 8949, Section 9.2:

Tag 1347571280:

Tag: 1347571280

Data Item: map

Semantics: PoP Evidence Packet (see Section 12 of this document)

Point of Contact: David Condrey (david@writerslogic.com)

Description of Semantics: [this document]

Tag 1463894560:

Tag: 1463894560

Data Item: map

Semantics: PoP Attestation Result (see [PoP-Appraisal])

Point of Contact: David Condrey (david@writerslogic.com)

Description of Semantics: [this document], [PoP-Appraisal]

14.2. SMI Private Enterprise Number

No SMI Private Enterprise Number is required by this specification's wire format. WritersLogic Inc has requested PEN 65074 for organizational identification purposes only.

14.3. EAT Profile

Registration of the EAT profile URI:
urn:ietf:params:rats:eat:profile:pop:1.0

14.4. Media Types

This document requests registration of the following media types per RFC 6838:

application/vnd.writerslogic-pop+cbor:

Type name: application

Subtype name: vnd.writerslogic-pop+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: binary (CBOR)

Security considerations: See Section 15 of this document

Interoperability considerations: See Section 12 of this document

Published specification: [this document]

Person and email address to contact: David Condrey
(david@writerslogic.com)

application/vnd.writerslogic-war+cbor:

Type name: application

Subtype name: vnd.writerslogic-war+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: binary (CBOR)

Security considerations: See [PoP-Appraisal]

Interoperability considerations: See [PoP-Appraisal]

Published specification: [this document], [PoP-Appraisal]

Person and email address to contact: David Condrey
(david@writerslogic.com)

14.5. TLS Exporter Label

This document registers the following TLS exporter label in the "TLS Exporter Labels" registry defined in [RFC5705]:

Value: EXPORTER-PoP-channel-binding

DTLS-OK: Y

Recommended: Y

Reference: [this document]

15. Security Considerations

This section provides security analysis following [RFC3552] guidelines. The threat model is defined in Section 5 with the adversarial Attester as the primary threat actor. Detailed forensic security analysis is provided in [PoP-Appraisal].

15.1. Primary Threat: Adversarial Attester

Unlike traditional remote attestation where external adversaries threaten system integrity, PoP's primary threat is the Attester operator themselves. The author controls the Attesting Environment and has incentive to claim authenticity for AI-generated or assisted content.

This threat model inversion has fundamental implications:

- * Software-only attestation (T1) provides minimal assurance since the Attester controls all software
- * Cryptographic proofs must be bound to physical constraints the Attester cannot circumvent
- * Behavioral entropy must be economically expensive to forge, not merely cryptographically secure

- * Trust in Evidence scales with the Attestation Tier and the cost of bypassing its guarantees

15.2. Retype Attack Defenses

The retype attack (see Section 5.3.1) is the canonical forgery vector. Defenses are layered:

Cognitive Load Correlation (CLC): Verifiers analyze correlation between content complexity and typing cadence as specified in [PoP-Appraisal].

Error Topology Analysis: Authentic authoring produces characteristic error patterns: corrections localized near recent insertions, deletion-to-insertion ratios consistent with human cognitive models [Salthouse1986], and fractal self-similarity in revision patterns. Retyping produces either unnaturally low error rates or randomly distributed artificial errors.

Temporal Cost: Even successful retype attacks require real-time effort. A 5,000-word document with 10-second checkpoint intervals requires 8+ hours of continuous typing effort to forge. The attack does not scale economically for high-volume forgery.

Relying Parties should be aware that retype attacks remain viable for short documents or high-value targets willing to invest real time. PoP provides graduated assurance proportional to document length and checkpoint density.

15.3. Relay and Replay Attack Defenses

As defined in Section 5.3.2 and Section 5.3.3, these attacks are defeated through Physical Freshness anchors binding Evidence to non-reproducible physical state:

- * Thermal trajectories captured during SWF computation cannot be replayed
- * Kernel entropy pool deltas are bound to specific execution moments
- * Out-of-band presence challenges (QR scans) verify real-time physical proximity

Verifiers MUST reject Evidence where physical freshness markers are stale, inconsistent with timestamps, or exhibit patterns suggesting simulation.

15.4. SWF Acceleration Defenses

As analyzed in Section 5.3.4, specialized hardware attacks are mitigated by:

- * `_Memory-hardness:` Argon2id computation is bounded by memory bandwidth (approximately 50 GB/s for DDR5), not ALU throughput. ASICs provide minimal advantage.
- * `_Hardware-Anchored Time (T3/T4):` SWF seeds are bound to TPM monotonic counters, preventing time compression even with faster computation.
- * `_Merkle sampling:` Skipping SWF iterations is detected probabilistically. With $k=100$ samples, skipping 5% of iterations has >99.4% detection probability.

15.5. Trust Gradation by Tier

Relying Parties should interpret Evidence according to its Attestation Tier:

- T1 (Software-Only): Provides temporal ordering and content binding only. Adversarial Attester can forge all behavioral claims. Suitable only for low-stakes applications or as supplementary evidence.
- T2 (Attested Software): Adds platform attestation hooks but degrades gracefully. Provides moderate assurance against casual forgery but not determined adversaries.
- T3 (Hardware-Bound): Signing keys are hardware-protected. Forgery requires physical access to the Secure Element. Provides strong assurance for most applications.
- T4 (Hardware-Hardened): Anti-tamper evidence and PUF binding. Forgery requires invasive hardware attacks. Suitable for high-stakes legal or financial applications.

15.6. Forgery Cost Bounds

Implementations SHOULD report quantified forgery cost estimates in Attestation Results. For CORE profile (10,000 iterations, $m=65536$ KiB):

- * Sequential computation time: Argon2id with $t=1$, $m=65536$ KiB requires approximately 50-100ms on consumer hardware (DDR4, ~25 GB/s memory bandwidth). The subsequent SHA-256 iterations add negligible time (<1ms for 10,000 iterations).
- * Memory requirement: 64 MiB per concurrent chain
- * Energy cost per checkpoint: approximately \$0.00001 USD at consumer electricity rates

These costs are low for individual checkpoints. Security derives from the conjunctive requirement across many checkpoints: an adversary must sustain consistent behavioral entropy, temporal ordering, and physical state data across the entire chain. The forgery cost scales superlinearly with checkpoint count due to session consistency requirements.

15.7. Denial of Service

SWF verification is asymmetric: Merkle-sampled proofs verify in $O(k \log n)$ versus $O(n)$ generation. Verifiers cannot be overwhelmed by expensive verification requests. Implementations SHOULD implement rate limiting on Evidence submission.

15.8. MAC Field Security Limitations

The entangled-mac and jitter-seal fields are HMAC values keyed from the SWF output. In the adversarial Attester model, the Attester generates the SWF output and therefore knows the MAC key. An adversarial Attester can compute valid MACs over fabricated data (synthetic jitter, manufactured physical state). These fields provide internal consistency checking but do NOT prevent forgery by the Attester. Their value is limited to:

- * Binding data fields to the SWF computation within an honestly-generated checkpoint
- * Providing internal consistency verification (note: the MAC keys are derivable from the public merkle-root field; these MACs do not provide third-party tamper detection)
- * In T3/T4 tiers, the packet-level hardware-bound signature (see Section 8) provides the actual integrity guarantee

15.9. Physical Freshness by Tier

In T1 (Software-Only) and T2 (Attested Software) tiers, the Attester controls all software including the operating system. Physical state readings (thermal trajectories, kernel entropy deltas) are obtained from OS interfaces that the adversarial Attester can intercept or fabricate. Verifiers MUST NOT treat physical-state or physical-liveness fields as evidence of physical freshness in T1/T2 Evidence Packets. Their value in these tiers is limited to increasing the dimensionality of data that an adversary must fabricate consistently.

Physical freshness provides meaningful anti-replay protection only in T3/T4 tiers where hardware attestation binds physical state readings to a trusted execution environment.

15.10. Implementation Security Requirements

Conforming implementations MUST:

- * Use constant-time comparison for all cryptographic operations
- * Zero sensitive memory (keys, jitter data) after use
- * Validate all input lengths and formats before processing
- * Reject Evidence with inconsistent internal state (e.g., checkpoint-hash verification failure)

T3/T4 implementations MUST additionally:

- * Store signing keys exclusively in hardware Secure Elements
- * Bind SWF seeds to TPM monotonic counters
- * Verify platform integrity before Evidence generation

16. Privacy Considerations

This section addresses privacy in accordance with [RFC6973].

16.1. Data Minimization

PoP Evidence Packets do not contain document content. Content binding uses cryptographic hashes (SHA-256) which are computationally irreversible. The author-salted mode (hash-salt-mode=1) provides additional protection by preventing rainbow-table correlation across documents.

16.2. Behavioral Fingerprinting

Jitter sequences in ENHANCED and MAXIMUM profiles constitute behavioral biometrics. To protect author privacy, Verifiers are expected to:

- * Discard jitter data after the verification session completes
- * Avoid correlating jitter across multiple Evidence Packets to prevent author deanonymization
- * Use jitter data solely for authenticity verification

Attesters SHOULD quantize jitter values to reduce fingerprinting precision while preserving statistical validity. A minimum quantization of 5ms is RECOMMENDED.

16.3. Physical State Leakage

Thermal trajectories and kernel entropy deltas in the physical-state field may reveal information about the Attester's hardware configuration. Implementations SHOULD normalize thermal data to relative deltas rather than absolute values to prevent device fingerprinting.

16.4. Unlinkability

Authors who wish to remain pseudonymous SHOULD use per-document signing keys and the author-salted content binding mode to prevent cross-document linkage.

17. References

17.1. Normative References

[PoP-Appraisal]

Condrey, D., "Proof of Process (PoP): Forensic Appraisal and Security Model", Work in Progress, Internet-Draft, draft-condrey-rats-pop-appraisal-04, February 2026, <<https://datatracker.ietf.org/doc/html/draft-condrey-rats-pop-appraisal-04>>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.

- [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>>.
- [RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <<https://www.rfc-editor.org/info/rfc5705>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.
- [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <<https://www.rfc-editor.org/info/rfc8017>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <<https://www.rfc-editor.org/info/rfc9052>>.
- [RFC9106] Biryukov, A., Dinu, D., Khovratovich, D., and S. Josefsson, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications", RFC 9106, DOI 10.17487/RFC9106, September 2021, <<https://www.rfc-editor.org/info/rfc9106>>.

- [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

- [Boneh2018] Boneh, D., Bonneau, J., Bunz, B., and B. Fisch, "Verifiable Delay Functions", CRYPTO 2018, 2018, <https://doi.org/10.1007/978-3-319-96884-1_25>.
- [Dolev-Yao] Dolev, D. and A. Yao, "On the Security of Public Key Protocols", IEEE Transactions on Information Theory 29(2), 198-208, 1983, <<https://doi.org/10.1109/TIT.1983.1056650>>.
- [RATS-Models] Birkholz, H., Eckel, M., Pan, W., and E. Voit, "Reference Interaction Models for Remote Attestation Procedures", Work in Progress, Internet-Draft, draft-ietf-rats-reference-interaction-models-15, 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-rats-reference-interaction-models-15>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.
- [RFC9266] Whited, S., "Channel Bindings for TLS 1.3", RFC 9266, DOI 10.17487/RFC9266, July 2022, <<https://www.rfc-editor.org/info/rfc9266>>.
- [RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <<https://www.rfc-editor.org/info/rfc9711>>.

[Salthouse1986]

Salthouse, T.A., "Perceptual, Cognitive, and Motoric Aspects of Transcription Typing", Psychological Review 93(3), 303-319, 1986, <<https://doi.org/10.1037/0033-295X.93.3.303>>.

[Sardar-RATS]

Sardar, M.U., "Security Considerations for Remote Attestation procedureS (RATS)", Work in Progress, Internet-Draft, draft-sardar-rats-sec-cons-02, February 2026, <<https://datatracker.ietf.org/doc/html/draft-sardar-rats-sec-cons-02>>.

[SEAT-EXPAT]

Sardar, M.U., Fossati, T., Reddy, T., Sheffer, Y., Tschofenig, H., and I. Mihalcea, "Remote Attestation with Exported Authenticators", Work in Progress, Internet-Draft, draft-fossati-seat-expat-01, January 2026, <<https://datatracker.ietf.org/doc/html/draft-fossati-seat-expat-01>>.

[SEAT-Timing]

Sardar, M.U., "Pre-, Intra- and Post-handshake Attestation", Work in Progress, Internet-Draft, draft-usama-seat-intra-vs-post-03, January 2026, <<https://datatracker.ietf.org/doc/html/draft-usama-seat-intra-vs-post-03>>.

[SEAT-UseCases]

Mihalcea, I., Sardar, M.U., Fossati, T., Reddy, T., Jiang, Y., and M. Chen, "Use Cases and Properties for Integrating Remote Attestation with Secure Channel Protocols", Work in Progress, Internet-Draft, draft-mihalcea-seat-use-cases-01, January 2026, <<https://datatracker.ietf.org/doc/html/draft-mihalcea-seat-use-cases-01>>.

SWF Test Vectors

The following test vectors validate SWF implementations.

NOTE: These test vectors use the construction from this specification revision. The salt is derived as SHA-256("PoP-salt" || seed). Implementers should verify their Argon2id output matches state_0 before proceeding with SHA-256 iterations.

Seed: "witnessd-genesis-v1"
Seed (hex): 7769746e657373642d67656e657369732d7631
Salt: SHA-256("PoP-salt" || seed)

Argon2id Parameters:
Time Cost (t): 1
Memory Cost (m): 65536 KiB
Parallelism (p): 1
Output Length: 32 bytes

Iterations: 10,000

Salt (hex): c5de0ba53fa83ab477ead9013bfca978
339e5072882cafb3d0efc8cc40299155

Intermediate States:
state_0 (Argon2id):
a40e0f73832f88dc8bfe5f8956fff4a0
ad2fc4de5455e9d85497c6083b3b1802
state_1000:
c727ead9631eef95ca9a5976a947f71a
6f4f29a5c80aa2dc7f120f9a4193d7b4
state_5000:
d6cba1225d1a2d25dddecfcf2d473020
a19df736878f40ccdfb9334df5af58a5
state_9999:
d7482a780c9e89c787f1ff1e2c566b7b
536260e37d24c539e46de1598321aea2
state_10000 (final):
e445a3cdc8152d66c71366d22b2c5975
cff4d0c8ee6ec0e76515b04d143bd148

Acknowledgements

The author thanks the participants of the RATS working group for their ongoing work on remote attestation architecture and security considerations that informed this specification.

Author's Address

David Condrey
WritersLogic Inc
San Diego, California
United States
Email: david@writerslogic.com