Remote ATtestation procedureS
Internet-Draft
Intended status: Standards Track
Expires: 15 August 2026

D. Condrey
Writerslogic

11 February 2026

Proof of Process: An Evidence Framework for Digital Authorship Attestation
draft-condrey-rats-pop-01

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, human-driven process of creation.

The framework defines a cryptographic mechanism for generating Evidence Packets containing Verifiable Delay Functions (VDFs) to enforce temporal monotonicity and Jitter Seals to bind behavioral entropy (motor-signal randomness) to the document state. These mechanisms allow a Verifier to cryptographically distinguish between human-generated keystrokes, algorithmic generation, and copy-paste operations. Crucially, this verification relies on statistical process metrics and cryptographic binding, enabling authorship attestation without disclosing the semantic content of the document, thereby preserving privacy by design.

About This Document

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

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."

Copyright Notice

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 15 August 2026.

Copyright Notice

Table of Contents

1. Introduction

In the Remote Attestation Procedures (RATS) architecture [RFC9334], "Evidence" is typically a snapshot of system state (e.g., firmware measurements) at a single point in time. However, verifying digital authorship requires attesting to a continuous process rather than a static state. Current mechanisms like digital signatures prove consent, and timestamps (RFC 3161) prove existence, but neither can attest to the provenance of effort—the specific expenditure of time, human attention, and mechanical interaction required to create a document.

This document specifies the Proof of Process (PoP) Evidence Framework, a specialized RATS profile for generating tamper-evident, non-repudiable evidence of an authoring session. It introduces Verifiable Delay Functions (VDFs) to enforce temporal monotonicity (preventing backdating) and Jitter Seals to bind behavioral entropy (human motor-signal randomness) to the document's evolution.

By entangling content hashes with these physical and behavioral constraints, this protocol enables an Attester to generate an Evidence Packet (.pop) that cryptographically distinguishes between human generation, algorithmic generation, and bulk mechanical insertion (paste operations), without requiring privacy-invasive surveillance or revealing the document's semantic content.

2. Claims and Non-Claims

This section is normative. Implementations and Verifier policies MUST distinguish between cryptographic assertions (facts proven by the protocol) and inferential judgements (probabilistic assessments).

2.1. Cryptographic Assertions (Hard Claims)

The Protocol guarantees the following properties relying solely on cryptographic primitives (SHA-256, VDF, HMAC):

* Temporal Ordering: Checkpoint N was created strictly after Checkpoint $N-1$.

* Minimum Effort Cost: The time spent generating the Evidence Chain is the sum of the VDF difficulties, establishing a lower bound on the "cost of forgery" in wall-clock time.

* Chain Integrity: The document state at Checkpoint N is the sole parent of Checkpoint N+1; no history has been inserted or deleted without breaking the hash chain.

* Entropy Binding: The timing data recorded in the evidence was captured prior to the computation of the subsequent VDF proof, preventing "look-ahead" or pre-computation attacks.

2.2. Behavioral Inferences (Soft Claims)

Based on the analysis of the authenticated Evidence, a Verifier MAY infer:

* Source Consistency: The statistical likelihood that the input stream (keystroke dynamics) belongs to a single continuous actor.

* Anomaly Detection: The presence of discontinuities (e.g., sudden changes in typing rhythm) that correlate with tool usage or copy-paste operations.

2.3. Excluded Claims (Non-Claims)

This protocol explicitly does NOT support the following claims:

* "Human vs. AI" Classification: The protocol measures signal characteristics (entropy, rhythm), not cognitive origin. A high-entropy signal is "consistent with human input," not "proven human thought."

* "Cheating" or "Plagiarism": These are policy judgements, not technical facts. The protocol reports events (e.g., "large text block inserted"); the Relying Party determines if this constitutes a policy violation.

* Identity Attribution: While the evidence binds to a signing key, it does not inherently bind to a specific legal identity unless combined with external PKI or biometric identity assertions.

3. Problem Statement

Digital documents lack creation-process provenance. COSE [RFC9052] signatures prove key possession; RFC 3161 [RFC3161] timestamps prove existence-but neither reveals _how_ the document evolved.

Existing approaches fail modern needs:

* Surveillance (screen/keystroke logging): Privacy-violating, requires third-party trust, unverifiable without archives.

* Content analysis (stylometry/AI detectors): Probabilistic, adversarial-vulnerable, product-only (no process).

Required traits: privacy-preserving (hash-only, SHA-256 [RFC6234]), independently verifiable (self-contained proofs), tamper-evident (hash/HMAC [RFC2104]/VDF chains), process-documenting (evolution, not contents).

Use cases: academic integrity (AI sophistication), legal provenance, creative attribution, professional standards.

4. Scope

4.1. What This Specification Defines

* Evidence format (.pop): Merkle trees (SHA-256), entropy bindings, VDF proofs [Pietrzak2019] [Wesolowski2019] (CBOR [RFC8949], tag 1347571280).

* Result format (.war): Verifier appraisals (COSE, EAT [RFC9711], tag 1463894560).

* Checkpoint structure: Content hashes (SHA-256 [RFC6234]), timing proofs, behavioral summaries.

* Verification procedures: Self-contained, optional RFC 3161 anchors.

* Claim taxonomy: Chain-verifiable vs. monitoring-dependent (CDDL [RFC8610]).

4.2. What This Specification Does NOT Define

* Content analysis: No stylometry/semantics (hash-only, SHA-256).

* Author ID: No person claims (key-bound via COSE [RFC9052]).

* Intent/cognition: No mental-state inference.

* AI classification: Process evidence only; policy-based interpretation.

* Surveillance: No capture/logging/monitoring (timing histograms only).

These exclusions enable privacy-by-construction in the RATS [RFC9334] profile.

4.3. Relationship to RATS

RATS roles:

Attester: witnessd-core (local .pop production: Merkle/SHA-256, VDF, entropy).

Verifier: Parses/appraises .pop -> signed .war (COSE).

Relying Party: Consumes .war (institutions/publishers/legal).

Extensions: HMAC-SHA256 [RFC2104] entropy; VDFs for sequential time (relative + RFC 3161 [RFC3161] absolute).

5. Design Goals

Four principles guide this RATS profile (SHA-256, COSE, HMAC, CBOR/CDDL, VDFs, RFC 3161):

5.1. Privacy by Construction

Structural enforcement (CBOR/CDDL): No content (SHA-256 [RFC6234] hashes only); no keystrokes (ms intervals, histogrammed); no visuals; aggregates prevent reconstruction. Schema violations impossible.

5.2. Zero Trust

RATS aligned: Local generation (SHA-256, VDF, HMAC, COSE); self-contained CBOR verification (optional RFC 3161 [RFC3161]); multi-Verifier adversarial appraisal (CDDL schemas).

5.3. Evidence Over Inference

CBOR facts (SHA-256/HMAC/VDF traceable); claims classified (computationally-bound vs. monitoring-dependent, CDDL ae-trust-basis); COSE results document verification (entropy/VDF/TPM [TPM2.0]/RFC 3161 factors); no authorship/intent/authenticity absolutes-Relying Party policy (EAT [RFC9711]).

5.4. Cost-Asymmetric Forgery

VDFs enforce sequential time; SHA-256 entropy commits irrecoverable timings; HMAC chains cascade invalidation. Selective forgery recomputes downstream VDFs (non-parallel). Section 26 quantifies (economics > value). _Forgery possible but costly_-complements SHA-256/HMAC/COSE.

6. Terminology

BCP 14 [RFC2119] [RFC8174] applies. PPPP avoids PPP (RFC 1661)/PoP (RFC 5280) conflicts.

Key terms (CBOR, SHA-256, HMAC, COSE, VDFs):

PPPP Evidence (.pop): [RFC9334] Attester artifact: Merkle trees (SHA-256), HMAC entropy, VDFs (CBOR tag 1347571280, hex 0x50505020, ASCII "PPPP"; CDDL [RFC8610]). Raw metrics (linearity, edits, fatigue, spectral) uninterpreted.

PPPP Result (.war): Verifier Attestation Result (COSE, CBOR tag 1463894560, hex 0x57415220, ASCII "WAR "). Policy-based source consistency; varying Verifier outputs.

Residency: Hardware origin (software -> TPM 2.0 [TPM2.0]/Enclave).

Sequence: VDF min-time (non-parallel).

Behavioral Consistency: Unified process stats (timing/edit evolution). Histograms privacy-protect raw intervals.

SA-VDF: Pietrzak VDF HMAC hardware-bound (no fast migration).

7. Document Structure

Builds on RATS: CBOR/CDDL, SHA-256/COSE verification.

* Section 9: Architecture, RATS roles, formats (tags 1347571280/1463894560).
* Section 10: HMAC-SHA256 entropy binding.
* Section 24: VDFs temporal proofs.
* Section 25: Claims (SHA-256/HMAC bound vs. monitoring).
* Section 26: VDF economics.
* Section 32: Threats/mitigations.
* Section 33: Behavioral handling.
* Section 37: Tags/EAT/media types.

Appendices: CDDL schemas, SHA-256 vectors, guidance (RATS Attesters/ Verifiers).

Companion documents: [I-D.condrey-rats-pop-protocol] (transcript format), [I-D.condrey-rats-pop-schema] (CDDL schema), [I-D.condrey-rats-pop-examples] (examples and test vectors).

8. Conventions and Definitions

8.1. Domain Separation Constants

To prevent cross-protocol attacks, all HMAC and KDF operations MUST use explicit domain separation labels. The following constants are defined:

* ‘DST_JITTER‘: "witnessd-jitter-binding-v1"
* ‘DST_CHAIN‘: "witnessd-chain-mac-v1"

* ‘DST_CLOCK‘: "witnessd-entropic-clock-v1"

* ‘DST_LINK‘: "witnessd-link-token-v1"

8.2. CDDL Notation

Data structures in this architecture document are specified using the Concise Data Definition Language (CDDL) [RFC8610], a notation by which CBOR [RFC8949] and JSON data structures may be expressed with precision and clarity, ensuring that implementers have unambiguous guidance for encoding and decoding Evidence Packets and Attestation Results. The normative CDDL definitions appear inline in the relevant sections, providing immediate context for the structures being described, and a complete consolidated schema is afforded in the appendices for implementers who require a single authoritative reference. The CDDL notation is used throughout this specification to define structures including checkpoints with SHA-256 [RFC6234] hash bindings, jitter-binding structures with HMAC [RFC2104] authentication, VDF proofs [Pietrzak2019] [Wesolowski2019], and COSE [RFC9052] signatures, with all type definitions following the conventions established in RFC 8610.

8.3. CBOR Encoding

CBOR encoding per RFC 8949 is used by both Evidence Packets and Attestation Results, providing efficient binary encoding with support for semantic tags and extensibility that is well-suited for the compact representation of cryptographic evidence including SHA-256 hashes, HMAC bindings, VDF proofs, and COSE signatures. Semantic tags for type identification are employed to enable format detection without external metadata: Evidence Packets use the PPPP tag (1347571280) and Attestation Results use the WAR tag (1463894560), as defined in Section 6. Integer keys in the range 1-99 are reserved for core protocol fields defined by this specification to minimize encoding size, while string keys are used for vendor extensions and application-specific fields that extend beyond the base CDDL schema. Deterministic encoding as specified in RFC 8949 Section 4.2 is RECOMMENDED for signature verification, ensuring that the same logical structure always produces identical byte sequences when computing SHA-256 hashes or verifying COSE signatures, with map keys sorted in bytewise lexicographic order, integers encoded in minimal representation, and floating-point values canonicalized.

8.4. COSE Signatures

COSE (CBOR Object Signing and Encryption) per RFC 9052 is used for cryptographic signatures throughout this specification, providing a standardized mechanism for authenticating Evidence Packets and Attestation Results within the CBOR encoding framework. Single-signer signatures suitable for Evidence and Attestation Result authentication are afforded by the COSE_Sign1 structure defined in RFC 9052, which includes a protected header containing the algorithm identifier, an unprotected header for optional metadata, and the signature bytes computed over the CBOR encoded payload. EdDSA with Ed25519 is RECOMMENDED for new implementations due to its performance characteristics (fast signing and verification), resistance to timing attacks through constant-time implementation, and compact signature size (64 bytes), while ECDSA with P-256 as defined in RFC 9052 is supported for compatibility with existing PKI infrastructures and hardware security modules including TPM 2.0 [TPM2.0]. The algorithm selection is indicated within the COSE protected header using registered algorithm identifiers, allowing Verifiers to determine the appropriate verification procedure without external negotiation.

8.5. EAT Tokens

An Entity Attestation Token (EAT) profile per RFC 9711 [RFC9711] is delineated by this architecture document, extending the RATS [RFC9334] attestation framework with domain-specific claims for behavioral evidence and process documentation. A framework for attestation claims with support for custom claim types is afforded by EAT, making possible the expression of Proof of Process claims including forensic-assessment verdicts, presence-score values, evidence-tier levels, and AI-composite-scores within a standardized structure encoded in CBOR and signed with COSE. The EAT profile URI for Proof of Process evidence is https://example.com/rats/eat/profile/pop/1.0, with IANA registration to be requested upon working group adoption as detailed in Section 37. Custom EAT claims proposed for IANA registration extend the standard EAT claim set with claims specific to behavioral evidence (pop-presence-score, pop-ai-composite-score), temporal evidence (VDF duration bounds), and process documentation (segment counts, entropy thresholds), enabling interoperability between RATS implementations that support this profile.

8.6. Hash Function Notation

The following notation for cryptographic hash functions is used throughout this architecture document, with all hash operations conforming to the algorithms specified in RFC 6234 unless otherwise indicated: H(x) denotes the SHA-256 hash of input x, producing a 256-bit (32-byte) output that serves as the default hash algorithm for content hashes, segment hashes, and entropy commitments; H^n(x) denotes n iterations of hash function H as used in iterated-hash VDF constructions; and HMAC(k, m) denotes HMAC-SHA256 per RFC 2104 with key k and message m, used for binding operations including the chain-mac and jitter binding-mac. SHA-256 is the RECOMMENDED hash algorithm for all operations, being widely implemented across platforms (including hardware acceleration in modern processors), well-analyzed by the cryptographic community, and resistant to known cryptanalytic attacks including collision, preimage, and second-preimage attacks. Implementations MAY support SHA3-256 for algorithm agility as indicated in the CDDL hash-algorithm enumeration, particularly in environments where resistance to potential future attacks on the SHA-2 family is prioritized or where regulatory requirements mandate SHA-3 support; when SHA3-256 is used, the HMAC construction remains valid as HMAC is hash-function-agnostic.

9. Evidence Model

In this section, the top-level architecture of the witnessd Proof of Process evidence model is delineated, with the design following the RATS (Remote ATtestation procedureS) architecture [RFC9334] while introducing domain-specific extensions for behavioral evidence encoded in CBOR [RFC8949], cryptographic proofs computed using SHA-256 [RFC6234] and HMAC [RFC2104], temporal ordering via VDFs [Pietrzak2019] [Wesolowski2019], and process documentation structured according to CDDL [RFC8610] schemas. Both the structural components and their relationships are described, establishing the foundation upon which subsequent sections build, with particular attention to the cryptographic bindings that ensure tamper-evidence, the COSE [RFC9052] signatures that provide authentication, and the EAT [RFC9711] profile that enables interoperability with other RATS implementations.

9.1. RATS Architecture Mapping

A RATS profile is implemented by this specification with the following role mappings that establish the correspondence between RATS entities and Proof of Process components: the witnessd-core library acts as Attester in the RATS model, producing Evidence Packets (.pop files) encoded in CBOR with semantic tag 1347571280, containing segment-based Merkle trees with SHA-256 hash linkage, VDF

proofs, and jitter bindings authenticated via HMAC; verification implementations act as Verifiers in the RATS model, parsing CBOR encoded Evidence Packets per the CDDL schema and producing Attestation Results (.war files) signed with COSE; and consuming entities (academic institutions, publishers, legal systems) act as Relying Parties in the RATS model, interpreting the EAT claims in Attestation Results to make trust decisions. Evidence is generated locally on the Attester device without network dependency, with all cryptographic operations including SHA-256 hashing, VDF computation, and COSE signing performed using only local resources. Verification requires only the CBOR encoded Evidence packet itself, cryptographic hashes computed via SHA-256 are contained in Evidence rather than document content, and behavioral signals are aggregated into histograms before inclusion, affording a privacy-preserving attestation mechanism that requires no trusted infrastructure beyond the Attesting Environment and optional external anchors such as RFC 3161 timestamps.

9.2. Evidence Flow

PPPP operates in the RATS passport model: the Attester generates Evidence locally without network dependency, and Evidence is conveyed to the Verifier out of band for deferred appraisal. No real-time interaction between Attester and Verifier is required for evidence generation.

The evidence flow proceeds as follows:

1. The Attesting Environment runs locally alongside the authoring tool, capturing edit operations, timing intervals, and document state transitions as they occur.

2. At each checkpoint, the Attesting Environment computes a content hash (SHA-256), commits behavioral entropy via HMAC, and computes a VDF temporal proof binding content, timing, and previous checkpoint state into a chain.

3. On session completion, the Attesting Environment packages all checkpoints into a signed Evidence Packet (.pop) using COSE.

4. The Evidence Packet is conveyed to a Verifier at a time determined by the author or Relying Party - potentially minutes, days, or months after creation.

5. The Verifier independently appraises the Evidence Packet, producing an Attestation Result (.war) documenting what was verified, with confidence scores and caveats.

When a Relying Party requires proof of freshness, an OPTIONAL verifier-provided nonce MAY be incorporated into the Evidence Packet's final signature. This is the only interactive element in the protocol and is not required for evidence generation.

9.3. Source Consistency Analysis

The core analytical claim of PPPP is source consistency: whether the evidence chain reflects a single coherent generative process throughout a document's lifecycle. The framework does not classify content as human-written or AI-generated. It detects transitions in the character of the generative process and maps them as source consistency events.

Source consistency is evaluated across the checkpoint chain by measuring behavioral characteristics at each checkpoint and analyzing their coherence over time. Characteristics include edit operation type distribution (ratio of insertions, deletions, revisions, structural edits, and navigation events), timing patterns relative to content complexity, revision density, and temporal evolution of behavioral metrics across the session.

The following source consistency transition patterns are defined as informational guidance for Verifier implementers:

Pattern	Signature	Interpretation
Consistent	All checkpoints conform	Single source, stable process throughout
Sudden transition	Conforming then non-conforming	Late-stage process change or handoff
Gradual drift	Conformity degrades progressively	Increasing process assistance over time
Intermittent	Alternating conformity	Hybrid workflow with multiple sources
Bookend	Non-conforming opening and closing	Different process for introduction and conclusion

Table 1

These patterns are not normative verification gates. The Verifier records the pattern; the Relying Party decides whether the pattern is acceptable for their use case. A hybrid workflow may be entirely appropriate for some domains and unacceptable in others.

9.4. Decision History

Every edit operation in the evidence chain - every insertion, deletion, revision, and structural edit - represents a creative decision. The sequence of these decisions constitutes the authoring process. PPPP captures this decision history as the primary evidence artifact.

Edit operations are classified by type without recording content:

Composition: New text creation - insertions that extend the document.

Revision: Modification of existing text - deletions followed by insertions at the same location, select-and-replace operations.

Structural: Document reorganization - cut and paste, section reordering, large-scale moves.

Navigation: Cursor movement without content change - reading, reviewing, positioning for subsequent edits.

The distribution and sequencing of these operation types over the evidence chain is itself a fingerprint of the authoring process. Composition produces varied operation sequences with revisions, cursor movements, and structural edits interspersed among insertions. Transcription produces predominantly monotonic insertion streams with occasional single-character corrections. The evidence chain records these patterns without judging them.

9.5. Privacy-Preserving Document Classification

Source consistency is evaluated against domain-appropriate expectations. A short essay legitimately written front-to-back has different expected characteristics than a novel written over months with non-linear revision. The document profile is derived from behavioral signals without accessing content:

Sentence length distribution: Character count between sentence-boundary keystrokes (period, space, shift sequences).

Paragraph rhythm: Frequency and regularity of paragraph-break operations.

Vocabulary complexity proxy: Word length distribution derived from character counts between space keystrokes.

Revision density: Edit operations per checkpoint, ratio of deletions to insertions.

Structural edit frequency: Cut/paste operations, cursor movements beyond local context, select-and-replace events.

The Attesting Environment computes this classification locally and includes it as a document-profile field in the Evidence Packet. The author MAY additionally declare a document type. When both behavioral classification and author declaration are present, the Verifier can assess their consistency - divergence between declared type and observed behavioral profile is itself a signal that the Relying Party may evaluate.

9.6. Input Event Trust Boundary

The Attesting Environment captures input timing at the OS HID event layer. This establishes the trust boundary for behavioral entropy collection. The trust boundary differs by assurance tier:

Tier	Input Trust Boundary	Injection Defense	Residual Risk
T1-T2	OS HID event layer	VDF cost asymmetry, chain HMAC, content binding	Privileged software injection
T3	OS HID + TPM signing	Above + hardware-bound key, platform measurement	Injection without boot chain alteration
T4	TEE interrupt capture	Above + pre-OS event capture	Enclave compromise

Table 2

At T1 and T2, the adversary model assumes the OS input stack is not compromised. Synthetic event injection by a privileged attacker is not prevented by the protocol but is made economically costly by VDF-jitter entanglement and content binding. At T3, TPM-bound signing constrains evidence to specific hardware without protecting the input path. At T4, TEE-based capture moves the trust boundary below the OS, requiring enclave compromise for input injection.

Evidence metadata includes the input transport class (USB HID, built-in keyboard, Bluetooth Classic, BLE) so that Verifiers can adjust confidence based on the transport's timing fidelity. Bluetooth connections introduce variable latency (5-30ms) that degrades behavioral signal quality; this is reflected in reduced confidence scores rather than evidence rejection.

9.7. Two Complementary Formats

Two file formats that serve distinct roles in the attestation workflow defined by the RATS architecture are delineated by the witnessd protocol, each encoded using CBOR per the CDDL schemas in the appendices, with registered semantic tags for type identification that enable parsers to determine the packet type by examining the leading tag value.

9.7.1. Evidence Packet (.pop)

The primary Evidence artifact produced by the Attester in the RATS architecture is the .pop (Proof of Process) file, containing all cryptographic proofs including SHA-256 hash chains, HMAC bindings, VDF outputs, and behavioral evidence accumulated during document authorship, encoded using CBOR with the PPPP tag (1347571280) and structured according to the evidence-packet type in the CDDL schema. The authoritative record of the authoring process is constituted by the Evidence packet, which may be submitted to a Verifier for appraisal per the RATS workflow, archived alongside the document for future verification using only the cryptographic primitives (SHA-256, HMAC, VDF) without access to external services, or shared with Relying Parties who perform their own verification using the CDDL schema and verification procedures defined in this specification. Larger file sizes than the .war file are typical for .pop files because complete segment-based Merkle trees with SHA-256 linkage, full VDF proofs for each inter-segment interval, and behavioral evidence including jitter histograms and entropy commitments are contained within them.

9.7.2. Attestation Result (.war)

The Attestation Result produced by a Verifier after appraising an Evidence packet per the RATS architecture is the .war (Writers Authenticity Report) file, which serves as a portable verification certificate signed with COSE that may be distributed independently of the original Evidence, encoded using CBOR with the WAR tag (1463894560) and conforming to the EAT profile defined in this specification. Distribution alongside published documents is the intended use of the Attestation Result, which affords a COSE signed verdict from a trusted Verifier (the forensic-assessment enumeration value), a summary of verified claims derived from SHA-256 hash chain verification and VDF recomputation without including the full evidence, a confidence score in the range [0.0, 1.0] for Relying Party decision-making incorporating factors such as entropy sufficiency and calibration attestation presence, and caveats documenting verification limitations such as missing hardware attestation via TPM [TPM2.0] or pending external anchor confirmations from RFC 3161 timestamps. The .war file may be trusted by Relying Parties based on the Verifier's reputation and the COSE signature validation, or the original .pop file may be requested for independent verification using the CDDL schema and cryptographic primitives (SHA-256, HMAC, VDF) defined in this specification. This flexibility makes possible a range of trust models within the RATS framework, from fully delegated verification where Relying Parties trust the Verifier's EAT claims, to adversarial multi-verifier scenarios where multiple independent Verifiers appraise the same Evidence.

9.7.3. Format Relationship

Linkage between the two CBOR encoded formats is established by the reference-packet-id field in the Attestation Result, which matches the packet-id of the appraised Evidence packet, with both identifiers being UUIDs per RFC 9562 [RFC9562] to ensure global uniqueness across all Evidence packets ever produced. The reference-packet-id is included in the COSE signed payload of the Attestation Result, ensuring that any attempt to modify the binding would invalidate the Verifier's signature. Unambiguous binding of each Attestation Result to a specific Evidence packet is ensured by this construction, preventing substitution attacks wherein an Attestation Result signed with COSE might be fraudulently associated with a different Evidence packet, a property that is critical for the RATS trust model where Relying Parties may receive Attestation Results from Verifiers they trust without access to the original Evidence. The UUID format provides 122 bits of entropy when using random UUIDs (version 4), making collision probability negligible even across billions of Evidence packets.

9.8. Evidence Packet Structure

The complete attestation evidence produced by the Attester in the RATS architecture is contained in the evidence-packet structure, which encapsulates all cryptographic proofs including SHA-256 hash chains, HMAC bindings, and VDF outputs, as well as behavioral evidence captured during the authoring process. A normative CDDL definition is afforded in the schema appendix with complete type definitions and constraints; in this section, the semantic meaning of each component is described to guide implementers in constructing and parsing CBOR encoded Evidence packets. The structure employs CBOR encoding throughout with integer keys in the range 1-99 reserved for core protocol fields to minimize encoding size, while string keys are permitted for vendor extensions that extend the base CDDL schema.

```
evidence-packet = #6.1347571280({
        1 => uint,                      ; version (1)
        2 => vdf-structure,             ; VDF
        3 => jitter-seal-structure ; Mandatory in v1.1+,     ; Jitter Seal
        4 => content-hash-tree,         ; Merkle for segments
        5 => correlation-proof,         ; Spearman Correlation
        6 => error-topology,            ; Fractal Error Pattern
        7 => hardware-attestation,       ; Hardware Assurance Binding
        8 => process-metrics,             ; Raw Process Measurements


        * tstr => any,                  ; extensions
})

vdf-structure = {
        1 => bstr,                      ; input: H(DST_CHAIN || content || jitter_seal)
        2 => bstr,                      ; output
        3 => uint64,                    ; iterations
        4 => [* uint64],                ; rdtsc_checkpoints (Continuous calib)
        5 => bstr,                      ; entropic_pulse: HMAC(SK, T ^ E)



}

jitter-seal-structure = {
        1 => tstr,                      ; lang (e.g., "en-US")
        2 => bstr,                      ; bucket_commitment (ZK-Private)
        5 => int .within -100..100,     ; pink_noise_slope_decibits (-10.0..10.0)

        3 => uint,                      ; entropy_millibits
}
```

```
content-hash-tree = {
        1 => bstr,                      ; root
        2 => uint16 .ge 20,             ; segment_count
}

correlation-proof = {
        1 => int16 .within -1000..1000, ; rho (Scaled: -1000..1000)
        2 => 700,                       ; threshold (0.7 * 1000)
}

process-metrics = {
        1 => ratio-millibits,           ; linearity-score
        2 => ratio-millibits,           ; structural-edit-ratio
        3 => int,                       ; hesitation-phase-offset (signed millibits)
        4 => ratio-millibits,           ; revision-clustering
        5 => ratio-millibits,           ; fatigue-slope
        6 => uint,                      ; checkpoint-count
        7 => uint,                      ; total-duration-ms
        ? 8 => [+ ratio-millibits],     ; per-checkpoint-conformity-scores
}
```

9.8.1. Required Fields

The required fields in the evidence-packet structure provide the essential metadata and cryptographic content needed for verification per the RATS architecture, with each field encoded according to the CDDL schema in the appendix. The version field (key 1) indicates the schema version number, currently 1, and implementations MUST reject packets with unrecognized major versions to ensure forward compatibility with future revisions of this CBOR schema. The profile field (key 2) contains the EAT profile URI (https://example.com/rats/eat/profile/pop/1.0) that identifies this specification, with IANA registration to be requested upon working group adoption as detailed in Section 37. The packet-id field (key 3) is a UUID per RFC 9562 [RFC9562] uniquely identifying this Evidence packet, generated by the Attester at packet creation time using a cryptographically secure random source. The created field (key 4) is a timestamp indicating when this packet was finalized, encoded using CBOR tag 1 (epoch-based date/time) per RFC 3339 [RFC3339] conventions; note that this timestamp is informational and not cryptographically protected, with temporal ordering established instead by VDF causality. The document field (key 5) contains the document-ref structure binding the Evidence to the documented artifact via SHA-256 content hash as described in Section 9.10. The checkpoints field (key 6) is an segment-based Merkle tree of content hashes forming the evidence chain with SHA-256 hash linkage and VDF proofs as described in Section 9.9.

9.8.2. Tiered Optional Sections

The optional sections (keys 10-17) in the CDDL schema correspond to evidence tiers that determine the strength of assurance provided by the CBOR encoded Evidence packet within the RATS architecture. Higher tiers require additional data collection and produce larger packets, but afford stronger evidence for Verifiers appraising claims. The presence-section (key 10) contributes to Standard tier by adding human presence challenges with timing verified against human reaction time baselines. The forensics-section (key 11) and keystroke-section (key 12) and hardware-section (key 13) are REQUIRED for Enhanced tier by adding edit topology analysis, AI indicator scores, and detailed jitter samples with entropy commitments computed using SHA-256 and bound via HMAC. The hardware-section (key 13) is REQUIRED for Enhanced and Maximum tiers by adding TPM 2.0 or Secure Enclave attestation with device-bound keys. The external-section (key 14) contributes to Maximum tier by adding RFC 3161 timestamps and optional blockchain anchors for absolute time binding. The absence-section (key 15, detailed in Section 25) contributes to Maximum tier by documenting negative evidence claims with explicit trust requirements. The forgery-cost-section (key 16, detailed in Section 26) contributes to Maximum tier by quantifying the computational cost of VDF recomputation and behavioral simulation. The declaration (key 17) may appear at all tiers and contains author attestations signed with COSE.

9.8.3. Extensibility

The evidence-packet structure defined in CDDL supports forward-compatible extensions through string-keyed fields, following the CBOR conventions for extensible maps that allow new fields to be added without breaking existing implementations. Integer keys in the range 1-99 are reserved for this specification and future versions thereof, providing space for additional standardized fields while maintaining compact CBOR encoding. String keys MAY be used for vendor or application-specific extensions that are not part of the core CDDL schema, enabling domain-specific customizations such as additional metadata fields or alternative evidence formats. Verifiers MUST ignore unrecognized string-keyed fields per the RATS extensibility model, allowing Evidence packets with vendor extensions to be verified by any compliant implementation. Verifiers MUST reject packets containing unrecognized integer keys in the reserved range (1-99) to prevent future standardized fields from being misinterpreted by older implementations, ensuring that cryptographic verification using SHA-256 and HMAC is only performed on packets that conform to a known schema version.

9.9. Segment Tree Chain

The core evidentiary structure in the RATS profile defined by this specification is constituted by the segment chain, which forms the backbone of the Evidence packet's cryptographic guarantees. Each checkpoint represents a witnessed document state at a specific point in the authoring process, cryptographically linked to its predecessor via SHA-256 hashes that create an immutable sequence. This chain construction, wherein each element commits to its predecessor through the prev-hash field, makes possible tamper-evident sequences that cannot be modified without invalidating all subsequent elements: any change to segment N invalidates the prev-hash in segment N+1, which in turn invalidates segment N+1's hash used in segment N+2, and so on through the entire chain. The VDF proofs entangled with each checkpoint further strengthen this construction by ensuring that recomputation of the chain from any modification point requires sequential time proportional to the number of subsequent checkpoints, with jitter bindings authenticated via HMAC ensuring that behavioral entropy cannot be transplanted between checkpoints, and the chain-mac computed using HMAC-SHA256 preventing checkpoint transplantation between sessions.

9.9.1. Checkpoint Structure

```
checkpoint = {
        1 => uint,                      ; sequence
        2 => uuid,                      ; checkpoint-id
        3 => pop-timestamp,             ; timestamp
        4 => hash-value,                ; content-hash
        5 => uint,                      ; char-count
        6 => uint,                      ; word-count
        7 => edit-delta,                ; delta
        8 => hash-value,                ; prev-hash
        9 => hash-value,                ; tree-root
        10 => vdf-proof,                ; vdf-proof
        11 => jitter-binding,           ; jitter-binding
        12 => bstr .size 32,            ; chain-mac

        * tstr => any,                  ; extensions
}
```

sequence (key 1): Zero-indexed ordinal position in the segment chain. MUST be strictly monotonically increasing.

checkpoint-id (key 2): UUID uniquely identifying this checkpoint within the packet.

timestamp (key 3): Local timestamp when the checkpoint was created.

Note that local timestamps are untrusted; temporal ordering is established by VDF causality.

content-hash (key 4): Cryptographic hash of the document content at this checkpoint. SHA-256 RECOMMENDED.

char-count (key 5), word-count (key 6): Document statistics at this checkpoint. Informational only; not cryptographically bound.

delta (key 7): Edit delta since previous checkpoint. Contains character counts for additions, deletions, and edit operations. No content is included.

prev-hash (key 8): Hash of the previous checkpoint (tree-root{N-1}). For the genesis checkpoint (sequence = 0), this MUST be 32 zero bytes.

tree-root (key 9): Binding hash computed over all checkpoint fields, creating the hash chain.

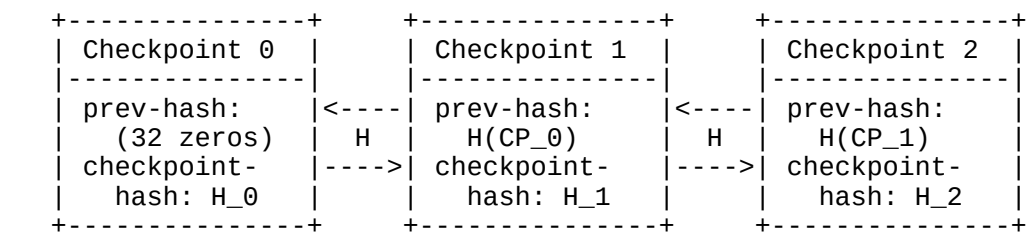
vdf-proof (key 10): Verifiable Delay Function proof establishing minimum elapsed time. See Section 24.

jitter-binding (key 11): Captured behavioral entropy binding. See Section 10.

chain-mac (key 12): HMAC-SHA256 binding the checkpoint to the chain key, preventing transplantation of checkpoints between sessions.

9.9.2. Hash Chain Construction

A cryptographic hash chain is formed by the segment chain through the prev-hash linkage. The construction employs SHA-256 as the default hash algorithm, though algorithm agility is supported for future requirements:



The tree-root is computed as:

```
tree-root = H(
        "witnessd-checkpoint-v1" ||
        sequence ||
        checkpoint-id ||
        timestamp ||
        content-hash ||
        char-count ||
        word-count ||
        CBOR(delta) ||
        prev-hash ||
        CBOR(vdf-proof) ||
        CBOR(jitter-binding)
)
```

By this construction, any modification to any field in any checkpoint is ensured to invalidate all subsequent segment hashes, thereby affording tamper-evidence for the entire chain. The cascading nature of this invalidation makes selective tampering impractical, as an adversary would need to recompute all VDF proofs from the modification point forward.

9.9.3. Merkle Tree Construction

The segment chain is further structured as a standard binary Merkle Tree (RFC 6962), where each segment hash serves as a leaf. This construction enables efficient logarithmic-time inclusion proofs for subsets of segments.

External anchors commit to the Merkle root of the entire authoring session, thereby affording tamper-evidence for all segments with a single external signature. Verifiers MAY validate inclusion of specific segments by verifying the Merkle path from the segment leaf to the anchored root.

9.9.4. Evidence Format Versions

The evidence-packet version field (key 1) indicates the format version used for evidence generation. This specification defines two versions with distinct security properties:

Version 1.0 (Legacy Parallel Mode): In version 1.0, VDF computation and jitter capture MAY proceed in parallel. The jitter commitment is bound to the final evidence packet but is not entangled with the VDF input chain. This mode permits faster evidence generation but provides weaker temporal guarantees: an adversary with pre-computed VDF outputs could potentially substitute jitter data without VDF recomputation. Version 1.0 evidence SHOULD be treated with reduced confidence for temporal claims.

Version 1.1 (Entangled Computation Mode): In version 1.1, jitter capture MUST complete before VDF computation begins for each checkpoint. The jitter-binding entropy-commitment is incorporated into the VDF input:

```
VDF_input{N} = H(
    VDF_output{N-1} ||
    content-hash{N} ||
    jitter-binding{N}.entropy-commitment ||
    sequence{N}
)
```

This entanglement creates a causal dependency: the jitter data MUST exist before VDF computation can proceed. An adversary attempting to substitute jitter data must recompute the entire VDF chain from that point forward, incurring the full temporal cost. Version 1.1 is REQUIRED for new implementations and provides the security guarantees described throughout this specification.

Verifiers MUST check the version field and SHOULD apply appropriate confidence adjustments:

* Version 1.1: Full confidence in temporal binding and VDF guarantees.

* Version 1.0: Reduced confidence; temporal claims limited to "evidence existed at some point" rather than "evidence was generated over the claimed duration."

* Unknown versions: Verification SHOULD fail with an error indicating unsupported format version.

The verifier_nonce field (when present) is incorporated into the packet signature regardless of version: SIG_k(H3 || verifier_nonce). This provides replay prevention independent of VDF entanglement mode.

9.10. Document Binding

Binding of the Evidence packet to a specific document without including the document content is accomplished by the document-ref structure. Cryptographic hashes computed using SHA-256 are employed to establish this binding, allowing verification that a document corresponds to an Evidence packet without revealing the document content to parties who do not already possess it.

```
document-ref = {
        1 => hash-value,                ; content-hash
        2 => tstr,                      ; filename (optional)
        3 => uint,                      ; byte-length
        4 => uint,                      ; char-count
        ? 5 => hash-salt-mode,          ; salt mode
        ? 6 => bstr,                    ; salt-commitment
}
```

9.10.1. Content Hash Binding

The cryptographic hash of the final document state is represented by the content-hash (key 1), which is the same value as the content-hash in the final checkpoint. Document binding verification is accomplished by computing H(document-content) using SHA-256, comparing with document-ref.content-hash, comparing with checkpoints{-1}.content-hash, and confirming that all three values match. A mismatch indicates either that the document has been modified since the Evidence was generated, or that the Evidence packet does not correspond to the presented document.

9.10.2. Salt Modes for Privacy

Control over how the content hash is computed is afforded by the hash-salt-mode field, making possible privacy-preserving verification scenarios where global verifiability is not desired:

Value	Mode	Hash Computation	Verification
0	unsalted	H(content)	Anyone with document can verify
1	author-salted	H(salt || content)	Author reveals salt to chosen verifiers

Table 3

For salted modes, the salt is provided by the author out-of-band for verification; and confirmation that H(provided-salt) matches salt-commitment is performed by Verifiers before using the salt. Scenarios where the document binding should not be globally verifiable (e.g., unpublished manuscripts, confidential documents) are made possible by Author-Salted mode, affording authors control over who may verify the binding between their Evidence and their document.

9.11. Evidence Content Tiers

PPPP Evidence packets are classified by which optional sections are present. The content tier describes the depth of behavioral and forensic data collected, independent of the attestation assurance level (Section 9.12). Content tiers align with the implementation profiles defined in Section 9.13, which specify the Mandatory-to-Implement requirements for each tier.

The three content tiers are:

CORE (Tier Value 1): Checkpoint chain with VDF proofs, SHA-256 content binding, and RFC 3161 timestamps. Proves temporal ordering and content integrity. See Section 9.13.2 for MTI requirements.

ENHANCED (Tier Value 2): All CORE components plus behavioral entropy (jitter samples), presence challenges, and intra-checkpoint correlation. Adds evidence of interactive authoring behavior. See Section 9.13.3 for MTI requirements.

MAXIMUM (Tier Value 3): All ENHANCED components plus error topology analysis, STARK proofs, CEE binding, absence proofs, and forgery cost bounds. Provides the strongest available evidence for adversarial scenarios. See Section 9.13.4 for MTI requirements.

9.11.1. Tier Selection Guidance

Selection of the minimum tier that meets verification requirements is RECOMMENDED for authors. Higher tiers collect more behavioral data and create larger Evidence packets, which may raise privacy concerns or storage constraints in some deployment scenarios.

Content Tier	Typical Use Cases	Privacy Impact
CORE	Personal notes, internal docs, low-stakes records	Minimal
ENHANCED	Academic submissions, professional reports, business records	Moderate
MAXIMUM	Litigation support, forensic investigation, regulatory compliance	Higher

Table 4

9.11.2. Relationship to Attestation Assurance

Content tier and attestation assurance level (Section 9.12) are orthogonal dimensions. An Evidence packet has both:

* A content tier (CORE/ENHANCED/MAXIMUM) describing what evidence sections are present

* An attestation tier (T1/T2/T3/T4) describing how strongly the evidence is hardware-bound

For example, a MAXIMUM content tier packet may be generated with T1 (software-only) attestation on devices lacking hardware security, while a CORE content tier packet may have T4 (hardware-hardened) attestation when strong device binding is available but behavioral data collection is not desired.

Relying Parties SHOULD establish minimum requirements for both dimensions based on their risk tolerance and regulatory obligations.

9.12. Attestation Assurance Levels

Attestation Assurance Levels define the strength of hardware binding and cryptographic protection for PPPP Evidence packets. This dimension is orthogonal to the content tier (Section 9.11): content tier describes what evidence is collected, while attestation tier describes how strongly that evidence is bound to hardware trust anchors.

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 [I-D.ietf-rats-eat].

Each Evidence packet MUST declare its attestation tier in key 10 of the evidence-packet structure, enabling Verifiers to enforce tier-based acceptance policies. The attestation tier reflects the actual hardware capabilities used during evidence generation, not the maximum capabilities available on the device.

9.12.1. Tier T1: Software-Only

T1 provides baseline evidence generation using pure software implementations without hardware security features.

Attestation Mode: software

Binding Strength: none (no hardware binding) or hmac_local (local

key only)

NIST AAL Mapping: AAL1 - Single-factor authentication equivalent

ISO LoA Mapping: LoA1 - Low confidence in identity

EAT Security Level: unrestricted (0) or restricted (1)

Security Properties:
* VDF timing provides temporal ordering
* Hash chains provide tamper evidence
* Jitter entropy provides behavioral binding
* No hardware root of trust
* Keys stored in software (file system)

Limitations:
* DEVICE_BINDING_NOT_VERIFIED - Device identity not cryptographically bound
* KEY_EXTRACTION_POSSIBLE - Signing keys may be extracted by malware
* NO_HARDWARE_ATTESTATION - Cannot prove hardware integrity

9.12.2. Tier T2: Attested Software

T2 extends T1 with optional hardware attestation hooks when available. The Attesting Environment attempts to use platform security features but degrades gracefully when hardware is unavailable.

Attestation Mode: attested_software

Binding Strength: hmac_local or cryptographic (when hardware available)

NIST AAL Mapping: AAL1-AAL2 - Depending on hardware availability

ISO LoA Mapping: LoA1-LoA2 - Low to medium confidence

EAT Security Level: restricted (1) or secure_restricted (2)

Security Properties:
* All T1 properties

* Hardware attestation when available (opportunistic)
* Platform security APIs used when present
* Keychain/Credential Guard integration on supported platforms

Limitations:
* HARDWARE_OPTIONAL - Hardware features may not be present
* DEGRADED_MODE_POSSIBLE - May fall back to T1 behavior
* VARIABLE_ASSURANCE - Assurance depends on runtime environment

9.12.3. Tier T3: Hardware-Bound

T3 requires hardware security module binding via TPM 2.0 or platform Secure Enclave. Evidence generation MUST fail if hardware attestation is unavailable.

Attestation Mode: hardware_bound

Binding Strength: cryptographic - TPM or Secure Enclave key binding required

NIST AAL Mapping: AAL3 - Hardware cryptographic authenticator

ISO LoA Mapping: LoA3 - High confidence in identity

EAT Security Level: hardware (3)

Security Properties:
* All T2 properties (non-degraded)
* Hardware-protected signing keys (non-exportable)
* Platform integrity measurement (PCR values)
* Device binding cryptographically verified
* Attestation includes hardware identity

Hardware Requirements:
* TPM 2.0 with attestation capability, OR
* Apple Secure Enclave with attestation, OR
* ARM TrustZone with attestation capability

Limitations:
* NO_PUF_BINDING - Physical unclonable function not required
* FIRMWARE_TRUST_REQUIRED - Relies on hardware vendor firmware

9.12.4. Tier T4: Hardware-Hardened

T4 represents maximum attestation strength with discrete TPM, Physical Unclonable Function (PUF) binding, and enclave execution.

Attestation Mode: hardware_hardened

Binding Strength: physical - PUF-derived key binding with TPM attestation

NIST AAL Mapping: AAL3+ - Exceeds AAL3 with physical binding

ISO LoA Mapping: LoA4 - Very high confidence in identity

EAT Security Level: hardware (3) with enhanced claims

Common Criteria Reference: EAL4+ evaluation target equivalent

Security Properties:
* All T3 properties
* PUF-derived entropy binding
* Discrete TPM (not firmware TPM)
* Secure enclave execution for sensitive operations
* Side-channel resistance for timing operations
* Physical tamper evidence

Hardware Requirements:
* Discrete TPM 2.0 (hardware module, not fTPM)
* PUF capability (SRAM PUF or equivalent)
* Secure enclave (SGX, TrustZone, or Secure Enclave)

Limitations:
* LIMITED_DEVICE_SUPPORT - Requires specific hardware
* HIGHER_LATENCY - Additional cryptographic operations

9.12.5. Assurance Level Mapping

The following table summarizes the mapping between PPPP Attestation Tiers and external assurance frameworks. For use case guidance based on content tier, see Section 9.11.1.

PPPP Tier	NIST AAL	ISO LoA	EAT Level	Binding Strength
T1	AAL1	LoA1	0-1	Software-only
T2	AAL1-2	LoA1-2	1-2	Opportunistic hardware
T3	AAL3	LoA3	3	Required TPM/ Enclave
T4	AAL3+	LoA4	3+	Discrete TPM + PUF

Table 5

9.12.6. Relying Party Guidance

Relying Parties SHOULD establish minimum requirements for both attestation tier (this section) and content tier (Section 9.11.1) based on their risk tolerance and regulatory obligations. The following guidance addresses attestation tier requirements specifically:

Accept T1 or higher when:

* Evidence is for personal reference only
* Author reputation provides sufficient trust
* Consequences of forgery are minimal
* Hardware security is impractical for the user population

Require T2 or higher when:

* Evidence supports business decisions
* Multiple parties rely on the evidence
* Moderate financial or reputational risk exists
* Professional standards apply

Require T3 or higher when:

* Legal proceedings may reference the evidence
* Regulatory compliance requires hardware binding
* Non-repudiation is a business requirement
* High-value intellectual property is at stake

Require T4 when:

* Evidence must withstand adversarial forensic analysis
* Litigation is anticipated or ongoing
* Maximum available assurance is mandated by policy
* Sophisticated adversaries with substantial compute resources are anticipated (note: HSM compromise by nation-states is out of scope per Section 32.2.3)

Verifiers MUST include the declared attestation tier in attestation results (WAR files), enabling Relying Parties to enforce tier-based acceptance policies. Verifiers SHOULD also include any attestation-limitations that apply to the Evidence, as these document specific security properties that cannot be claimed at the declared tier.

9.12.7. Behavior When Hardware Unavailable

The Attesting Environment behavior when required hardware is unavailable depends on the configured tier:

T1 Configuration: Hardware availability has no effect. Evidence generation proceeds using software-only implementation.

T2 Configuration: Evidence generation proceeds with available capabilities. The attestation-limitations array MUST include HARDWARE_NOT_AVAILABLE if hardware attestation was attempted but failed. The actual tier achieved MAY be lower than T2 if only software capabilities were available.

T3 Configuration: Evidence generation MUST fail if TPM or Secure Enclave attestation is unavailable. Implementations MUST NOT silently degrade to T2 or T1. An appropriate error code MUST be returned to the caller.

T4 Configuration: Evidence generation MUST fail if discrete TPM, PUF, or enclave capability is unavailable. Implementations MUST NOT silently degrade to lower tiers.

Implementations MUST accurately report the tier achieved, not the tier configured. A T2-configured implementation that lacks hardware MUST report T1 in the evidence packet, not T2.

9.13. Profile Architecture

The PPPP specification defines three implementation profiles that establish Mandatory-to-Implement (MTI) requirements for interoperability. Each profile represents a coherent set of features that implementations MUST support to claim conformance at that level. Profile declarations are carried in key 9 of the evidence-packet structure as specified in the companion CDDL schema [I-D.condrey-rats-pop-schema].

Implementation profiles define what features an implementation MUST support. This is related to, but distinct from:

* Evidence Content Tiers (Section 9.11): describe what optional sections are present in a given Evidence packet

* Attestation Assurance Levels (Section 9.12): describe hardware binding strength for a given Evidence packet

A CORE profile implementation may generate packets at any content tier (by including optional features), while an ENHANCED profile implementation MUST be capable of generating ENHANCED content tier packets.

9.13.1. Profile Identifiers

Each profile is identified by a URN in the IETF RATS namespace with the following format:

urn:ietf:params:rats:pop:profile:<name>

The registered profile URNs are:

Profile	Tier Value	URN
CORE	1	urn:ietf:params:rats:pop:profile:core
ENHANCED	2	urn:ietf:params:rats:pop:profile:enhanced
MAXIMUM	3	urn:ietf:params:rats:pop:profile:maximum

Table 6

9.13.2. CORE Profile

The CORE profile establishes the minimum viable implementation for PPPP interoperability. All implementations claiming PPPP conformance MUST implement at least the CORE profile. The security guarantees provided by CORE are:

* Temporal ordering: VDF proofs establish minimum elapsed time between checkpoints with cryptographic assurance.

* Content integrity: SHA-256 hash binding ensures tamper-evidence for the attested document.

* External anchoring: RFC 3161 timestamps provide independent temporal witnesses from trusted third parties.

The following features are Mandatory-to-Implement for CORE:

Feature ID	Feature Name	Description
1	vdf-iterated-sha256	Iterated SHA-256 VDF construction per Section 24.2.2
2	content-binding	SHA-256 content hash binding per Section 9.10.1
3	external-anchor-rfc3161	RFC 3161 timestamp anchor support
4	checkpoint-chain	Hash-linked checkpoint chain per Section 9.9
5	cose-sign1	COSE_Sign1 packet signature

Table 7

9.13.3. ENHANCED Profile

The ENHANCED profile adds behavioral entropy capture and correlation analysis to the CORE features. Implementations claiming ENHANCED conformance MUST implement all CORE features plus the ENHANCED MTI features. The additional security guarantees provided by ENHANCED are:

* Behavioral entropy: Jitter-based entropy capture provides evidence of interactive authoring behavior in the creation process.

* Intra-checkpoint correlation (C_intra): Statistical correlation between timing patterns and content evolution within checkpoints.

* Cognitive load indicators: Metrics derived from typing patterns that reflect human cognitive processing characteristics.

The following features are Mandatory-to-Implement for ENHANCED (in addition to all CORE features):

Feature ID	Feature Name	Description
50	behavioral-entropy	Jitter-based behavioral entropy per Section 10
51	c-intra-correlation	Intra-checkpoint Spearman correlation
52	cognitive-load	Cognitive load indicators derived from timing
53	presence-challenges	Human presence verification challenges
54	keystroke-jitter	Keystroke timing jitter capture

Table 8

9.13.4. MAXIMUM Profile

The MAXIMUM profile provides the strongest available evidence through comprehensive behavioral analysis, cryptographic proofs, and hardware attestation. Implementations claiming MAXIMUM conformance MUST implement all CORE and ENHANCED features plus the MAXIMUM MTI features. The additional security guarantees provided by MAXIMUM are:

* Error topology analysis: Fractal pattern analysis of editing errors that distinguishes human error patterns from automated generation.

* STARK proofs: Succinct transparent arguments of knowledge for efficient verification of complex evidence structures.

* Cryptographic Entropy Entanglement (CEE): VDF outputs entangled with behavioral entropy to prevent backdating attacks.

* Hardware attestation: TPM 2.0 or Secure Enclave binding for device-level trust anchoring.

The following features are Mandatory-to-Implement for MAXIMUM (in addition to all CORE and ENHANCED features):

Feature ID	Feature Name	Description
100	error-topology	Fractal error pattern analysis per Section 14
101	stark-proofs	STARK-based verification proofs
102	cee-binding	Cryptographic Entropy Entanglement per Section 18
103	absence-proofs	Negative evidence claims per Section 25
104	forgery-cost-bounds	Economic attack cost analysis per Section 26
105	hardware-attestation	TPM/Secure Enclave binding

Table 9

9.13.5. Profile Declaration Structure

Evidence packets MAY include a profile declaration in key 9 of the evidence-packet structure. The declaration specifies the profile tier and URI, with optional indication of features enabled beyond the MTI requirements. The CDDL [RFC8610] structure is:

```
profile-declaration = {
        1 => profile-tier,              ; tier (1=core, 2=enhanced, 3=maximum)
        2 => profile-uri,               ; URN identifier
        ? 3 => [+ feature-id],          ; enabled-features (beyond MTI)
        ? 4 => tstr,                    ; implementation-id
}

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

profile-uri = tstr .regexp "urn:ietf:params:rats:pop:profile:(core|enhanced|maximum)"
```

The enabled-features array (key 3) lists feature IDs that are implemented beyond the MTI requirements for the declared tier. This allows CORE implementations to indicate support for specific ENHANCED

or MAXIMUM features without claiming full conformance to those tiers. The implementation-id (key 4) is an opaque string identifying the software that generated the Evidence packet, useful for debugging and ecosystem analysis but carrying no normative weight.

9.13.6. Verification Behavior

Verifiers MUST handle Evidence packets according to the following rules based on the presence or absence of profile declarations:

9.13.6.1. Profile Declaration Present

When key 9 (profile-declaration) is present in the evidence-packet, Verifiers MUST:

1. Validate that the profile-uri corresponds to a known profile.
2. Verify that all MTI features for the declared tier are present in the Evidence packet with valid data.
3. If MTI validation fails, the Verifier MUST reject the packet with error code PROFILE_INCOMPLETE.
4. If MTI validation succeeds, the Verifier MAY rely on the security guarantees associated with the declared profile tier.

9.13.6.2. Profile Declaration Absent

When key 9 is absent from the evidence-packet, Verifiers MUST apply defensive processing:

1. The Verifier MUST NOT assume any specific profile tier.
2. The Verifier SHOULD attempt to infer the effective tier by examining which structures are present in the packet.
3. The inferred tier MUST be reported in the attestation-result with caveat PROFILE_INFERRED indicating that the profile was not explicitly declared by the Attester.
4. Relying Parties SHOULD treat inferred profiles with lower confidence than explicitly declared profiles.

9.13.6.3. Unknown Profile URI

When the profile-uri value is not recognized by the Verifier:

1. The Verifier MUST NOT reject the packet solely because the profile URI is unknown.

2. The Verifier SHOULD process the packet as if no profile were declared, applying the inference rules from Section 9.13.6.2.

3. The attestation-result MUST include caveat PROFILE_UNKNOWN with the unrecognized URI value.

This forward-compatibility behavior allows future profile extensions without breaking existing Verifiers while ensuring that Relying Parties are informed when unfamiliar profiles are encountered.

9.13.7. MTI Summary

The following table summarizes the Mandatory-to-Implement requirements across all profiles. An "M" indicates the feature is mandatory for that profile tier; an "O" indicates the feature is optional but MAY be declared in the enabled-features array.

Feature ID	Feature Name	CORE	ENHANCED	MAXIMUM
1	vdf-iterated-sha256	M	M	M
2	content-binding	M	M	M
3	external-anchor-rfc3161	M	M	M
4	checkpoint-chain	M	M	M
5	cose-sign1	M	M	M
50	behavioral-entropy	O	M	M
51	c-intra-correlation	O	M	M
52	cognitive-load	O	M	M
53	presence-challenges	O	M	M
54	keystroke-jitter	O	M	M
100	error-topology	O	O	M
101	stark-proofs	O	O	M
102	cee-binding	O	O	M
103	absence-proofs	O	O	M
104	forgery-cost-bounds	O	O	M
105	hardware-attestation	O	O	M

Table 10

9.14. Attestation Result Structure

The attestation-result structure contains the Verifier’s assessment of an Evidence packet. It implements a witnessd-specific profile of EAR (Entity Attestation Results) as defined in [I-D.ietf-rats-ear].

```
attestation-result = {
        1 => uint,                       ; version
        2 => uuid,                       ; reference-packet-id
        3 => pop-timestamp,              ; verified-at
        4 => forensic-assessment,        ; verdict
        5 => confidence-millibits,       ; confidence (0-1000 = 0.0-1.0)
        6 => [+ result-claim],           ; verified-claims
        7 => cose-signature,             ; verifier-signature
        ? 8 => tstr,                     ; verifier-identity
        ? 9 => verifier-metadata,        ; additional info
        ? 10 => [+ tstr],                ; caveats
        ? 11 => source-consistency-analysis, ; Verifier's interpretation
        * tstr => any,                   ; extensions
}

source-consistency-analysis = {
        1 => tstr,                       ; detected-pattern
        2 => ratio-millibits,            ; aggregate-consistency (0-1000)
        ? 3 => [+ uint],                ; deviation-checkpoint-indices
        ? 4 => tstr,                    ; verifier-policy-id
}

; Fixed-point type definitions (see schema spec for details)
confidence-millibits = uint .le 1000   ; 0-1000 representing 0.000-1.000
ratio-millibits = uint .le 1000        ; generic 0.0-1.0 ratio
entropy-decibits = uint .le 640        ; 0-640 representing 0.0-64.0 bits
cost-microdollars = uint               ; USD * 1,000,000
duration-ms = uint                     ; milliseconds
p-value-centibits = uint .le 10000     ; p-values with 4 decimal precision
```

9.14.1. Verdict Field

The verdict (key 4) is the Verifier's overall forensic assessment using the forensic-assessment enumeration:

Value	Assessment	Meaning
0	not-assessed	Verification incomplete or not attempted
1	source-consistent	Evidence chain shows consistent generative process throughout
2	source-consistent-partial	Evidence chain shows consistency with minor deviations
3	inconclusive	Insufficient evidence to characterize source consistency
4	source-transition-detected	Evidence chain contains measurable process transitions
5	source-inconsistent	Evidence chain shows significant process inconsistency

Table 11

IMPORTANT: The verdict characterizes the consistency of the evidence chain, not the identity or nature of the author. A verdict of "source-transition-detected" means the behavioral metrics changed measurably at specific checkpoints. What caused that change - a tool switch, a collaborator, fatigue, or something else - is not determined by the Verifier. The Relying Party applies domain-specific policy to decide whether the observed pattern is acceptable.

9.14.2. Confidence Score

The confidence-score (key 5) is an unsigned integer in millibits (0-1000) representing the Verifier’s confidence in the verdict. Divide by 1000 to convert to the 0.0-1.0 range:

* 0 - 300: Low confidence (limited evidence)

* 300 - 700: Moderate confidence (typical evidence)

* 700 - 1000: High confidence (strong evidence)

The confidence score incorporates:

* Evidence tier (higher tiers increase confidence ceiling)

* Segment chain completeness

* Entropy sufficiency in jitter bindings

* VDF calibration attestation presence

* External anchor confirmations

9.14.3. Verified Claims

The verified-claims array (key 6) contains individual claim verification results:

```
result-claim = {
        1 => uint,                      ; claim-type
        2 => bool,                      ; verified
        ? 3 => tstr,                    ; detail
        ? 4 => confidence-level,        ; claim-confidence
}
```

The claim-type values correspond to the absence-claim-type enumeration, enabling direct mapping between Evidence claims and Attestation Result verification outcomes.

9.14.4. Verifier Signature

The verifier-signature (key 7) is a COSE_Sign1 signature over the Attestation Result payload (fields 1-6 plus any optional fields 8-10). This signature:

* Authenticates the Verifier identity

* Ensures integrity of the Attestation Result

* Enables Relying Parties to verify the result came from a trusted Verifier

9.14.5. Caveats

The caveats array (key 10) documents limitations and warnings that Relying Parties should consider:

* "No hardware attestation available"

* "External anchors pending confirmation"

* "Jitter entropy below recommended threshold"

* "Author declares AI tool usage"

Verifiers MUST include appropriate caveats when the Evidence has known limitations. Relying Parties SHOULD review caveats before making trust decisions.

9.15. CBOR Encoding

Both Evidence packets and Attestation Results use CBOR (Concise Binary Object Representation) encoding per RFC 8949.

9.15.1. Semantic Tags

Top-level structures use semantic tags for type identification:

Tag	Hex	ASCII	Structure
1347571280	0x50505020	"PPPP"	tagged-evidence-packet
1463894560	0x57415220	"WAR "	tagged-attestation-result

Table 12

These tags enable format detection without external metadata. Parsers can identify the packet type by examining the leading tag value.

9.15.2. Key Encoding Strategy

The schema uses a dual key encoding strategy for efficiency and extensibility:

Integer Keys (1-99): Reserved for core protocol fields defined in this specification. Provides compact encoding and enables efficient parsing.

String Keys: Used for vendor extensions, application-specific fields, and future protocol extensions before standardization. Provides self-describing field names at the cost of encoding size.

Example size comparison for a field named "forensics":

```
Integer key (11):     1 byte  (0x0B)
String key ("forensics"): 10 bytes (0x69666F72656E73696373)
```

For a typical Evidence packet with dozens of fields, integer keys reduce packet size by 20-40%.

9.15.3. Deterministic Encoding

Evidence packets MUST use deterministic CBOR encoding (RFC 8949 Section 4.2) (RFC 8949 Section 4.2) to enable:

* Byte-exact reproduction of packets for signature verification
* Consistent hashing for cache and deduplication purposes
* Simplified debugging and comparison

Deterministic encoding requirements:

* Map keys sorted in bytewise lexicographic order
* Integers encoded in minimal representation
* Floating-point values canonicalized

9.16. EAT Profile

This specification defines an EAT (Entity Attestation Token) profile for Proof of Process evidence. The profile URI is:

https://example.com/rats/eat/profile/pop/1.0

9.16.1. Custom EAT Claims

The following custom claims are proposed for IANA registration upon working group adoption:

Claim Name	Type	Description
pop-forensic-assessment	uint	forensic-assessment enumeration value
pop-presence-score	uint (millibits)	Presence challenge pass rate (0-1000)
pop-evidence-tier	uint	Evidence tier (1-4)
pop-ai-composite-score	uint (millibits)	AI indicator composite score (0-1000)

Table 13

9.16.2. AR4SI Trustworthiness Extension

The Attestation Result includes a proposed extension to the AR4SI ([I-D.ietf-rats-ar4si]) trustworthiness vector:

behavioral-consistency: -1..3
-1 = no claim
0 = behavioral evidence inconsistent with human authorship
1 = behavioral evidence inconclusive
2 = behavioral evidence consistent with human authorship
3 = behavioral evidence strongly indicates human authorship

This extension enables integration of witnessd Attestation Results with broader trustworthiness assessment frameworks.

The following table provides guidance for mapping PPPP forensic-assessment verdicts to AR4SI behavioral-consistency values:

PPPP Verdict	AR4SI behavioral-consistency	Rationale
not-assessed (0)	-1 (no claim)	Verification not performed
source-consistent (1)	2 or 3	2 for moderate confidence, 3 for high confidence
source-consistent-partial (2)	2	Consistency with acceptable deviations
inconclusive (3)	1	Insufficient evidence for determination
source-transition-detected (4)	1	Transitions detected but not classified
source-inconsistent (5)	0	Evidence inconsistent with single-source composition

Table 14

Note: The mapping from PPPP verdicts to AR4SI values depends on the confidence score and Relying Party policy. The table above provides default guidance; implementations MAY adjust based on domain-specific requirements.

9.17. Security Considerations

9.17.1. Tamper-Evidence vs. Tamper-Proof

The evidence model provides tamper-EVIDENCE, not tamper-PROOF:

* Tamper-evident:

Modifications to Evidence packets are detectable through cryptographic verification. The hash chain, VDF entanglement, and HMAC bindings ensure that any alteration invalidates the Evidence.

* Not tamper-proof:

  An adversary with sufficient resources can fabricate Evidence by investing the computational time required by VDF proofs and generating plausible behavioral data. The forgery-cost-section quantifies this investment.

Relying Parties should understand this distinction when making trust decisions.

9.17.2. Independent Verification

Evidence packets are designed for independent verification:

* All cryptographic proofs are included in the packet
* Verification requires no access to the original device
* Verification requires no network access (except for external anchor validation)
* Multiple independent Verifiers can appraise the same Evidence

This property enables adversarial verification: a skeptical Relying Party can verify Evidence without trusting the Attester's infrastructure.

9.17.3. Privacy by Construction

The evidence model enforces privacy through structural constraints:

* No content storage:

  Evidence contains hashes of document states, not content. The document itself is never included in Evidence packets.

* No keystroke capture:

  Individual characters typed are not recorded. Timing intervals are captured without association to specific characters.

* Aggregated behavioral data:

Raw timing data is aggregated into histograms before inclusion in Evidence. Optional raw interval disclosure is user-controlled.

* No screenshots or screen recording:

  Visual content is never captured by the Attesting Environment.

9.17.4. Attesting Environment Trust

The evidence model assumes a minimally trusted Attesting Environment:

* Chain-verifiable claims (absence-claim-types 1-15):

  Can be verified from Evidence alone without trusting the AE beyond basic data integrity.

* Monitoring-dependent claims (absence-claim-types 16-63):

  Require trust that the AE accurately reported monitored events. The ae-trust-basis field documents these assumptions.

Hardware attestation (hardware-section) increases AE trust by binding Evidence to verified hardware identities.

10. Jitter Seal: Captured Behavioral Entropy

In this section, the Jitter Seal mechanism is delineated, a novel contribution to behavioral evidence within the RATS [RFC9334] architecture that binds captured timing entropy to the segment chain using HMAC-SHA256 [RFC2104] [RFC6234] commitments. Unlike injected entropy (random delays added by software that could be regenerated if the seed is known), actual measured timing from human input events is committed to by captured entropy, with the commitment computed using SHA-256 before histogram aggregation and bound to the VDF chain [Pietrzak2019] [Wesolowski2019] through the jitter-binding structure defined in CDDL [RFC8610]. This creates evidence encoded in CBOR [RFC8949] that cannot be regenerated without access to the original input stream, because the entropy-commitment fixes the raw timing data before any statistical summarization that might allow reconstruction.

10.1. Design Principles

A fundamental limitation in existing attestation frameworks, including the base RATS architecture, is addressed by the Jitter Seal: the inability to distinguish evidence generated during genuine human interaction from evidence reconstructed after the fact. By cryptographically committing to captured timing entropy using SHA-256 before histogram aggregation, and binding this commitment to the VDF chain via HMAC, evidence is produced that bears an indelible relationship to the moment of its creation. Three key design principles guide the Jitter Seal mechanism: Captured vs. Injected Entropy distinguishes between injected entropy (random delays inserted by software that can be regenerated if the seed is known) and captured entropy that commits to timing measurements via SHA-256 that existed only at the moment of observation, meaning an adversary cannot regenerate captured entropy without access to the original input stream; Commitment Before Observation ensures that the entropy-commitment is computed using SHA-256 and bound to the segment chain via HMAC before the histogram summary is finalized, preventing an adversary from crafting statistics that match a predetermined commitment encoded in CBOR; and Privacy-Preserving Aggregation ensures that raw timing intervals are aggregated into histogram buckets defined in the CDDL schema, preserving statistical properties needed for entropy verification while preventing reconstruction of the original keystroke sequence, with raw intervals optionally disclosed per the user’s privacy preferences.

10.2. Jitter Binding Structure

Appearance of the jitter-binding structure in each checkpoint is mandated by this specification, with five fields being contained therein that together provide cryptographic binding between the behavioral entropy captured during authoring and the segment chain protected by SHA-256 hash linkage and VDF proofs. The structure is encoded using CBOR per the CDDL schema below, and employs HMAC-SHA256 for binding integrity that prevents jitter data from being transplanted between checkpoints.

```
    jitter-binding = {
        1 => hash-value,           ; entropy-commitment
        2 => [+ entropy-source],   ; sources
        3 => jitter-summary,       ; summary
        4 => bstr .size 32,        ; binding-mac
        ? 5 => [+ uint],           ; raw-intervals (optional)
        ? 6 => checkpoint-behavioral,     ; Per-checkpoint behavioral measurements
    }

    checkpoint-behavioral = {
        1 => ratio-millibits,      ; spectral-slope (pink noise alpha)
        2 => ratio-millibits,      ; hurst-exponent
        3 => ratio-millibits,      ; intra-checkpoint-consistency
        ? 4 => uint,               ; edit-operation-count
        ? 5 => uint,               ; composition-operation-count
        ? 6 => uint,               ; revision-operation-count
        ? 7 => uint,               ; structural-operation-count
    }
```

10.2.1. Entropy Commitment (Key 1)

A cryptographic hash of the raw timing intervals concatenated in observation order is constituted by the entropy-commitment, computed as H(interval{0} || interval{1} || ... || interval{n}) where H denotes the hash algorithm specified in the hash-value structure with SHA-256 being RECOMMENDED. Each interval is encoded as a 32-bit unsigned integer representing milliseconds, conforming to the CBOR unsigned integer encoding (major type 0). Computation of this SHA-256 commitment BEFORE the histogram summary is mandated by this specification, thereby ensuring that the raw data cannot be manipulated to match a desired statistical profile after the commitment is fixed. This ordering constraint is critical to the security of the Jitter Seal mechanism: once the entropy-commitment is computed using SHA-256 and bound to the VDF input, the raw timing data is cryptographically fixed even though only the aggregated histogram appears in the final CBOR encoded Evidence packet.

10.2.2. Entropy Sources (Key 2)

Identification of which input modalities contributed to the captured entropy is accomplished by the sources array:

Value	Source	Description
1	keystroke-timing	Inter-key intervals from keyboard input
2	pause-patterns	Gaps between editing bursts (>2 seconds)
3	edit-cadence	Rhythm of insertions/ deletions over time
4	cursor-movement	Navigation timing within document
5	scroll-behavior	Document scrolling patterns
6	focus-changes	Application focus gain/loss events

Table 15

Inclusion of at least one source is REQUIRED by implementations conforming to this RATS profile, with the source array encoded in CBOR per the CDDL schema. The highest entropy density is afforded by the keystroke-timing source (1), and its inclusion SHOULD be ensured when keyboard input is available, as this source provides the finest-grained timing measurements that contribute most significantly to the entropy-commitment computed using SHA-256. Multiple sources may be combined to increase the total entropy density and make possible verification even when some input modalities are unavailable, with the estimated-entropy-bits calculation aggregating Min-Entropy (H_min) across all contributing sources.

10.2.3. Jitter Summary (Key 3)

Verifiable statistics without exposure of raw timing data are afforded by the jitter-summary structure, encoded in CBOR per the CDDL schema below, enabling Verifiers to assess entropy sufficiency per the RATS architecture without accessing the privacy-sensitive raw intervals.

```
    jitter-summary = {
        1 => uint,                  ; sample-count
        2 => [+ histogram-bucket], ; timing-histogram
        3 => entropy-decibits,      ; estimated-entropy (decibits, /10 for bits)
        ? 4 => [+ anomaly-flag],    ; anomalies (if detected)
    }

    histogram-bucket = {
        1 => uint,                  ; lower-bound-ms
        2 => uint,                  ; upper-bound-ms
        3 => uint,                  ; count
    }
```

Calculation of the estimated-entropy-bits field is accomplished using Shannon entropy over the histogram distribution:

```
    H = -sum(p[ij] * log2(p[ij])) (conditional probabilities) for all buckets where p[i]
> 0
    p[i] = count[i] / total_samples
```

The following bucket boundaries (in milliseconds) are RECOMMENDED: 0, 50, 100, 200, 500, 1000, 2000, 5000, +infinity. The typical range of human typing and pause behavior is captured by these boundaries, having been determined empirically through analysis of diverse authoring sessions.

10.2.4. Binding MAC (Key 4)

Cryptographic binding of the jitter data to the segment chain is accomplished by the binding-mac:

```
        binding-mac = HMAC-SHA256(
            key = checkpoint-chain-key,
            message = entropy-commitment ||
                      CBOR(sources) ||
                      CBOR(summary) ||
                      prev-tree-root
        )
```

The following properties are ensured by this HMAC-SHA256 binding within the RATS architecture: transplantation of jitter data between checkpoints is prevented because the prev-tree-root included in the HMAC input fixes the binding to a specific position in the SHA-256 hash chain; modification of jitter data without invalidating the segment chain is prevented because the binding-mac is included in the tree-root computation; and preservation of the temporal ordering of jitter observations is enforced because the VDF entanglement includes the entropy-commitment. These properties, taken together, make possible strong guarantees about the authenticity and integrity of the captured behavioral entropy, with any tampering detectable through cryptographic verification using SHA-256 and HMAC.

10.2.5. Raw Intervals (Key 5, Optional)

Inclusion of the raw-intervals array for enhanced verification is permitted but not required. When present, the following capabilities are afforded to verifiers: recomputation of the entropy-commitment with verification that it matches, recomputation of the histogram with consistency verification, and performance of statistical analysis beyond the histogram. As a privacy consideration, it should be noted that raw intervals may constitute biometric-adjacent data; this concern is addressed in Section 22.

10.3. Hardware Assurance Requirements

High-assurance evidence (Process Score >= 0.9) requires specific hardware capabilities at the Attesting Environment:

* TPM 2.0: MUST support PCR banks with SHA-256 and provide signed quotes (TPM2_Quote) binding evidence to platform state.

* Secure Enclave: MUST provide hardware-backed key storage and monotonic counter operations.

* Certificate Validation: Verifiers MUST validate the Attester’s Attestation Key against the manufacturer’s Root CA.

At higher assurance tiers (T3-T4), the hardware anchors evidence generation to specific physical silicon, preventing migration of evidence generation to faster or different hardware. At lower assurance tiers (T1-T2), evidence generation proceeds in software with reduced confidence scores. Evidence metadata MUST indicate the hardware assurance level so that Verifiers can adjust confidence accordingly.

10.4. Attestation Nonce Binding

For hardware-attested evidence (T3-T4 tiers), a 32-byte cryptographically random attestation nonce MUST be generated at session initialization using a cryptographically secure random number generator. This attestation_nonce serves distinct purposes from the verifier_nonce:

* TPM Quote Binding:

  The attestation_nonce is passed to TPM2_Quote operations, binding hardware attestation to this specific evidence session. This prevents replay of TPM quotes from previous sessions.

* TEE Session Binding:

  For Secure Enclave implementations, the attestation_nonce binds enclave attestation reports to the current session.

* Session Uniqueness:

  The attestation_nonce ensures each evidence generation session produces cryptographically distinct hardware attestations, even for identical content.

The attestation_nonce MUST be included in the evidence packet for hardware-attested evidence, enabling Verifiers to confirm the TPM quote or TEE attestation report matches the claimed session.

10.5. Timing Value Clipping

To prevent outlier timing samples from leaking sensitive behavioral information, all timing values are clipped to a normative range [0, 5000ms]. Values exceeding this range are coerced to the boundary. This bounds the sensitivity of timing data and provides consistent input ranges for behavioral analysis across all authoring environments.

10.6. Software-Only Mode

Implementations lacking access to a TEE or TPM operate in software-only mode. Evidence produced in this mode MUST be flagged with a maximum Process Score of 0.7. Verifiers SHOULD treat software-only evidence as a behavioral claim rather than a hardware-attested proof of platform binding. Software-only evidence still provides VDF temporal ordering, hash chain integrity, and behavioral entropy - the limitation is that these computations are not bound to specific hardware.

11. Behavioral Entropy Analysis

The Attesting Environment computes behavioral entropy metrics locally from captured input timing intervals. These metrics characterize the statistical properties of the authoring process without recording keystroke content. All analysis is performed on the local device; no timing data leaves the Attesting Environment except as aggregated statistical summaries committed via HMAC.

11.1. Timing Spectral Analysis

Human motor systems exhibit characteristic spectral properties in inter-keystroke timing intervals. The Attesting Environment computes the power spectral density of timing intervals and derives two metrics:

Pink noise slope (alpha): Human typing typically exhibits 1/f noise where power density inversely scales with frequency, with slope alpha between 0.8 and 1.2. Mechanical injection tends toward white noise (alpha near 0) or periodic patterns (discrete frequency peaks).

Hurst exponent (H): Computed via Rescaled Range (R/S) analysis or Detrended Fluctuation Analysis (DFA) of timing intervals. Human motor systems typically show $H \in [0.55, 0.85]$, indicating long-range temporal dependence characteristic of natural behavioral processes. Values near 0.5 indicate white noise (rejection: likely synthetic or random input). Values approaching 1.0 indicate highly deterministic sequences (rejection: likely automated or mechanical generation). Implementations MUST reject timing sequences with H outside the [0.55, 0.85] validation range.

These metrics are included in the behavioral entropy summary at each checkpoint. The Verifier evaluates them as informational signals contributing to the source consistency assessment, not as binary pass/fail gates.

11.2. Intra-Session Consistency

The Attesting Environment evaluates statistical stability of authoring behavior across checkpoints within a session. Each checkpoint’s behavioral digest captures a timing distribution. The Attesting Environment computes the statistical distance (KL Divergence) between each checkpoint’s distribution and the cumulative session baseline.

The Intra-Session Consistency score (C_intra) is high when timing distributions remain within a stable statistical cluster. Significant divergence (exceeding a configurable threshold) indicates a potential change in the generative process. This divergence is recorded in the evidence chain as a source consistency transition event - the Attesting Environment does not interpret the cause of the divergence.

11.3. Temporal Evolution of Behavioral Metrics

Interactive authoring sessions exhibit characteristic temporal evolution over extended durations. The Attesting Environment tracks variance evolution across checkpoints:

* Timing variance typically increases over multi-hour sessions due to motor fatigue.

* The Hurst exponent may drift toward 0.5 as fatigue reduces long-range motor correlation.

* Error rate (ratio of deletions to insertions in a sliding window) typically increases over time.

These evolution patterns are included in the evidence chain as informational metrics. The absence of temporal evolution in a long session is a source consistency signal - not proof of fabrication, but a measurable characteristic that the Relying Party can evaluate in context.

12. Clock Integrity

To harden against clock spoofing at the kernel or hypervisor level, the Attesting Environment employs Clock-Entropy Entanglement (CEE). Rather than reporting raw timestamps, the Attesting Environment generates an entropic pulse for each checkpoint:

```
P = HMAC(K_session, DST_CLOCK || timestamp || hardware_entropy)
```

By binding the monotonic timestamp to non-deterministic hardware entropy (where available), the protocol ensures that clock manipulation produces cryptographic mismatches in the VDF chain. In software-only mode, system-provided entropy sources are used with correspondingly reduced assurance.

13. Privacy-Preserving Timing Protection

To prevent cross-session correlation of behavioral timing patterns, the Attesting Environment applies a session-specific non-linear transformation to timing metrics before committing them to the evidence chain:

```
T_committed = Transform(T_measured, K_session)
```

The transformation preserves the internal statistical properties required for source consistency analysis (relative distributions, spectral characteristics, evolution patterns) while altering absolute values that could serve as a biometric fingerprint. The Verifier, possessing the session key derivation material, can evaluate consistency properties without recovering the original timing values.

14. Error Topology and Fractal Invariants

The "Error Topology" invariant captures the physio-biological signature of human mistakes. Unlike mathematical randomness, human typos follow a "Fractal Error Pattern" comprising four phases: Physical Mistake (e.g., adjacent key strike, [Grudin1983]) -> Cognitive Recognition Gap (avg 150ms saccade-feedback loop, [Rayner1998]) -> Reflexive Burst (rapid backspacing) -> Correction.

The Evidence includes a ZK-Proof (STARK) attesting that the distribution of deletions and corrections satisfy a composite biological score S >= 0.75, derived from Spearman correlation of gaps to complexity (rho_gap), the Hurst exponent (H) for self-similar persistence [Mandelbrot1982], and the physical adjacency ratio (adj_phys):

```
S = 0.4*rho_gap + 0.4*H + 0.2*adj_phys >= 0.75
```

This topology proves that the editing process adheres to biological motor-skill constraints, which are computationally expensive for non-biological agents to simulate within the sequential constraints of the VDF chain.

15. Cognitive Load and Semantic Correlation

The Behavioral Consistency invariant is grounded in the neurobiological constraint of Cognitive Load Delays (CLD). Human typing exhibits 50-300ms inter-keystroke spikes when processing complex or rare tokens [Kushniruk1991].

Verification of human origin is achieved through the correlation of Information Density (D) and Timing Density. To protect author privacy, timing histograms are ZK-Private inputs; only the cryptographic commitment is revealed. Segments with LZ Complexity < 0.2 are excluded from scoring (Complexity Gating) to prevent Signal Starvation.(tau) per segment i:

D_i = LZ_Complexity(s_i) / |s_i|

Where s_i is the segment content, and D is the normalized compression ratio (zlib) of the deterministic Lempel-Ziv (LZ) complexity algorithm (RFC 1951). The Jitter Seal reports ranked delays (tau_i) in quantized buckets. The Evidence is valid if and only if the Spearman rank correlation satisfies:

rho(D, tau) = corr(rank(D), rank(tau)) >= theta = 0.7

This mechanism ensures that "Semantic Spikes" in timing align with spikes in linguistic complexity.

16. Zero-Knowledge Cognitive Load Verification

The Spearman correlation verification described in Section 15 requires access to both the timing histogram (pause durations per segment) and the complexity histogram (LZ compression ratios per segment) to compute rho. This requirement creates a tension with the content-agnosticism principle: while the timing data is already ZK-private via bucket commitments, revealing the complexity histogram to a Verifier would disclose information about the document’s linguistic structure, potentially enabling reconstruction of content patterns or stylometric fingerprinting.

16.1. Problem Statement

The core challenge is proving that a correlation exists between two private datasets without revealing either dataset:

* The pause histogram (tau) captures inter-keystroke intervals aggregated into timing buckets. This data is privacy-sensitive as it may constitute biometric-adjacent behavioral information.

* The complexity histogram (D) captures normalized LZ compression ratios per segment. While derived from content, revealing the distribution exposes structural information about the document.

* The Verifier needs assurance that rho(D, tau) >= 0.7 without learning either D or tau individually.

Zero-knowledge proofs resolve this tension by enabling the Attester to prove the correlation relationship holds while revealing only the Boolean outcome (satisfied/not satisfied) and a confidence band.

16.2. SNARK-Based Verification (Maximum Tier)

For Maximum tier Evidence, a Succinct Non-interactive ARgument of Knowledge (SNARK) is employed to prove the correlation claim. The circuit encodes:

Public inputs:
- threshold: 700 (representing rho >= 0.7)
- segment_count: n
- pause_commitment: H(tau_1, ..., tau_n)
- complexity_commitment: H(D_1, ..., D_n)

Private inputs (witness):
- tau[]: pause histogram values
- D[]: complexity histogram values

Circuit constraints:
1. H(tau[]) == pause_commitment
2. H(D[]) == complexity_commitment
3. rank(tau[]) computed correctly
4. rank(D[]) computed correctly
5. spearman_rho(rank(tau), rank(D)) >= threshold

The SNARK proof is approximately 200-300 bytes (for Groth16) or 1-2 KB (for PLONK/STARK variants) and verifies in constant time. The public-parameters-hash binds the proof to a specific circuit version, enabling algorithm agility while preventing substitution attacks.

SNARK verification is computationally efficient for Verifiers (milliseconds) but proof generation requires significant Attester resources (seconds to minutes depending on segment count). This asymmetry is acceptable for the Maximum tier where the strongest assurance is required.

16.3. Pedersen Commitment Fallback (Enhanced Tier)

For Enhanced tier Evidence where SNARK proving infrastructure may not be available, a Pedersen commitment scheme with Bulletproof range proofs provides weaker but still meaningful ZK assurance:

commitment-rho: Pedersen commitment to the computed Spearman rho value, C = g^rho * h^r where r is the nonce.

commitment-pause-histogram: Pedersen commitment to the pause

histogram vector, binding the Attester to specific timing data.

commitment-complexity-histogram: Pedersen commitment to the complexity histogram vector, binding the Attester to specific structural data.

range-proofs: Bulletproof range proofs demonstrating that: (a) rho falls within [-1.0, 1.0], (b) rho >= threshold (0.7), (c) all histogram values are non-negative.

consistency-binding-proof: Proof that the committed rho was correctly computed from the committed histograms using Spearman's formula.

The Pedersen approach requires larger proofs (1-5 KB depending on segment count) and provides computational hiding rather than perfect zero-knowledge. However, it uses well-established elliptic curve cryptography without trusted setup requirements.

16.4. What ZK Proofs Do and Do Not Claim

Zero-knowledge cognitive load proofs establish the following:

Correlation Verified (PROVEN): The Spearman rank correlation between the Attester's private pause histogram and private complexity histogram meets or exceeds the threshold. This is cryptographically bound.

Data Consistency (PROVEN): The committed histograms match those used in correlation computation. The Attester cannot claim correlation with fabricated data without invalidating the proof.

Confidence Band (DOCUMENTED): Statistical confidence intervals account for sample size effects and provide Verifiers with uncertainty bounds.

Zero-knowledge cognitive load proofs explicitly do NOT claim:

Cognitive Origin: Correlation is consistent with but does not prove cognitive engagement. The proof establishes a statistical relationship, not a causal mechanism. Sophisticated simulation could potentially produce correlated timing, though at significant computational cost (see Section 26).

Human Authorship: No claim is made that a human (as opposed to a sufficiently sophisticated automation) produced the input. The proof documents observable correlation, not its source.

Content Quality: The proof says nothing about the semantic quality, originality, or value of the document content. It attests only to process characteristics.

Absence of Assistance: The proof does not exclude the possibility that the author used tools, references, or other aids during creation. It documents the observable editing process, not the author's cognitive sources.

16.5. Evidence Tier Mapping

The correlation-algorithm enumeration maps to evidence tiers:

Algorithm	Tier	ZK Property	Verifier Trust
no-proof (0)	Basic	None	Trusts Attester's rho claim
spearman-pedersen (2)	Enhanced	Computational hiding	Verifies commitment consistency
spearman-snark (1)	Maximum	Perfect ZK	Cryptographic proof of relation

Table 16

Verifiers SHOULD require ZK proofs (algorithm > 0) for Enhanced and Maximum tier claims. Basic tier Evidence with no-proof is suitable only for contexts where the Attesting Environment is fully trusted or where process documentation rather than adversarial verification is the goal.

16.6. Explicit Scope Limitations

Per the architectural constraints in this specification, the ZK cognitive load verification mechanism:

* Does NOT perform AI detection or classification. The mechanism documents correlation patterns without inferring their source.

* Does NOT make stylometric claims. Linguistic analysis of content is explicitly out of scope.

* Does NOT infer intent or cognitive state. Observable timing correlation is documented, not interpreted.

* Does NOT capture document content. Both histograms are derived measurements, not content reproductions.

* Does NOT provide surveillance capabilities. Aggregate statistics are verified, not raw input streams.

Evidence generated through this mechanism is tamper-evident and independently verifiable, but interpretation of what the evidence means remains the responsibility of Relying Parties applying their own policies and risk tolerances.

17. Biology Invariant Parameter Configuration

The composite biological score formula presented in Section 14 uses hardcoded weights and thresholds that lack empirical validation. To enable transparent evolution of these parameters as research matures, this section defines a versioned parameter configuration structure with explicit confidence levels indicating the validation status of each parameter.

17.1. Validation Status Taxonomy

Each parameter set carries a validation-status indicator that communicates the epistemic basis for the parameter values to Verifiers and Relying Parties:

Empirical (1): Parameters validated through published peer-reviewed studies with reproducible methodology. The validation-reference field MUST contain a DOI or equivalent stable identifier for the validating research.

Theoretical (2): Parameters derived from established literature on human motor control, cognitive load, or psycholinguistics, but not directly validated for the specific use case of behavioral attestation. This is the current status of all parameters defined in this specification.

Unsupported (3): Parameters that are placeholders or heuristics without theoretical or empirical basis. Relying Parties SHOULD treat claims using unsupported parameters with heightened skepticism and MAY reject such evidence entirely depending on policy.

17.2. Parameter Configuration Structure

The biology-scoring-parameters structure encapsulates all configurable parameters for the biological invariant evaluation, encoded in CBOR per the following CDDL schema:

```
; Biology Invariant Scoring Parameters
; Version: v1.0-draft (validation-status: theoretical)

biology-scoring-parameters = {
    1 => tstr,                  ; version (e.g., "v1.0-draft")
    2 => weight-config,         ; scoring weights
    3 => threshold-config,      ; threshold values
    4 => validation-status,     ; epistemic basis
    ? 5 => tstr,                ; validation-reference (DOI/URL)
    ? 6 => context-profile,     ; optional context-specific profile
}

weight-config = {
    1 => uint,                  ; rho-gap-weight-millibits (400 = 0.4)
    2 => uint,                  ; hurst-weight-millibits (400 = 0.4)
    3 => uint,                  ; adj-phys-weight-millibits (200 = 0.2)
}

threshold-config = {
    1 => uint,                  ; composite-score-min-millibits (750 = 0.75)
    2 => uint,                  ; rho-correlation-min-millibits (700 = 0.7)
    3 => uint,                  ; pink-noise-slope-min-millibits (800 = 0.8)
    4 => uint,                  ; pink-noise-slope-max-millibits (1200 = 1.2)
    5 => uint,                  ; hurst-min-millibits (550 = 0.55)
    6 => uint,                  ; hurst-max-millibits (850 = 0.85)
    ? 7 => uint,                ; lz-complexity-min-millibits (200 = 0.2)
}

validation-status = &(
    empirical: 1,               ; Validated via published study
    theoretical: 2,             ; Based on literature, not validated
    unsupported: 3,             ; Parameters need validation
)

context-profile = &(
    default_v1: 1,              ; General-purpose defaults
    prose_v1: 2,                ; Optimized for natural language prose
    technical_v1: 3,            ; Optimized for code/technical content
    mixed_v1: 4,                ; Mixed prose and technical content
)

; Biology invariant claim structure for inclusion in Evidence
biology-invariant-claim = {
    1 => uint,                  ; score-millibits (computed composite score)
    2 => validation-status,     ; parameter validation level
    3 => tstr,                  ; parameters-version
    4 => bstr,                  ; parameters-hash (SHA-256 of params)
    ? 5 => [* tstr],            ; context-warnings
```

```
    ? 6 => context-profile,    ; profile used for evaluation
}
```

17.3. Current Parameter Values (v1.0-draft)

The following parameter values are defined for version "v1.0-draft". All parameters carry validation-status: theoretical (2), indicating they are derived from literature but not empirically validated for behavioral attestation:

Parameter	Millibits Value	Decimal Equivalent	Literature Basis
rho-gap-weight	400	0.4	Grudin 1983 (error patterns)
hurst-weight	400	0.4	Mandelbrot 1982 (fractal time series)
adj-phys-weight	200	0.2	QWERTY adjacency heuristic
composite-score-min	750	0.75	Heuristic threshold
rho-correlation-min	700	0.7	Kushniruk 1991 (cognitive load)
pink-noise-slope-min	800	0.8	1/f noise literature
pink-noise-slope-max	1200	1.2	1/f noise literature
hurst-min	550	0.55	Mandelbrot 1982; empirical validation
hurst-max	850	0.85	Mandelbrot 1982; empirical validation
lz-complexity-min	200	0.2	Signal starvation prevention

Table 17: Default Profile (default_v1) Parameters

IMPORTANT: These parameters are designated validation-status: theoretical (2). The weights (0.4, 0.4, 0.2) were selected based on general principles from motor control and cognitive load literature, NOT from empirical validation against adversarial simulation or controlled authorship studies. Relying Parties SHOULD interpret biological invariant claims accordingly and MAY apply additional policy constraints for high-stakes verification contexts.

17.4. Context-Specific Profiles

Different content types exhibit different behavioral signatures. The following profiles provide context-specific parameter adjustments:

17.4.1. Prose Profile (prose_v1)

Optimized for natural language prose authorship. Assumes higher cognitive load variation during complex sentence construction and lower physical adjacency errors compared to code. Uses default parameters with the following adjustments:

* rho-gap-weight: 450 (0.45) - Higher weight on cognitive correlation
* hurst-weight: 400 (0.4) - Unchanged
* adj-phys-weight: 150 (0.15) - Lower weight on physical adjacency

Validation-status: unsupported (3). These adjustments are hypothetical and require empirical validation.

17.4.2. Technical Profile (technical_v1)

Optimized for source code and technical content. Assumes higher physical adjacency error rates due to specialized characters and lower cognitive load correlation due to repetitive syntax patterns.

* rho-gap-weight: 300 (0.3) - Lower weight on cognitive correlation
* hurst-weight: 400 (0.4) - Unchanged
* adj-phys-weight: 300 (0.3) - Higher weight on physical adjacency
* composite-score-min: 700 (0.70) - Lower threshold for code

Validation-status: unsupported (3). These adjustments are hypothetical and require empirical validation.

17.5. Parameter Versioning

Parameter version strings follow the format "v{major}.{minor}-{status}" where status is one of "draft", "experimental", or "stable":

* *draft:* Initial parameters under active development. MAY change without notice. Suitable only for testing.

* *experimental:* Parameters undergoing validation studies. Changes require documentation. Suitable for non-adversarial contexts.

* *stable:* Parameters with empirical validation. Changes require major version increment. Suitable for adversarial review.

Implementations MUST include the parameters-hash in biology-invariant-claim to enable Verifiers to confirm which exact parameter values were used, regardless of version string. The hash is computed as:

```
parameters-hash = SHA-256(
    CBOR-encode(biology-scoring-parameters)
)
```

Verifiers SHOULD maintain a registry of known parameter hashes and their associated validation status to enable policy-based acceptance or rejection of Evidence using specific parameter versions.

17.6. Research Limitations Acknowledgment

The behavioral invariant parameters defined in this specification are subject to the following research limitations that Relying Parties MUST consider when interpreting biological invariant claims:

1. *No adversarial validation:* Parameters have not been tested against sophisticated simulation attacks. An adversary with knowledge of the scoring formula could potentially craft timing patterns that satisfy the thresholds.

2. *Population variance:* Human typing behavior varies significantly across individuals, input devices, fatigue levels, and content types. Fixed thresholds may produce false negatives for atypical but genuine authors.

3. *Content dependence:* The correlation between cognitive load and timing delays depends on content complexity. Highly formulaic content (forms, templates, repetitive text) may not exhibit expected behavioral signatures.

4. *Device dependence:* Timing resolution and jitter characteristics vary across input devices and platforms, potentially affecting score reproducibility.

5. *Literature extrapolation:* Referenced studies (Grudin 1983, Mandelbrot 1982, Kushniruk 1991, Rayner 1998) address related phenomena but were not designed for behavioral attestation. Extrapolation to this context requires validation.

Future versions of this specification MAY update parameters based on empirical research. Implementations SHOULD support parameter version negotiation to enable graceful migration as the evidence base matures.

17.7. Active Behavioral Probes

Beyond passive timing analysis, implementations MAY employ active behavioral probes that analyze response characteristics to specific interaction patterns. These probes provide additional validation signals that are difficult to synthesize without genuine human motor system involvement.

17.7.1. Galton Invariant Probe

The Galton Invariant measures rhythm perturbation recovery by analyzing how quickly and consistently timing returns to baseline after disruption events (e.g., errors, pauses, context switches). Named after the Galton board's probability distribution, this probe characterizes the "absorption coefficient" of behavioral rhythm perturbations.

Parameters:

* *Absorption coefficient (α):* Valid range $\alpha \in [0.3, 0.8]$. Values below 0.3 indicate insufficient rhythm recovery (possibly synthetic constant-rate input). Values above 0.8 indicate excessive damping (possibly mechanically smoothed).

* *Time constant (τ):* Recovery half-life in milliseconds. Typical human values: 200-800ms.

* *Asymmetry factor:* Ratio of positive to negative perturbation recovery. Human motor systems typically show mild asymmetry (factor 0.8-1.2).

17.7.2. Reflex Gate Probe

The Reflex Gate measures minimum achievable latency and its variability, characterizing neural pathway delay constraints. Human motor responses exhibit floor latencies imposed by physiological signal propagation that cannot be bypassed by simulation.

Parameters:

* *Minimum latency:* MUST be 100ms for valid human input. Latencies consistently below 100ms indicate either hardware/ software injection or pre-computed responses.

* *Coefficient of variation (CV):* Valid range CV $\in$ [0.15, 0.40]. Values below 0.15 indicate mechanical consistency. Values above 0.40 indicate either extreme fatigue or non-physiological variation patterns.

* *Distribution shape:* Human reaction times follow ex-Gaussian distributions. Significant deviation from this shape indicates synthetic generation.

17.7.3. Active Probe Security Considerations

Active probes increase the cost of successful simulation but do not provide absolute guarantees:

* Adversaries with knowledge of probe parameters could craft timing sequences that satisfy the validation ranges.

* Probe parameters are based on typical human physiology; atypical but genuine users may produce out-of-range values.

* Implementations SHOULD use active probes as supplementary signals rather than hard rejection criteria.

17.8. Labyrinth Structure Analysis

The Labyrinth structure applies dynamical systems theory to characterize the phase space topology of timing sequences. Based on Takens' theorem for delay-coordinate embedding, this analysis reconstructs attractor geometry from the one-dimensional timing series, enabling detection of non-linear behavioral dynamics that are difficult to synthesize.

17.8.1. Delay-Coordinate Embedding

Given a timing interval sequence {t_1, t_2, ..., t_n}, the phase space reconstruction creates m-dimensional vectors:

$$v_i = (t_i, t_{i+\tau}, t_{i+2\tau}, ..., t_{i+(m-1)\tau})$$

where m is the embedding dimension and τ is the delay parameter. Parameters:

* *Embedding dimension (m):* Valid range 3-8. Lower values may miss attractor structure; higher values introduce spurious correlations.

* *Delay parameter (τ):* Selected via mutual information minimization or autocorrelation zero-crossing.

17.8.2. Topological Invariants

The reconstructed phase space is characterized by topological invariants that distinguish genuine behavioral dynamics from synthetic sequences:

Correlation dimension (D_2): Measures the complexity of the attractor. Valid range: $D_2 \in [1.5, 5.0]$. Values near 1.0 indicate deterministic periodic behavior; values near the embedding dimension indicate stochastic noise.

Betti numbers (β_0, β_1, β_2): Topological invariants counting connected components (β_0), loops (β_1), and voids (β_2) in the attractor. Human behavioral attractors typically show specific Betti number patterns reflecting cognitive-motor coupling dynamics.

Recurrence rate: Fraction of recurrence points in the reconstructed phase space. Synthetic sequences often show either too high (periodic) or too low (random) recurrence rates compared to genuine behavioral dynamics.

Determinism: Fraction of recurrence points forming diagonal lines in recurrence plots. Genuine behavioral sequences show intermediate determinism reflecting cognitive influence on motor timing.

17.8.3. Labyrinth Analysis Security Considerations

Phase space analysis provides additional forgery cost but is not a complete defense:

* Adversaries could potentially train generative models to produce timing sequences with specific topological properties.

* The computational cost of labyrinth analysis is significant; implementations MAY perform this analysis only for high-value evidence or as a secondary verification step.

* Topological analysis is sensitive to sequence length; short sessions may not provide sufficient data for reliable invariant estimation.

17.9. Guidance for Interpreting Unsupported Confidence Levels

When Evidence contains biology-invariant-claim with validation-status: unsupported (3), Verifiers and Relying Parties SHOULD apply the following interpretation guidance:

* The claim SHOULD NOT be treated as dispositive evidence of human authorship or the absence thereof.

* The claim MAY contribute to a broader forensic assessment when combined with other evidence types (VDF temporal bounds, external anchors, hardware attestation).

* High-stakes verification contexts (legal proceedings, academic integrity decisions with significant consequences) SHOULD require at least validation-status: theoretical (2) and preferably validation-status: empirical (1).

* Policy engines MAY define minimum validation-status thresholds for claim acceptance, expressed in the trust-policy structure defined in Section 29.

* Attestation Results SHOULD include a caveat when biological invariant claims rely on unsupported parameters, using the caveats mechanism defined in Section 9.14.5.

18. VDF Entanglement

The Jitter Seal achieves temporal binding through entanglement with the VDF proof chain. The VDF input for segment N includes the jitter binding from segment N:

```
VDF_input{N} = H(
    tree-root{N-1} ||
    content-hash{N} ||
    jitter-binding{N}.entropy-commitment
)

VDF_output{N} = VDF(VDF_input{N}, iterations{N})
```

This entanglement creates a causal dependency: the VDF output cannot be computed until the jitter entropy is captured and committed. Combined with the VDF's sequential computation requirement, this ensures that:

1. The jitter data existed before the VDF computation began

2. The checkpoint cannot be backdated without recomputing the entire VDF chain from that point forward

3. The minimum time between checkpoints is bounded by VDF computation time plus jitter observation time

19. Verification Procedure

A Verifier appraises the Jitter Seal through the following procedure:

1. Structural Validation:

   Verify all required fields are present and correctly typed per the CDDL schema.

2. Binding MAC Verification:

   Recompute the binding-mac using the segment chain key and verify it matches the provided value.

3. Entropy Commitment Verification (if raw-intervals present):

   Recompute H(intervals) and verify it matches entropy-commitment.

4. Histogram Consistency (if raw-intervals present):

   Recompute histogram buckets from raw intervals and verify consistency with the provided summary.

5. Entropy Threshold Check:

   Verify estimated-entropy-bits meets the minimum threshold for the claimed evidence tier. RECOMMENDED minimum: 32 bits for Standard tier, 64 bits for Enhanced tier.

6. Sample Count Check:

   Verify sample-count is consistent with the document size and claimed editing duration. Anomalously low sample counts relative to content length indicate potential evidence gaps.

7. Anomaly Assessment:

   If anomaly-flags are present, incorporate them into the overall forensic assessment. The presence of anomalies does not invalidate the evidence but affects confidence.

8. VDF Entanglement Verification:

   Verify the entropy-commitment appears in the VDF input computation for this checkpoint.

The verification result contributes to the computationally-bound claims defined in the absence-section:

* jitter-entropy-above-threshold (claim type 8): PROVEN if estimated-entropy-bits exceeds threshold

* jitter-samples-above-count (claim type 9): PROVEN if sample-count exceeds threshold

20. Anomaly Detection

The Attesting Environment MAY flag anomalies in the captured timing data:

Value	Flag	Indication
1	unusually-regular	Timing distribution has lower variance than typical human input (coefficient of variation < 0.1)
2	burst-detected	Sustained high-speed input exceeding 200 WPM for >30 seconds
3	gap-detected	Significant editing gap (>5 minutes) within what appears to be a continuous session
4	paste-heavy	>50% of content added via paste operations in this segment interval
5	semantic-mismatch	Low correlation (rho < 0.5) between Information Density and Timing Density across segment intervals

Table 18

Anomaly flags are informational and do not constitute claims about authorship or intent. They provide context for Verifier appraisal and Relying Party decision-making.

21. Relationship to RATS Evidence

The Jitter Seal extends the RATS evidence model [RFC9334] in several ways:

Behavioral Evidence: Traditional RATS evidence attests to system state (software versions, configuration, integrity). The Jitter Seal attests to behavioral characteristics of the input stream, capturing properties that emerge only during genuine interaction.

Continuous Attestation: Unlike point-in-time attestation, Jitter Seals are accumulated throughout an authoring session. Each checkpoint adds to the behavioral evidence corpus, with earlier seals constraining what later seals can claim.

Non-Reproducible Evidence: RATS evidence can typically be regenerated by returning to the same system state. Jitter Seal evidence cannot be regenerated because the timing entropy existed only at the moment of capture.

Epoch Marker Compatibility: The VDF-entangled Jitter Seal can function as a local freshness mechanism compatible with the Epoch Markers framework [I-D.ietf-rats-epoch-markers]. The VDF output chain provides relative ordering; external anchors provide absolute time binding.

22. Privacy Considerations

Keystroke timing data is behavioral biometric data: while not traditionally classified as biometric data, timing patterns can potentially identify individuals or reveal sensitive information about cognitive state or physical condition.

22.1. Mitigation Measures

* Histogram Aggregation:

  By default, only aggregated histogram data is included in the evidence packet. Raw intervals are optional and SHOULD only be disclosed when enhanced verification is required.

* Bucket Granularity:

  The RECOMMENDED bucket boundaries (50ms minimum width) prevent reconstruction of exact keystroke sequences while preserving statistically significant patterns.

* No Character Mapping:

  Timing intervals are recorded without association to specific characters or words. The evidence captures rhythm without content.

* Session Isolation:

  Jitter data is bound to a specific evidence packet and segment chain. Cross-session correlation requires access to multiple evidence packets.

22.2. Disclosure Recommendations

Implementations SHOULD inform users that:

1. Typing rhythm information is captured and included in evidence packets

2. Evidence packets may be shared with Verifiers and potentially with Relying Parties

3. Raw timing data (if disclosed) could theoretically be used for behavioral analysis

Users SHOULD have the option to:

1. Disable raw-intervals disclosure (histogram-only mode)

2. Request deletion of evidence packets after verification

3. Review captured entropy statistics before packet finalization

23. Security Considerations

23.1. Replay Attacks

An adversary might attempt to replay captured jitter data from a previous session. This attack is mitigated by:

1. VDF entanglement:

   The jitter commitment is bound to the VDF chain, which includes the previous checkpoint hash.

2. Chain MAC:

   The binding-mac includes the previous checkpoint hash, preventing transplantation.

3. Content binding:

   The jitter data is associated with specific content hashes that change with each edit.

4. Verifier nonce binding:

   When verification freshness is required, Verifiers SHOULD provide a 32-byte cryptographically random nonce. The Attesting Environment incorporates this nonce into the packet signature:

SIG_k(H3 || verifier_nonce). This proves the evidence was generated in response to a specific verification request, preventing replay of previously generated evidence packets. The verifier_nonce field MUST be present when replay prevention is required by the verification policy.

23.2. Simulation Attacks

An adversary might attempt to generate synthetic timing data that mimics human patterns. The cost of this attack is bounded by:

1. Entropy requirement:

   Meeting the entropy threshold requires sufficient variation in timing. Perfectly regular synthetic input will fail the entropy check.

2. Real-time constraint:

   The VDF entanglement requires that jitter data be captured before VDF computation. Generating synthetic timing that passes statistical tests while maintaining real-time constraints is non-trivial.

3. Statistical consistency:

   Synthetic timing must be consistent across all checkpoints. Anomaly detection may flag statistically improbable patterns.

The Jitter Seal does not claim to make simulation impossible, only to make it costly relative to genuine interaction. The forgery-cost-section provides quantified bounds on attack costs.

23.3. Attesting Environment Trust

The Jitter Seal relies on the Attesting Environment to accurately capture and report timing data. A compromised AE could fabricate jitter data. This is addressed by:

1. Hardware binding (hardware-section) for AE integrity

2. Calibration attestation for VDF speed verification

3. Clear documentation of AE trust assumptions in absence-claim structures (ae-trust-basis field)

Chain-verifiable claims (1-15) do not depend on AE trust beyond basic data integrity. Monitoring-dependent claims (16-63) explicitly document their AE trust requirements.

24. Verifiable Delay Functions

In this section, the Verifiable Delay Function (VDF) mechanisms used to establish temporal ordering and minimum elapsed time between checkpoints are specified, providing the temporal guarantees that distinguish this RATS [RFC9334] profile from attestation frameworks that rely solely on timestamps. Algorithm-agility is afforded by the CDDL [RFC8610] schema design, with both iterated hash constructions using SHA-256 [RFC6234] or SHA3-256 (which provide O(n) verification through recomputation) and succinct VDF schemes per Pietrzak [Pietrzak2019] and Wesolowski [Wesolowski2019] (which provide O(log n) or O(1) verification through cryptographic proofs) being supported. VDFs are functions that require a specified amount of sequential computation time to evaluate regardless of available parallelism, yet whose outputs can be verified efficiently, a property that makes them ideal for establishing unforgeable temporal ordering in the CBOR [RFC8949] encoded Evidence packets without reliance on RFC 3161 [RFC3161] timestamps or other trusted third parties.

24.1. Post-Quantum Iteration Parameters

To provide security against quantum adversaries using Groverx27s algorithm, the minimum iteration count T MUST be doubled (2*T) compared to classical security parameters. Implementers MUST assume a quadratic speedup in parallel preimage search.

24.2. VDF Construction

Appearance of a VDF proof in each checkpoint is mandated by this specification, with the following fields encoded in CBOR per the CDDL schema. The vdf-proof structure captures the algorithm selection, parameters, input/output pairs, cryptographic proof (for succinct VDFs), and calibration attestation that enables Verifiers to assess whether the claimed-duration is plausible for the iteration count on the attested hardware.

```
vdf-proof = {
    1 => vdf-algorithm,           ; algorithm
    2 => vdf-params,              ; params
    3 => bstr,                    ; input
    4 => bstr,                    ; output
    5 => bstr,                    ; proof
    6 => duration,                ; claimed-duration
    7 => uint,                    ; iterations
    ? 8 => calibration-attestation, ; calibration (REQUIRED)
}
```

24.2.1. Algorithm Registry

The following VDF algorithms are delineated in the algorithm registry, with algorithm selection indicated by the algorithm field in the vdf-proof structure encoded in CBOR. The iterated hash algorithms (1-2) use SHA-256 or SHA3-256 with implicit proofs (verification by recomputation), while the succinct VDF algorithms (16-19) use the constructions from Pietrzak and Wesolowski with explicit cryptographic proofs enabling efficient verification.

Value	Algorithm	Status	Proof Size
1	iterated-sha256	MUST support	0 (implicit)
2	iterated-sha3-256	SHOULD support	0 (implicit)
16	pietrzak-rsa3072	MAY support	~1 KB
17	wesolowski-rsa3072	MAY support	~256 bytes
18	pietrzak-class-group	MAY support	~2 KB
19	wesolowski-class-group	MAY support	~512 bytes

Table 19

Reservation of algorithm values 1-15 for iterated hash constructions using SHA-256 or similar hash functions is established, with values 16-31 being reserved for succinct VDF schemes based on the constructions of Pietrzak and Wesolowski. Availability of values 32+ for future allocation is maintained to accommodate advances in VDF research, with the CDDL schema extensibility ensuring forward compatibility within the RATS architecture.

24.2.2. Iterated Hash Construction

Computation by the iterated hash VDF is accomplished using repeated application of SHA-256 or SHA3-256 as the hash function, with the output of each iteration becoming the input to the next. This construction is the simplest VDF implementation, requiring only the hash function from RFC 6234 and providing inherent parallelization resistance because each iteration depends on the previous output. The mathematical definition follows:

```
output = H^n(input)

where H^n denotes n iterations of hash function H:
  H^0(x) = x
  H^n(x) = H(H^(n-1)(x))
```

Parameters for iterated hash VDFs:

```
iterated-hash-params = {
    1 => hash-algorithm,    ; hash-function
    2 => uint,              ; iterations-per-second
}
```

Recording of the calibrated performance of the Attesting Environment is accomplished by the iterations-per-second field in the CBOR encoded params structure, making possible assessment by Verifiers per the RATS architecture of whether the claimed-duration is plausible for the iteration count on the attested hardware. When TPM 2.0 [TPM2.0] or Secure Enclave attestation is available, the calibration can be hardware-signed for additional trust. The following properties are exhibited by iterated hash VDFs using SHA-256:

Verification Cost: O(n) -- Verifier must recompute all iterations. This is acceptable for the iteration counts typical in authoring scenarios (10^6 to 10^9 iterations).

Parallelization Resistance: Inherently sequential. Each iteration depends on the previous output. No known parallelization attack.

Hardware Acceleration: SHA-256 acceleration (e.g., Intel SHA Extensions, ARM Cryptography Extensions) provides ~3-5x speedup over software. This is accounted for in calibration.

24.2.3. Succinct VDF Construction

O(log n) or O(1) verification time is afforded by succinct VDFs based on the constructions of Pietrzak and Wesolowski, at the cost of larger proof sizes encoded in CBOR (approximately 1-2 KB depending on construction and security parameters) and more complex computation involving modular exponentiation. These constructions are based on repeated squaring in groups with unknown order (RSA groups or class groups), a mathematical structure that inherently resists parallelization because the order of the group is not known to the prover, unlike the iterated SHA-256 construction where the mathematical structure is simpler but verification requires O(n) recomputation.

```
succinct-vdf-params = {
    10 => uint,             ; modulus-bits (minimum 3072)
    ? 11 => uint,           ; security-parameter
}
```

Key set 10-19 disambiguates succinct params from iterated hash params (key set 1-9) without requiring a type tag.

OPTIONAL status is assigned to succinct VDFs per Pietrzak and Wesolowski within this RATS profile, with their intended use being in scenarios where verification must complete in bounded time regardless of delay duration (enabling real-time verification in time-constrained environments), where CBOR encoded Evidence packets may contain very long VDF chains (millions of checkpoints accumulated over extended authoring periods), or where O(n) SHA-256 recomputation cannot be afforded by third-party Verifiers with limited computational resources. When succinct VDFs are used, the proof field in the CDDL schema contains the cryptographic proof of correct computation (approximately 256 bytes for Wesolowski or 1 KB for Pietrzak); for iterated hash VDFs using SHA-256, the proof field is empty and verification is accomplished by recomputation of the hash chain.

24.3. Causality Property

Unforgeable temporal ordering is established by the VDF chain through structural causality, wherein each checkpoint’s VDF output depends on the previous checkpoint’s output, creating a sequence that can only be computed in order regardless of available computational resources. A key novel contribution of the Proof of Process framework within the RATS architecture is constituted by this property, which distinguishes this approach from timestamp-based ordering using RFC 3161 that relies on trusted third parties. While external anchors via RFC 3161 may be used to bind the VDF chain to absolute time, the

relative ordering is established cryptographically through SHA-256 hash entanglement without any external trust assumptions.

24.3.1. Checkpoint Entanglement

Computation of the VDF input for segment N is accomplished using SHA-256 to combine the previous VDF output, current content hash, jitter commitment (bound via HMAC), and sequence number into a single input for the VDF function, as shown in the formula below. This entanglement is encoded in CBOR per the CDDL schema and creates the causal dependencies that establish temporal ordering.

```
VDF_input{N} = H(
    VDF_output{N-1} ||      ; Previous VDF output
    content-hash{N} ||      ; Current document state
    jitter-commitment{N} || ; Captured behavioral entropy
    sequence{N}             ; Checkpoint sequence number
)
```

For the genesis checkpoint (N = 0):

```
VDF_input{0} = H(
    session-entropy ||      ; Random 256-bit session seed
    content-hash{0} ||      ; Initial document state
    jitter-commitment{0} ||
    0x00000000              ; Sequence zero
)
```

The following properties are ensured by this construction within the RATS architecture: Sequential Dependency is established such that VDF_output{N} cannot be computed without VDF_output{N-1}, making the chain inherently sequential regardless of available parallelism; Content Binding is established such that each VDF output is bound to a specific document state computed using SHA-256, with changing the content invalidating all subsequent VDF proofs in the CBOR encoded chain; Jitter Binding is established such that the behavioral entropy commitment computed via SHA-256 and bound via HMAC is entangled with the VDF, as detailed in Section 18; and Precomputation is prevented because the SHA-256 input depends on runtime values (content hash, jitter commitment) that are unknown until the checkpoint is created.

24.3.2. Temporal Ordering Without Trusted Time

Relative temporal ordering without reliance on trusted timestamps such as RFC 3161 [RFC3161] is afforded by the VDF causality property, distinguishing this RATS profile from attestation schemes that require trusted time sources:

Relative Ordering: Checkpoint N necessarily occurred after segment N-1, because VDF_input{N} requires VDF_output{N-1}.

Minimum Elapsed Time: The time between checkpoints N-1 and N is at least:

```
min_elapsed{N} = iterations{N} / calibration_rate
```

where calibration_rate is the attested iterations-per-second for the device.

Cumulative Time Bound: The total minimum time to produce the evidence packet is:

```
min_total = sum(iterations[i] / calibration_rate) for i = 0..N
```

Absolute Time Binding: External anchors including RFC 3161 timestamps and blockchain proofs bind the SHA-256 segment chain to absolute time. The VDF provides the relative ordering through causality; the RFC 3161 anchors provide the epoch binding to wall-clock time.

24.3.3. Backdating Resistance

The following steps must be accomplished by an adversary attempting to backdate evidence within this RATS profile: content that produces the desired content-hash computed using SHA-256 must be generated (this is prevented by preimage resistance of SHA-256); jitter data that produces valid entropy-commitment via SHA-256 must be generated (this requires access to the original timing stream or statistical simulation); the VDF chain from the backdated checkpoint forward must be computed (this requires sequential time proportional to iterations); and all of the above must be completed before any external anchor such as RFC 3161 timestamp or blockchain proof confirms a later checkpoint (this creates a race condition that the adversary loses if anchors are obtained promptly). Linear growth with the number of subsequent checkpoints and the iteration count per checkpoint characterizes the cost of VDF recomputation, with this cost being quantified in the forgery-cost-section using concrete metrics. Crucially, parallelization of VDF computation cannot be accomplished by the adversary: inherent sequentiality is exhibited by both iterated SHA-256 constructions and the group-theoretic constructions of Pietrzak and Wesolowski. Even with unlimited computational resources, completion of each VDF must be awaited before starting the next, creating an irreducible time cost for any backdating attack that exceeds the original authoring time.

24.3.4. Time Evidence and Degradation

Verifiable Delay Functions provide relative temporal ordering but cannot independently establish absolute time. When external anchors are unavailable, the strength of temporal evidence degrades. This section defines explicit tiers that document the achievable temporal binding based on available anchor sources, enabling Verifiers to make informed trust decisions.

24.3.4.1. Time Binding Tier Definitions

Four tiers of temporal binding are defined, based on the combination of external anchors available at evidence generation time:

Tier	Value	Anchor Requirements	Time Binding Strength
MAXIMUM	1	>=2 blockchain + >=2 TSA	Strong absolute time
ENHANCED	2	>=1 blockchain OR >=2 TSA	Probable absolute time
STANDARD	3	>=1 TSA	Weak absolute time
DEGRADED	4	VDF + local clock only	Relative time only

Table 20: Time Binding Tier Requirements

The tier classification follows the principle that redundancy across independent anchor types provides stronger temporal assurance than reliance on any single source or type.

24.3.4.2. Tier Capabilities and Limitations

MAXIMUM Tier: Evidence at this tier can prove: the document existed before a specific absolute time (via blockchain confirmation); the document was timestamped by multiple independent authorities (via TSA tokens); the relative ordering of checkpoints (via VDF); and the minimum elapsed time between states (via VDF calibration).

Suitable for: litigation support, regulatory compliance, forensic investigation, and contexts requiring independently verifiable absolute time claims.

ENHANCED Tier: Evidence at this tier can prove: probable absolute

time binding through either blockchain proof or redundant TSA timestamps; relative ordering of checkpoints; and minimum elapsed time. However, the absence of cross-type redundancy introduces single-source risk if the chosen anchor type is later compromised or disputed.

Suitable for: professional documentation, academic submissions, and contexts where absolute time is important but cross-verification requirements are moderate.

STANDARD Tier: Evidence at this tier can prove: absolute time binding dependent on a single Time Stamping Authority; relative ordering; and minimum elapsed time. The temporal claim is only as trustworthy as the specific TSA, with no independent corroboration.

Suitable for: internal records, personal documentation, and contexts where the TSA is trusted by all relevant parties.

LIMITATIONS: If the TSA is compromised, unavailable for verification, or disputed, no independent time evidence exists. Verifiers SHOULD document TSA identity in their assessment.

DEGRADED Tier: Evidence at this tier can ONLY prove: relative ordering of checkpoints (checkpoint N necessarily occurred after checkpoint N-1); and minimum elapsed time between checkpoints (via VDF and calibration). The local clock timestamp is recorded but is untrusted for verification purposes.

CANNOT PROVE: absolute time of evidence creation; that the evidence was not pre-computed and held before claimed timestamps; or epoch binding to any external reference.

Suitable for: offline scenarios, air-gapped environments, and contexts where relative ordering is sufficient. NOT suitable for contexts requiring absolute time claims or adversarial verification of creation time.

24.3.4.3. Explicit DEGRADED Tier Limitations

When evidence is generated at DEGRADED tier, Attesters MUST understand and Verifiers MUST document the following limitations:

* The local timestamp in the evidence packet reflects the Attesting Environment’s clock at generation time, which may be manipulated or misconfigured. It is NOT cryptographically bound to any external reference.

* An adversary with sufficient computational resources could generate evidence with a past local timestamp by computing the VDF chain forward from any starting point. The cost of this attack is bounded by the forgery-cost-bounds analysis but is not prevented by temporal binding.

* DEGRADED evidence provides process documentation suitable for honest parties but offers limited protection against adversarial backdating beyond the VDF computational cost.

* The time-evidence structure MUST contain a null value for absolute-time-bounds when the tier is DEGRADED, explicitly indicating the absence of absolute time claims.

Attestation results for DEGRADED tier evidence SHOULD include a caveat explicitly stating that no absolute time claims can be verified.

24.3.4.4. Re-anchoring for Progressive Strengthening

Evidence generated at a lower tier MAY be progressively strengthened by obtaining additional anchors after initial generation:

Post-generation Anchoring: If an evidence packet was generated at DEGRADED or STANDARD tier, the Attester MAY subsequently obtain additional anchors (blockchain proofs, TSA timestamps) that bind the packet-hash to later absolute times. This does NOT retroactively prove when the evidence was originally created, but does prove that the evidence existed before the anchor confirmation time.

Anchor Status Tracking: The anchor-status structure documents both successful and failed anchor attempts. Verifiers can assess whether anchor unavailability was due to network conditions, service outages, or deliberate omission. A pattern of consistently failed anchors across multiple services may indicate intentional avoidance rather than legitimate unavailability.

Upgrade Path: DEGRADED to STANDARD: Obtain one RFC 3161 timestamp on the packet-hash.

STANDARD to ENHANCED: Obtain either one blockchain anchor OR a second TSA timestamp from an independent authority.

ENHANCED to MAXIMUM: Obtain both blockchain diversity (>=2 chains) AND TSA diversity (>=2 authorities).

Time Bound Updates: When additional anchors are obtained, the

absolute-time-bounds SHOULD be updated to reflect the tighter constraints. The earliest-possible timestamp is the creation time of the earliest confirmed anchor; the latest-possible is the creation time of the latest confirmed anchor before any modifications. Re-anchoring narrows the uncertainty window.

24.3.4.5. Admissibility Guidance by Tier

Relying Parties SHOULD consider the following guidance when assessing temporal claims based on time binding tier:

Tier	Absolute Time Claims	Relative Time Claims	Recommended Contexts
MAXIMUM	Strong - independently verifiable	Strong - VDF + calibration	Legal, regulatory, forensic
ENHANCED	Moderate - single-type dependency	Strong - VDF + calibration	Professional, academic
STANDARD	Weak - single-authority dependency	Strong - VDF + calibration	Internal, trusted-party
DEGRADED	None - cannot verify	Moderate - VDF + calibration	Offline, process documentation

Table 21: Temporal Admissibility by Tier

Verifiers MUST NOT make absolute time claims for DEGRADED tier evidence. Attestation results for DEGRADED evidence SHOULD explicitly state that temporal claims are limited to relative ordering and minimum elapsed time.

Policy engines MAY require minimum tier thresholds for specific use cases. For example, a litigation support policy might require ENHANCED or MAXIMUM tier for temporal claims to be considered in evidence assessment.

24.3.4.6. Time Evidence Structure

The time-evidence structure captures the complete temporal binding assessment for an evidence packet:

```
    time-binding-tier = &(
        maximum: 1,     ; >=2 blockchain + >=2 TSA anchors
        enhanced: 2,    ; >=1 blockchain OR >=2 TSA anchors
        standard: 3,    ; >=1 TSA anchor
        degraded: 4,    ; VDF + local clock only
    )

    time-evidence = {
        1 => time-binding-tier,         ; tier
        2 => absolute-time-bounds / null, ; bounds (null if degraded)
        3 => relative-time-proof,       ; vdf-duration
        4 => [* anchor-status],         ; anchor-statuses
        5 => [* tstr],                  ; recommendations
    }

    absolute-time-bounds = {
        1 => pop-timestamp,             ; earliest-possible
        2 => pop-timestamp,             ; latest-possible
        3 => uint,                      ; uncertainty-seconds
        4 => uint,                      ; anchor-count
    }

    relative-time-proof = {
        1 => uint,                      ; total-vdf-iterations
        2 => uint,                      ; min-elapsed-seconds
        3 => uint,                      ; max-elapsed-seconds
        4 => uint,                      ; checkpoint-count
    }

    anchor-status = {
        1 => anchor-type,               ; type
        2 => anchor-state,              ; status
        ? 3 => tstr,                    ; reason (if unavailable/failed)
        ? 4 => pop-timestamp,           ; last-attempt
        ? 5 => tstr,                    ; anchor-id
    }

    anchor-type = &(
        bitcoin: 1,
        ethereum: 2,
        rfc3161: 3,
        drand: 4,
        opentimestamps: 5,
    )

    anchor-state = &(
        confirmed: 1,
        pending: 2,
```

```
        unavailable: 3,
        failed: 4,
        expired: 5,
    )
```

The recommendations field (key 5) SHOULD contain actionable guidance for strengthening the temporal binding. Examples include: "Obtain blockchain anchor within 24 hours to upgrade to ENHANCED tier", "Re-attempt TSA anchoring when network connectivity is restored", or "Current tier is MAXIMUM; no further strengthening available".

24.4. Calibration Attestation

Calibration attestation addresses a critical verification problem within the RATS architecture: how does a Verifier know whether the claimed VDF iterations could have been computed in the claimed duration on the Attester's hardware? Without calibration, an adversary could claim a slow device while actually using fast hardware, thereby appearing to have spent more time than actually elapsed. The calibration-attestation structure encoded in CBOR per the CDDL schema addresses this by recording a hardware-attested measurement of VDF performance, optionally signed using TPM 2.0 or Secure Enclave keys via COSE.

24.4.1. Attestation Structure

```
       calibration-attestation = {
           1 => uint,              ; calibration-iterations
           2 => pop-timestamp,     ; calibration-time
           3 => cose-signature,    ; hw-signature
           4 => bstr,              ; device-nonce
           ? 5 => tstr,            ; device-model
           ? 6 => tstr,            ; hardware-class
       }
```

calibration-iterations (key 1): The number of VDF iterations completed in a 1-second calibration burst at session start.

calibration-time (key 2): Timestamp when calibration was performed. SHOULD be within 24 hours of the first checkpoint.

hardware-signed attestation (key 3): COSE_Sign1 signature over the calibration data, produced by hardware-bound keys (Secure Enclave, TPM, etc.).

device-nonce (key 4): Random 256-bit value generated at calibration time. Prevents replay of calibration attestations across sessions.

device-model (key 5, optional): Human-readable device identifier for reference purposes. Not used in verification.

hardware-class (key 6, optional): An identifier for the hardware security module or processor generation (e.g., "tpm-2.0-infineon-v1", "apple-se-m3"). Enables Verifiers to perform plausibility checks against a whitelist of expected hash rates for the attested hardware.

24.4.2. Calibration Procedure

The Attesting Environment performs calibration as follows:

1. Generate Nonce:

   Generate a cryptographically random 256-bit device-nonce.

2. Initialize Timer:

   Record high-resolution start time T_start.

3. Execute Calibration Burst:

   Compute VDF iterations using the session's VDF algorithm, starting from H(device-nonce), until 1 second has elapsed.

4. Record Result:

   calibration-iterations = number of iterations completed.

5. Generate Attestation:

   Construct the attestation payload and sign with hardware-bound key.

The attestation payload for signing:

```
attestation-payload = CBOR({
    "alg": vdf-algorithm,
    "iter": calibration-iterations,
    "nonce": device-nonce,
    "time": calibration-time
})
```

24.4.3. Calibration Verification

A Verifier validates calibration attestation as follows:

1. Signature Verification:

   Verify the COSE_Sign1 signature using the device's public key (from hardware-section or certificate chain).

2. Nonce Uniqueness:

   Verify the device-nonce has not been seen in other sessions (optional, requires Verifier state).

3. Plausibility Check:

   Verify calibration-iterations falls within expected range for the device class:

   * Mobile devices: 10^5 - 10^7 iterations/second

   * Desktop/laptop: 10^6 - 10^8 iterations/second

   * Server-class: 10^7 - 10^9 iterations/second

4. Consistency Check:

   For each checkpoint, verify:

   claimed-duration >= iterations / (calibration-iterations * tolerance)

   where tolerance accounts for measurement variance (RECOMMENDED: 1.1, i.e., 10% margin).

24.4.4. Trust Model

Calibration attestation relies on hardware-bound key integrity:

* With hardware attestation:

  The calibration rate is trustworthy to the extent that the hardware security module is trustworthy. An adversary cannot claim faster-than-actual calibration without compromising the HSM.

* Without hardware attestation:

  The calibration rate is self-reported by the Attesting Environment. The Verifier should apply conservative assumptions and may require external anchors for time verification.

The hardware-section documents whether hardware attestation is available and which platform is used.

24.5. Verification Procedure

A Verifier appraises VDF proofs through the following procedure:

24.5.1. Iterated Hash Verification

For iterated hash VDFs, verification requires recomputation:

1. Reconstruct Input:

   Compute VDF_input{N} from the segment data using the entanglement formula in Section 24.3.1.

2. Recompute VDF:

   Execute iterations{N} hash iterations starting from VDF_input{N}.

3. Compare Output:

   Verify the computed output matches the claimed VDF_output{N}.

4. Verify Duration (if calibration present):

   Apply the consistency check from Section 24.4.3.

For large evidence packets, Verifiers MAY use sampling strategies:

* Verify first and last checkpoints fully
* Randomly sample intermediate checkpoints
* Verify chain linkage (prev-hash) for all checkpoints

24.5.2. Succinct VDF Verification

For succinct VDFs, verification uses the cryptographic proof:

1. Reconstruct Input:

   Compute VDF_input{N} as above.

2. Parse Proof:

   Decode the proof field according to the algorithm specification.

3. Verify Proof:

Execute the algorithm-specific verification procedure (Pietrzak or Wesolowski).

4. Verify Duration:

Apply calibration consistency check.

24.6. Algorithm Agility

24.6.1. Migration Path

Evidence packets MAY contain checkpoints using different VDF algorithms. This enables migration scenarios:

* Upgrading from iterated-sha256 to iterated-sha3-256
* Transitioning from iterated hash to succinct VDF
* Adopting post-quantum secure constructions

Algorithm changes SHOULD occur at session boundaries. Within a session, algorithm consistency is RECOMMENDED for simplicity.

24.6.2. Post-Quantum Considerations

Current VDF constructions have varying post-quantum security:

Iterated Hash (SHA-256, SHA3-256): Grover's algorithm provides quadratic speedup for preimage attacks. This affects collision resistance but not the sequential computation property. The VDF remains secure with doubled iteration counts.

RSA-based (Pietrzak, Wesolowski): Vulnerable to Shor's algorithm. Not recommended for long-term evidence that must remain verifiable in a post-quantum era.

Class-group based: Based on class group computations in imaginary quadratic fields. Quantum security is less well understood but believed to be stronger than RSA.

For evidence intended to remain valid for decades, iterated hash VDFs are RECOMMENDED.

24.7. Security Considerations

24.7.1. Hardware Acceleration Attacks

An adversary with specialized hardware (ASICs, FPGAs) may compute VDF iterations faster than the calibrated rate. Mitigations:

* Calibration Reflects Actual Hardware:

  Calibration is performed on the actual device, so the calibration rate already accounts for any acceleration available to the Attester.

* Asymmetric Advantage Limited:

  SHA-256 is widely optimized. The speedup from custom hardware over commodity CPUs with SHA extensions is typically less than 10x.

* Economic Analysis:

  The forgery-cost-section quantifies the cost of acceleration attacks in terms of hardware investment and time.

24.7.2. Parallelization Resistance

VDFs are designed to resist parallelization:

Iterated Hash: Each iteration depends on the previous output. No parallelization is possible without breaking the hash function's preimage resistance.

Succinct VDFs: Based on repeated squaring in groups with unknown order. Parallelization would require factoring the modulus (RSA-based) or solving the class group order problem (class-group based).

The key insight: an adversary with P processors cannot compute the VDF P times faster. The best known attacks provide negligible parallelization advantage.

24.7.3. Time-Memory Tradeoffs

For iterated hash VDFs, an adversary might attempt to precompute and store intermediate values:

* Rainbow Tables:

Precomputing H^n(x) for many x values. Mitigated by the unpredictable VDF input (includes content hash and jitter commitment).

* Checkpoint Tables:

  Storing every k-th intermediate value during legitimate computation. Enables faster recomputation from nearby checkpoints but does not help with backdating attacks (which require computing from a specific starting point).

No practical time-memory tradeoff significantly reduces the sequential computation requirement.

24.7.4. Calibration Attacks

Attacks on the calibration system:

Throttled Calibration: Adversary intentionally slows device during calibration to report lower iterations-per-second, then computes VDFs faster than claimed.

  Mitigation: Plausibility checks based on device class. Anomalously slow calibration for a known device model triggers Verifier skepticism.

Calibration Replay: Adversary reuses calibration attestation from a slower device.

  Mitigation: Device-nonce binds calibration to session. Hardware signature binds to specific device key.

Device Key Compromise: Adversary extracts hardware-bound signing key.

  Mitigation: Hardware security modules are designed to resist key extraction. This attack requires physical access and significant resources.

24.7.5. Timing Side Channels

VDF computation timing may leak information:

* Iteration Count Inference:

  Network observers may infer iteration counts from checkpoint timing. This reveals only what is already public in the evidence packet.

* Content Inference:

  VDF computation time is independent of content (fixed iteration count per checkpoint). No content leakage through timing.

VDF implementations SHOULD use constant-time hash operations where available, though timing variations in VDF computation itself do not compromise security.

25. Absence Proofs: Negative Evidence

In this section, the Absence Proofs mechanism is delineated, by which bounded claims about what did NOT occur during document creation are made possible. Unlike positive evidence (proving something happened), negative evidence is afforded by absence proofs (proving something did not happen, within defined bounds and trust assumptions). This capability extends the RATS [RFC9334] evidence model with a novel class of claims particularly suited to process attestation.

25.1. Design Philosophy

A fundamental question in process attestation is addressed by absence proofs: can meaningful claims about events that did not occur be made? The answer is nuanced and depends on carefully articulated trust boundaries, with different claim types requiring different levels of trust in the Attesting Environment.

25.1.1. The Value of Bounded Claims

Positive claims are the focus of traditional evidence systems: "X happened at time T." Extension to negative claims is afforded by absence proofs: "X did not exceed threshold Y during interval (T1, T2)." The value of bounded claims lies in their falsifiability, which distinguishes them from unbounded claims that cannot be meaningfully verified:

Positive Claim: "The author typed this document" -- difficult to verify, requires trust in the entire authoring environment.

Bounded Negative Claim: "No single edit added more than 500 characters" -- verifiable directly from the segment chain without additional trust assumptions.

The burden of proof is shifted by bounded claims: instead of claiming what DID happen (which requires comprehensive monitoring), claims about what did NOT happen are made (which can be bounded by observable evidence derived from the segment chain). This inversion of the evidentiary focus makes possible meaningful claims without comprehensive surveillance.

25.1.2. Inherent Limits of Negative Evidence

Fundamental limitations are exhibited by absence proofs that MUST be clearly communicated to Relying Parties. With respect to Monitoring Gaps, validity of absence claims is limited to monitored intervals, with gaps in monitoring creating gaps in absence guarantees. With respect to Trust Boundaries, trust in the Attesting Environment (AE) is required by some absence claims, and this trust must be explicitly documented.

With respect to Threshold Semantics, "No paste above 500 characters" does not imply "no paste"; claims are bounded, not absolute, and the specific thresholds must be considered by Relying Parties when assessing evidence. With respect to Behavioral Consistency versus Authorship, observable behavioral patterns are described by absence claims, NOT authorship, intent, or cognitive processes; consistency between declared process and observable evidence is documented, rather than claims about the identity or capabilities of the author.

25.2. Trust Boundary: Computationally Bound vs. Monitoring-Dependent

The critical architectural distinction in absence proofs is constituted by the difference between claims verifiable from the Evidence alone (trustless) and claims that require trust in the Attesting Environment's monitoring capabilities. This distinction aligns with the RATS separation between Evidence appraisal and Attesting Environment trust.

25.2.1. Computationally Bound Claims (1-15)

Verification by any party with access to the Evidence packet is made possible for computationally-bound claims. No trust in the Attesting Environment is required beyond basic data integrity, with these claims being derived purely from the segment chain structure. Independent confirmation of these claims by a Verifier is accomplished by parsing the segment chain, verifying chain integrity (hashes computed using SHA-256, MACs, VDF linkage), computing the relevant metrics from segment data, and comparing against the claimed thresholds. No interaction with the Attester or trust in its monitoring capabilities is needed for this class of claims.

25.2.2. Monitoring-Dependent Claims (16-20)

Trust that the Attesting Environment correctly observed and reported specific events is required by monitoring-dependent claims. Verification from the segment chain alone cannot be accomplished for these claims because dependence on real-time monitoring of events external to the document state is exhibited. Assessment of the following factors must be performed by the Verifier for monitoring-dependent claims: whether the capability to observe the relevant events (clipboard access, process enumeration, etc.) was possessed by the AE, whether operation with integrity during the monitoring period was maintained by the AE, whether monitoring was continuous or had gaps, and what attestation (if any) supports the AE integrity claim.

Explicit documentation of these trust assumptions is afforded by the ae-trust-basis structure, making possible informed Relying Party decisions. Hardware attestation via TPM [TPM2.0] or Secure Enclave may be used to strengthen the AE integrity claim, though such attestation is optional and its absence must be reflected in the Attestation Result caveats.

25.2.3. Trust Model Comparison

Aspect	Computationally Bound	Monitoring-Dependent
Verification	Independent, trustless	Requires AE trust
Data Source	Segment chain only	Real-time event monitoring
Confidence Basis	Cryptographic proof	AE integrity attestation
Forgery Resistance	Requires VDF recomputation	Requires AE compromise
Claim Types	1-15	16-63

Table 22

25.3. Computationally Bound Claims (Types 1-15)

Direct verification from the CBOR encoded Evidence packet without trusting the Attesting Environment's monitoring capabilities is afforded for the computationally-bound claims in the range 1-15, with verification requiring only the cryptographic primitives (SHA-256 for hash chains, HMAC for binding verification, VDF recomputation for temporal proofs) and the CDDL schema to parse the segment structures. These claims derive their truth value entirely from data present in the segment chain, with no dependency on external monitoring, making them the strongest form of evidence within the RATS architecture because they require only cryptographic verification and produce binary (PROVEN or FAILED) results.

Type	Claim	Proves	Verification Method
1	max-single-delta-chars	No single checkpoint added more than N characters	max(delta.chars-added) across all checkpoints
2	max-single-delta-bytes	No single checkpoint added more than N bytes	Derived from char counts with encoding factor
3	max-net-delta-chars	No single checkpoint had net change exceeding N chars	max(|chars-added - chars-deleted|) per checkpoint
4	min-vdf-duration-seconds	Total VDF time exceeds N seconds	sum(claimed-duration) across checkpoints
5	min-vdf-duration-per-kchar	At least N seconds of VDF time per 1000 characters	total_vdf_seconds / (final_char_count / 1000)
6	checkpoint-chain-complete	No gaps in segment sequence	Verify sequence numbers are consecutive
7	checkpoint-chain-consistent	All prev-hash values match prior tree-root	Verify hash chain linkage
8	jitter-	Captured entropy	sum(estimated-

	entropy-above-threshold	exceeds N bits	entropy-bits) from jitter-binding
9	jitter-samples-above-count	Jitter sample count exceeds N	sum(sample-count) from jitter-summary
10	revision-points-above-count	Document had at least N revision points	Count checkpoints where chars-deleted > 0
11	session-count-above-threshold	Evidence spans at least N sessions	Count distinct session boundaries in chain
12	cognitive-load-integrity	Complexity-timing correlation exceeds threshold	Spearman rank correlation between LZ complexity and jitter timing
13	intra-session-consistency	Behavioral timing remains in stable cluster (KL Divergence < delta)	Statistical distance between segment Jitter Seals
14	complexity-timing-correlation	Information Density correlates with Timing Density (rho > threshold)	Spearman rank correlation; segments with LZ Complexity < 0.2 excluded
15	reserved	Reserved for future computationally-bound claims	N/A

Table 23

25.3.1. Verification Details

For each computationally-bound claim, the Verifier performs a multi-step verification procedure that first establishes chain integrity through SHA-256 hash chain verification and VDF linkage validation, then computes the relevant metric from CBOR encoded segment data, and finally compares the observed value against the claimed threshold. The pseudocode below illustrates this procedure, with verify_chain_hashes implementing SHA-256 prev-hash verification and

verify_vdf_linkage implementing VDF entanglement verification. The key property of computationally-bound claims within the RATS architecture is that verification depends ONLY on cryptographically verifiable segment data parsed according to the CDDL schema, with no dependency on external monitoring claims or trust in the Attesting Environment's reporting accuracy.

```
 verify_chain_claim(evidence, claim):
     (1) Verify chain integrity first using SHA-256
     if not verify_chain_hashes(evidence.checkpoints):
         return INVALID("Chain integrity failure")
     if not verify_vdf_linkage(evidence.checkpoints):
         return INVALID("VDF linkage failure")

     (2) Compute the metric from CBOR segment data
     observed_value = compute_metric(evidence.checkpoints, claim.type)

     (3) Compare against threshold per CDDL schema
     match claim.type:
         case MAX_SINGLE_DELTA_CHARS:
             passes = (observed_value <= claim.threshold)
         case MIN_VDF_DURATION_SECONDS:
             passes = (observed_value >= claim.threshold)

     (4) Return verification result with cryptographic proof
     if passes:
         return PROVEN(observed_value, claim.threshold)
     else:
         return FAILED(observed_value, claim.threshold)
```

25.4. Monitoring-Dependent Claims (Types 16-63)

The claims in the range 16-63, unlike the computationally-bound claims that depend only on SHA-256 hash verification and VDF recomputation, require trust in the Attesting Environment's monitoring capabilities as documented in the ae-trust-basis field defined in the CDDL schema. Each claim documents the specific AE capability required (clipboard monitoring, process enumeration, network traffic inspection) and the basis for trusting that capability, which may range from unverified assumptions to TPM 2.0 or Secure Enclave attestation of the AE state. Within the RATS architecture, these claims represent a weaker form of evidence than computationally-bound claims because they depend on external trust relationships, but they provide valuable evidence about events (paste operations, AI tool usage, network traffic) that cannot be derived from the segment chain alone.

Type	Claim	AE Capability Required	Trust Basis
16	max-paste-event-chars	Clipboard monitoring	OS-reported paste events
17	max-clipboard-access-chars	Clipboard content access	Application-level clipboard hooks
18	no-paste-from-ai-tool	Clipboard source attribution	OS process enumeration + clipboard
20	max-insertion-rate-wpm	Real-time keystroke monitoring	Input event stream timing
21	no-automated-input-pattern	Input timing analysis	Statistical pattern recognition
22	no-macro-replay-detected	Input source verification	OS input subsystem attestation

Table 24

25.4.1. Trust Basis Documentation

Each monitoring-dependent claim MUST include an ae-trust-basis structure encoded in CBOR per the CDDL schema below, documenting the trust assumptions that underlie the claim. This explicit documentation of trust requirements is essential to the RATS architecture's goal of transparent attestation, enabling Verifiers to assess claim strength based on the trust basis rather than treating all claims uniformly. When hardware attestation via TPM 2.0 or Secure Enclave is available, the ae-trust-target field references the hardware-section for cross-verification, providing cryptographically grounded trust rather than mere assumption.

```
ae-trust-basis = {
    1 => ae-trust-target,   ; trust-target
    2 => tstr,              ; justification
    3 => bool,              ; verified
}

ae-trust-target = &(
    witnessd-software-integrity: 1,
    os-reported-events: 2,
    application-reported-events: 3,
    tpm-attested-elsewhere: 16,
    se-attested-elsewhere: 17,
    unverified-assumption: 32,
)
```

witnessd-software-integrity (1): Trust that the witnessd software itself is unmodified and correctly implements monitoring. Requires software attestation or code signing verification.

os-reported-events (2): Trust that the operating system correctly reports events (clipboard, process list, file access). Requires OS integrity.

application-reported-events (3): Trust that the authoring application correctly reports events. Weakest trust level; application may be compromised.

tpm-attested-elsewhere (16): TPM attestation of the AE state exists in the hardware-section. Cross-reference for verification.

se-attested-elsewhere (17): Secure Enclave attestation of the AE state exists in the hardware-section. Cross-reference for verification.

unverified-assumption (32): The claim is based on assumptions that cannot be verified. Relying Party must decide whether to accept based on context.

The justification field provides human-readable explanation of why the trust basis is believed adequate. The verified field indicates whether the trust basis was cryptographically verified (true) or merely assumed (false).

25.4.2. Monitoring Coverage

Honest documentation of monitoring gaps is essential for meaningful absence claims within the RATS architecture, and the monitoring-coverage structure defined in CDDL and encoded in CBOR captures the scope and limitations of AE monitoring. Unlike computationally-bound claims that can reference the complete segment chain verified through SHA-256 hash linkage, monitoring-dependent claims are only valid during periods when the relevant monitoring was active, making the coverage documentation critical for accurate confidence assessment. Timestamps in the monitoring-intervals array conform to RFC 3339 [RFC3339] format encoded using CBOR tag 1 (epoch-based date/time).

```
 monitoring-coverage = {
     1 => bool,                  ; keyboard-monitored
     2 => bool,                  ; clipboard-monitored
     3 => [+ time-interval],     ; monitoring-intervals
     4 => ratio-millibits,       ; coverage-fraction (0-1000 = 0.0-1.0)
     ? 5 => hardware-attestation, ; monitoring-attestation
 }

 time-interval = {
     1 => pop-timestamp,         ; start
     2 => pop-timestamp,         ; end
 }
```

25.4.2.1. Coverage Fields

keyboard-monitored (key 1): Boolean indicating whether keyboard input events were monitored during the session. If false, claims about typing patterns (20-22) cannot be made.

clipboard-monitored (key 2): Boolean indicating whether clipboard operations were monitored. If false, claims about paste events (16-18) cannot be made.

monitoring-intervals (key 3): Array of time intervals during which monitoring was active. Gaps between intervals represent periods where monitoring was suspended (application backgrounded, system sleep, etc.).

coverage-fraction (key 4): Fraction of total session time covered by monitoring, calculated as sum(interval_duration) / total_session_duration. Values below 0.95 indicate significant monitoring gaps that may affect absence claim confidence.

monitoring-attestation (key 5, optional): Hardware attestation that

monitoring was active during the claimed intervals. Provides stronger assurance than self-reported coverage.

25.4.2.2. Gap Semantics

Monitoring gaps have explicit semantic impact on absence claims:

* Covered Intervals:

  Absence claims apply fully during covered intervals. "No paste above 500 chars during (T1, T2)" means the AE would have detected any such paste.

* Gap Intervals:

  During gaps, monitoring-dependent claims cannot be made. An event could have occurred unobserved.

* Gap-Aware Claims:

  If coverage-fraction is below 1.0, absence claims SHOULD include a caveat noting the monitoring gap percentage.

Chain-verifiable claims (1-15) are NOT affected by monitoring gaps because they are derived from the segment chain, which has no gaps (checkpoint-chain-complete verifies this).

25.5. Absence Section Structure

The absence-section appears as an optional field (key 15) in the evidence-packet structure defined in CDDL and encoded in CBOR, contributing to the Maximum evidence tier when present. The structure contains the monitoring-coverage documentation, an array of absence-claim structures each with explicit confidence levels and trust basis documentation per the RATS transparency requirements, and an optional claim-summary that enables quick assessment of how many claims are computationally-bound (provable from SHA-256 hash chains and VDF proofs alone) versus monitoring-dependent (requiring AE trust).

```
absence-section = {
    1 => monitoring-coverage,     ; monitoring-coverage
    2 => [+ absence-claim],       ; claims
    ? 3 => claim-summary,         ; claim-summary
}

claim-summary = {
    1 => uint,                    ; computationally-bound-count
```

```
        2 => uint,                    ; monitoring-dependent-count
        3 => bool,                    ; all-claims-attested
    }

    absence-claim = {
        1 => absence-claim-type,      ; claim-type
        2 => absence-threshold,       ; threshold
        3 => absence-proof,           ; proof
        4 => absence-confidence,      ; confidence
        ? 5 => ae-trust-basis,        ; ae-trust-basis (monitoring)
    }

    absence-threshold = {
        1 => uint / null,             ; value (millibits or count, type-dependent)
    }

    absence-proof = {
        1 => absence-proof-method,    ; proof-method
        2 => absence-evidence,        ; evidence
    }

    absence-proof-method = &(
        checkpoint-chain-analysis: 1,
        keystroke-analysis: 2,
        platform-attestation: 3,
        network-attestation: 4,
        statistical-inference: 5,
    )

    absence-evidence = {
        ? 1 => [uint, uint],          ; checkpoint-range
        ? 2 => uint,                  ; max-observed-value
        ? 3 => uint,                  ; max-observed-rate-per-min (integer)
        ? 4 => tstr,                  ; statistical-test
        ? 5 => p-value-centibits,     ; p-value (0-10000 = 0.0000-1.0000)
        ? 6 => bstr,                  ; attestation-ref
    }

    absence-confidence = {
        1 => confidence-level,        ; level
        2 => [* tstr],                ; caveats
    }

    confidence-level = &(
        proven: 1,
        high: 2,
        medium: 3,
        low: 4,
```

)

25.5.1. Confidence Levels

proven (1): The claim is cryptographically provable from the Evidence. Only computationally-bound claims (1-15) can achieve this level.

high (2): Strong evidence supports the claim. For monitoring-dependent claims, requires hardware attestation of AE integrity and high monitoring coverage (>95%).

medium (3): Reasonable evidence supports the claim. AE integrity is assumed but not hardware-attested. Monitoring coverage is acceptable (>80%).

low (4): Weak evidence supports the claim. Significant caveats apply. Monitoring gaps exist or AE trust basis is unverified.

25.6. Verification Procedure

A Verifier appraises absence claims through a structured procedure that distinguishes computationally-bound from monitoring-dependent claims:

25.6.1. Step 1: Verify Computationally Bound Claims

For claims with type 1-15:

1. Verify Evidence Integrity:

   Verify segment chain hashes, VDF linkage, and structural validity per the base protocol.

2. Extract Metrics:

   Compute the relevant metric from segment data (e.g., max delta chars, total VDF duration).

3. Compare Threshold:

   Verify the computed metric satisfies the claimed threshold.

4. Assign Confidence:

   Chain-verifiable claims that pass receive confidence level "proven" (1).

25.6.2. Step 2: Appraise Monitoring-Dependent Claims

For claims with type 16-63:

1. Assess AE Trust Basis:

   Examine the ae-trust-basis for each claim. Determine whether the trust target is appropriate for the claim type and whether it was verified.

2. Evaluate Monitoring Coverage:

   Check monitoring-coverage to determine whether the relevant monitoring was active. Verify coverage-fraction is adequate for the confidence level claimed.

3. Cross-Reference Hardware Attestation:

   If ae-trust-target is tpm-attested-elsewhere (16) or se-attested-elsewhere (17), verify the corresponding attestation exists in hardware-section.

4. Evaluate Evidence:

   Examine the absence-evidence for supporting data. Statistical tests should have appropriate p-values; attestation references should be verifiable.

5. Assign Confidence:

   Based on the above factors, assign confidence level (2-4). Level 1 (proven) is NOT available for monitoring-dependent claims.

6. Document Caveats:

   Record any limitations or assumptions in the caveats array of the verification result.

25.6.3. Step 3: Produce Verification Summary

The Verifier produces a result-claim for each absence-claim examined:

```
result-claim = {
        1 => uint,                      ; claim-type
        2 => bool,                      ; verified
        ? 3 => tstr,                    ; detail
        ? 4 => confidence-level,        ; claim-confidence
}
```

25.6.4. RATS Architecture Mapping

Absence proofs extend the RATS (Remote ATtestation procedureS) evidence model in several ways:

25.6.4.1. Role Distribution

Attester Responsibility: The Attester (witnessd AE) generates absence claims based on its monitoring observations. For computationally-bound claims, the Attester merely assembles segment data in a format that enables Verifier computation. For monitoring-dependent claims, the Attester makes assertions about events it observed (or did not observe).

Verifier Responsibility: The Verifier independently verifies computationally-bound claims by recomputing metrics from Evidence. For monitoring-dependent claims, the Verifier appraises the trust basis and determines whether to accept the Attester’s monitoring assertions.

Relying Party Responsibility: The Relying Party consumes the attestation-result (.war file) and decides whether the verified claims meet their requirements. Different use cases may require different confidence levels or claim types.

25.6.4.2. Evidence Model Extension

Standard RATS evidence attests to system state (software versions, configuration). Absence proofs add a new category:

State Evidence (traditional RATS): "The system was in configuration C at time T."

Behavioral Consistency Evidence (absence proofs): "Observable behavior during interval (T1, T2) was consistent with constraint X."

This extension enables attestation about processes, not just states. The segment chain provides the evidentiary basis for process claims that would otherwise require continuous trusted monitoring.

25.6.4.3. Appraisal Policy Integration

Verifiers MAY define appraisal policies that specify:

* Which absence claim types are required for acceptance

* Minimum confidence levels for each claim type

* Required trust basis for monitoring-dependent claims

* Minimum monitoring coverage thresholds

Example policy (informative):

```
    policy:
      required_claims:
        - type: 1   # max-single-delta-chars
          threshold: 500
          min_confidence: proven
        - type: 4   # min-vdf-duration-seconds
          threshold: 3600
          min_confidence: proven
        - type: 16  # max-paste-event-chars
          threshold: 200
          min_confidence: high
          required_trust_basis: (1, 16, 17)  (SE or TPM attested)
      min_monitoring_coverage: 0.95
```

25.6.5. Security Considerations

25.6.5.1. What Absence Claims Do NOT Prove

Absence claims have explicit limits that MUST be understood by all parties:

Absence claims do NOT prove authorship: "No single edit added more than 500 characters" does not prove who performed the edits. It proves only that the observable edit pattern had this property.

Absence claims do NOT prove intent: "No paste from AI tool detected" does not prove the author intended to write without AI assistance. The author may have used AI tools in ways that evade detection.

Absence claims do NOT prove cognitive process: Behavioral patterns consistent with human typing do not prove human cognition produced the content. The claims describe observable behavior, not mental states.

Absence claims do NOT prove completeness: Claims apply only to monitored intervals. Events during monitoring gaps are not covered by absence claims.

Framing claims as "behavioral consistency" rather than "human authorship" avoids overclaiming and maintains intellectual honesty about what the evidence actually shows.

25.6.5.2. Attesting Environment Compromise

Monitoring-dependent claims are only as trustworthy as the Attesting Environment:

* Software Compromise:

  Modified witnessd software could fabricate monitoring observations. Mitigated by code signing and software attestation.

* OS Compromise:

  Compromised OS could report false clipboard contents or process lists. Mitigated by hardware attestation of OS integrity.

* Hardware Compromise:

  Physical access to device could enable hardware-level attacks. This is outside the threat model for most use cases.

The ae-trust-basis structure explicitly documents which trust assumptions apply, enabling Relying Parties to make informed decisions about acceptable risk.

25.6.5.3. Monitoring Evasion

Sophisticated adversaries may attempt to evade monitoring:

Timing-Based Evasion: Performing prohibited actions during monitoring gaps. Mitigated by high coverage requirements and gap documentation.

Tool-Based Evasion: Using tools not in the detection list (e.g., novel to known tools; unknown tools may evade detection.

Channel-Based Evasion: Using alternative input channels (screen readers, accessibility features) not monitored by the AE. Mitigated by comprehensive input monitoring.

Simulation: Generating input patterns that mimic human behavior. The jitter-seal and VDF mechanisms make this costly but not impossible. See forgery-cost-section.

Absence proofs do not claim to make evasion impossible, only to make it costly and to document the monitoring coverage that was actually achieved.

25.6.5.4. Statistical Claim Limitations

Claims based on statistical inference (proof-method 5) have inherent uncertainty:

* p-values indicate probability, not certainty
* Multiple testing increases false positive risk
* Adversarial inputs may exploit statistical assumptions

Statistical claims SHOULD be assigned confidence level "medium" (3) or "low" (4) unless supported by additional evidence.

25.6.6. Privacy Considerations

Absence claims may reveal information about the authoring process:

* Edit Pattern Disclosure:

  Chain-verifiable claims reveal aggregate statistics about edit sizes and frequencies. This is inherent in the segment chain and cannot be hidden without removing the evidentiary basis for claims.

* Tool Usage Disclosure:

  that the AE was monitoring for AI tool usage. Users should be informed of this monitoring.

* Behavioral Fingerprinting:

  Detailed jitter data and monitoring observations could theoretically enable behavioral fingerprinting. The histogram aggregation in jitter-binding mitigates this for timing data.

Users SHOULD be informed which absence claims will be generated and have the option to disable specific monitoring capabilities if privacy concerns outweigh the value of those claims.

26. Forgery Cost Bounds (Quantified Security)

This section defines the forgery cost bounds mechanism, which provides quantified security analysis for Proof of Process evidence. Rather than claiming evidence is "secure" or "insecure" in absolute terms, this framework expresses security as minimum resource costs that an adversary must expend to produce counterfeit evidence.

26.1. Design Philosophy

Traditional security claims are often binary: a system is either "secure" or "broken." This framing poorly serves attestation scenarios where:

* Adversary capabilities vary across resource levels
* Evidence value degrades gracefully rather than failing completely
* Relying Parties have different risk tolerances
* Hardware costs and computational speeds change over time

The Proof of Process framework adopts quantified security: expressing security guarantees in terms of measurable costs (time, entropy, economic resources) that bound adversary capabilities.

26.1.1. Quantified Forgery Cost Bounds

Forgery requires simulating the D_i <-> tau_i alignment during sequential VDF computation. This imposes a computational cost of O(n * VDF_iters), where n is the number of segments. Achieving psycholinguistic fidelity requires high-latency semantic processing synchronized with the VDF chain. Simulating the Error Topology (H=0.7, rho=0.8) within the sequential VDF phases requires approximately 10^3 trials per segment using a biological motor-skill model, further increasing the search space for forgery.

Bound: A 1-hour human authoring session generates approximately 10^12 hardware cycles (@ 4GHz). A bot must expend equivalent sequential cycles without the benefit of parallelism to produce a valid correlation proof.

26.1.2. What Forgery Cost Bounds Do NOT Claim

Forgery cost bounds explicitly avoid claims that evidence is:

* *Unforgeable:* Given sufficient resources, any evidence can be forged. The bounds quantify "sufficient."
* *Guaranteed authentic:* Bounds express minimum forgery costs, not maximum. Cheaper attacks may exist that have not been discovered.
* *Irrefutable proof:* Evidence supports claims with quantified confidence, not mathematical certainty.

* *Permanent:* Cost bounds depreciate as hardware improves. Evidence verified today may have different bounds when re-evaluated in the future.

26.2. Forgery Cost Section Structure

The forgery-cost-section appears in each evidence packet and contains four required components:

```
forgery-cost-section = {
    1 => time-bound,           ; time-bound
    2 => entropy-bound,        ; entropy-bound
    3 => economic-bound,       ; economic-bound
    4 => security-statement,   ; security-statement
}
```

These components represent orthogonal dimensions of forgery cost. A complete security assessment considers all four dimensions.

26.3. Time Bound

The time-bound quantifies the minimum wall-clock time required to recompute the VDF chain, establishing a lower bound on forgery duration that an adversary must necessarily expend regardless of computational resources available, with the bound being enforced through the inherent sequentiality of VDF constructions where each iteration depends cryptographically on the previous output computed via SHA-256 or similar hash functions, ensuring that even adversaries with parallel processing capabilities cannot reduce the wall-clock time required to forge evidence chains within the RATS architecture.

```
time-bound = {
    1 => uint,                 ; total-iterations
    2 => uint,                 ; calibration-rate
    3 => tstr,                 ; reference-hardware
    4 => uint,                 ; min-recompute-seconds (integer seconds)
    5 => bool,                 ; parallelizable
    ? 6 => uint,               ; max-parallelism
}
```

26.3.1. Field Definitions

The time-bound structure, encoded in CBOR according to the CDDL schema above, contains six fields that together quantify the temporal cost of forgery within the RATS evidence framework. The total-iterations field (key 1) represents the sum of all VDF iterations across all checkpoints in the evidence packet, computed as sum(checkpoint{i}.vdf-proof.iterations) for all i, providing the raw

count of sequential hash operations using SHA-256 that must be recomputed. The calibration-rate field (key 2) contains the attested iterations-per-second from the calibration attestation, representing the maximum VDF computation speed on the Attesting Environment's hardware as measured through TPM 2.0 or similar hardware attestation mechanisms. The reference-hardware field (key 3) provides a human-readable description of the hardware used for calibration (e.g., "Apple M2 Pro", "Intel i9-13900K"), used for plausibility assessment rather than cryptographic verification. The min-recompute-seconds field (key 4) specifies the minimum wall-clock seconds required to recompute the VDF chain on reference hardware, calculated as total-iterations divided by calibration-rate, representing a lower bound since actual recomputation on slower hardware takes longer. The parallelizable field (key 5) is a boolean indicating whether the VDF algorithm permits parallelization, with iterated hash VDFs using SHA-256 (algorithms 1-15) always reporting false due to inherent sequentiality, while certain succinct VDF constructions may permit limited parallelization. The optional max-parallelism field (key 6) specifies the maximum parallel speedup factor when parallelizable is true, remaining absent for iterated hash VDFs that enforce strict sequential computation.

26.3.2. Time Bound Verification

A Verifier within the RATS architecture computes and validates the time bound through a systematic procedure that ensures the claimed temporal costs are mathematically consistent with the VDF proofs embedded in the evidence chain. First, the Verifier traverses all checkpoints encoded in CBOR and sums the iterations field from each VDF proof, accumulating the total sequential hash operations using SHA-256 that comprise the evidence chain. Second, if calibration attestation is present (typically signed via COSE and backed by TPM 2.0 hardware attestation), the Verifier validates the hardware signature and checks that calibration-rate matches the attested iterations-per-second from the trusted hardware module. Third, the Verifier computes the minimum time by dividing total-iterations by calibration-rate and verifies that the result matches min-recompute-seconds within floating-point tolerance, confirming mathematical consistency of the VDF chain. Fourth, the Verifier performs a plausibility check to ensure min-recompute-seconds is consistent with the claimed authoring duration indicated by RFC 3339 timestamps, since significant discrepancy (e.g., a 10-hour claimed session with only 1-minute VDF time) indicates either misconfiguration of the Attesting Environment or potential manipulation of the evidence packet.

26.3.3. Parallelization Resistance

The security of time bounds within the RATS architecture depends critically on VDF parallelization resistance as established in the cryptographic literature, which provides formal proofs that sequential computation cannot be accelerated through parallel hardware deployment. For iterated hash VDFs using SHA-256, each iteration depends cryptographically on the previous output through the hash function's one-way property, no known technique computes H^n(x) faster than n sequential hash operations due to the preimage resistance of SHA-256, and an adversary with P processors fundamentally cannot compute the chain P times faster because the inherent data dependency between iterations prevents parallelization. This property ensures that time bounds reflect wall-clock time rather than aggregate compute time, meaning an adversary with access to an entire data center cannot forge 10 hours of evidence in 10 minutes by deploying 60x more processors, since the sequential VDF chain must still be computed one iteration at a time regardless of available parallel resources. See Section 24.7.2 for detailed analysis of parallelization resistance in each VDF algorithm supported by this RATS profile.

26.4. Entropy Bound

The entropy-bound quantifies the unpredictability in the evidence chain as captured through the Jitter Seal behavioral entropy mechanism, establishing a lower bound on the probability of guessing or replaying entropy commitments that are bound to the VDF chain through HMAC-SHA256 [RFC2104] commitments. Within the RATS architecture, this entropy bound represents the accumulated behavioral randomness from human input patterns that an adversary would need to predict or reproduce in order to forge authentic-appearing evidence, with the CBOR encoding following the CDDL schema specified below.

```
 entropy-bound = {
     1 => entropy-decibits,     ; total-entropy (decibits, /10 for bits)
     2 => uint,                 ; sample-count
     3 => entropy-decibits,     ; entropy-per-sample (decibits)
     4 => uint,                 ; brute-force-log2 (negative exponent, e.g., 64 = 2^-6
4)
     5 => bool,                 ; replay-possible
     ? 6 => tstr,               ; replay-prevention
 }
```

26.4.1. Field Definitions

The entropy-bound structure, encoded in CBOR according to the CDDL schema, contains six fields that together quantify the unpredictability barrier facing an adversary attempting to forge evidence within the RATS framework. The total-entropy-bits field (key 1) represents the aggregate entropy across all Jitter Seals in the evidence packet expressed in bits, computed as sum(jitter-summary[i].estimated-entropy-bits) for all i where each Jitter Seal captures behavioral timing bound via HMAC-SHA256. The sample-count field (key 2) contains the total number of timing samples captured across all Jitter Seals, with higher sample counts increasing confidence in the Min-Entropy (H_min) estimate derived from the timing histogram. The entropy-per-sample field (key 3) represents the average entropy contribution per timing sample calculated as total-entropy-bits divided by sample-count, with typical human typing contributing 2-4 bits per inter-key interval based on motor timing variance. The brute-force-probability field (key 4) quantifies the probability of successfully guessing the entropy commitment by brute force, calculated as 2^(-total-entropy-bits), yielding approximately 5.4 x 10^-20 for 64 bits of entropy. The replay-possible field (key 5) is a boolean indicating whether Jitter Seal replay is theoretically possible, set to false when VDF entanglement is properly configured such that the HMAC entropy commitment appears in the VDF input chain. The optional replay-prevention field (key 6) provides a human-readable description of replay prevention mechanisms, typically containing values such as "VDF entanglement with prev-checkpoint binding using SHA-256".

26.4.2. Entropy Bound Verification

A Verifier within the RATS architecture computes and validates the entropy bound through a systematic five-step procedure that ensures the claimed entropy costs are mathematically consistent with the Jitter Seal commitments embedded in the CBOR evidence chain. First, the Verifier aggregates entropy by summing estimated-entropy-bits from each checkpoint’s jitter-summary and verifying that the total matches the claimed total-entropy-bits field, ensuring no entropy claims have been inflated beyond what the underlying HMAC-SHA256 commitments support. Second, the Verifier counts samples by summing sample-count from each jitter-summary and verifying consistency with the claimed sample-count, confirming the behavioral timing data volume matches expectations for the authoring session duration indicated by RFC 3339 timestamps. Third, if raw-intervals are disclosed for transparency, the Verifier recomputes the histogram and Min-Entropy (H_min) independently, verifying consistency with the claimed entropy estimate to detect potential manipulation of entropy calculations. Fourth, the Verifier checks replay prevention by

verifying that each HMAC entropy-commitment appears in the corresponding VDF input per the VDF chain construction, setting replay-possible to true if VDF entanglement is absent since unentangled entropy commitments could theoretically be replayed from previous sessions. Fifth, the Verifier computes brute-force probability by calculating 2^(-total-entropy-bits) and verifying that the result matches the claimed brute-force-probability within floating-point tolerance, confirming the security bound is mathematically accurate for the accumulated behavioral entropy.

26.4.3. Minimum Entropy Requirements

The RATS profile defined in this specification establishes RECOMMENDED minimum entropy thresholds by evidence tier, with thresholds calibrated to provide meaningful security guarantees against brute-force attacks on the HMAC-SHA256 entropy commitments embedded in Jitter Seals. The Basic tier requires a minimum of 32 bits of total entropy, corresponding to a brute-force probability less than 2.3 x 10^-10, suitable for low-stakes evidence where moderate forgery resistance suffices. The Standard tier requires a minimum of 64 bits of total entropy, corresponding to a brute-force probability less than 5.4 x 10^-20, providing strong forgery resistance appropriate for most professional and academic authorship attestation use cases. The Enhanced tier requires a minimum of 128 bits of total entropy, corresponding to a brute-force probability less than 2.9 x 10^-39, offering cryptographically strong guarantees approaching the security level of the underlying SHA-256 hash function for high-stakes evidence requiring maximum assurance. Evidence packets encoded in CBOR that fail to meet the minimum entropy thresholds for their claimed tier SHOULD be flagged in the security-statement caveats, enabling Relying Parties to make informed trust decisions within the RATS architecture about whether the behavioral entropy is sufficient for their risk tolerance.

26.5. Economic Bound

The economic-bound translates time requirements derived from VDF chains and entropy requirements captured through HMAC-SHA256 Jitter Seals into monetary costs, enabling Relying Parties within the RATS architecture to assess forgery feasibility in concrete economic terms that can be compared against the potential value of forgery. The CBOR encoding follows the CDDL schema specified below, providing a standardized representation of cost estimates that can be independently verified and compared across different evidence packets.

```
    economic-bound = {
        1 => tstr,                ; cost-model-version
        7 => tstr,                ; oracle-uri (Signed Pricing Feed)

        2 => pop-timestamp,       ; cost-model-date
        3 => cost-estimate,       ; compute-cost
        4 => cost-estimate,       ; time-cost
        5 => cost-estimate,       ; total-min-cost
        6 => cost-estimate,       ; cost-per-hour-claimed
    }

    cost-estimate = {
        1 => cost-microdollars,   ; usd (microdollars, /1000000 for USD)
        2 => cost-microdollars,   ; usd-low
        3 => cost-microdollars,   ; usd-high
        4 => tstr,                ; basis
    }
```

26.5.1. Field Definitions

The economic-bound structure, encoded in CBOR according to the CDDL schema, contains six fields that translate the cryptographic and temporal costs of VDF chain recomputation into monetary terms for Relying Party assessment within the RATS framework. The cost-model-version field (key 1) contains an identifier for the cost model used (e.g., "witnessd-cost-2025Q1"), with versioning necessary because hardware prices and computational costs for SHA-256 operations change over time. The cost-model-date field (key 2) contains an RFC 3339 timestamp when the cost model was established. The compute-cost field (key 3) quantifies the cost of computational resources required to recompute the VDF chain, including cloud compute instance cost for min-recompute-seconds of sequential SHA-256 operations, electricity cost for sustained computation, and amortized hardware cost if using dedicated equipment rather than cloud resources. The time-cost field (key 4) represents the opportunity cost of the wall-clock time required for forgery, since an adversary attempting to forge 10-hour evidence cannot use that time for other purposes, modeled as the economic value of the adversary's time at skilled labor rates. The total-min-cost field (key 5) represents the minimum total cost to forge the evidence combining compute and time costs, serving as the primary metric for cost-benefit analysis by Relying Parties. The cost-per-hour-claimed field (key 6) normalizes forgery cost by claimed authoring duration (calculated as total-min-cost divided by claimed-duration-hours derived from RFC 3339 timestamps), enabling fair comparison across evidence packets of different lengths within the RATS trust framework.

26.5.2. Cost Estimate Structure

Each cost-estimate structure within the economic-bound, encoded in CBOR according to the CDDL schema, includes a point estimate and confidence range to account for uncertainty in adversary resource access within the RATS trust model. The usd field (key 1) contains the point estimate in US dollars, representing the expected cost under typical assumptions about cloud compute pricing and electricity rates for sustaining the sequential SHA-256 operations required by VDF recomputation. The usd-low field (key 2) contains the lower bound of a 90% confidence interval, representing cost assuming the adversary has access to discounted resources such as pre-existing infrastructure, bulk compute contracts, or subsidized electricity that reduce marginal costs. The usd-high field (key 3) contains the upper bound of the 90% confidence interval, representing cost assuming the adversary must acquire resources at full market rates without existing infrastructure or preferential pricing arrangements. The basis field (key 4) contains a human-readable description of the cost calculation basis (e.g., "AWS c7i.large @ $0.085/hr + $0.10/kWh electricity"), enabling Relying Parties to assess whether the cost model assumptions are reasonable for their deployment context and adjust estimates accordingly based on their knowledge of adversary capabilities.

26.5.3. Cost Computation

The reference cost computation for compute-cost quantifies the resources required to recompute the VDF chain with its sequential SHA-256 iterations, using the formula: hourly_rate = cloud_rate + elec_rate * power, where cloud_rate represents the cost of compute instances capable of sustained hashing, compute_hours = min_recompute_seconds / 3600 converts the VDF recomputation time to billable hours, and compute_cost_usd = hourly_rate * compute_hours yields the total computational expenditure. The 90% confidence interval assumes 50% rate variance to account for differences in adversary resource access, computed as compute_cost_low = compute_cost_usd * 0.5 for adversaries with discounted access and compute_cost_high = compute_cost_usd * 1.5 for those paying market rates.

The reference cost computation for time-cost represents the opportunity cost of wall-clock time required for sequential VDF recomputation, using a skilled labor rate model where hourly_value = 50.0 USD and time_cost_usd = hourly_value * (min_recompute_seconds / 3600), reflecting the economic value of the adversary's time that cannot be used for other purposes during the forgery attempt. The confidence interval for time cost accounts for labor rate variance, computed as time_cost_low = time_cost_usd * 0.2 for adversaries in low-cost labor markets and time_cost_high = time_cost_usd * 4.0 for highly skilled adversaries whose time commands premium rates.

These are reference calculations within the RATS framework, and implementations MAY use different cost models appropriate to their deployment context, provided the CBOR encoding follows the CDDL schema and the basis field documents the alternative model for Relying Party assessment.

26.6. Security Statement

The security-statement provides a formal claim about evidence security within the RATS architecture, including explicit assumptions about VDF parallelization resistance, SHA-256 preimage resistance, and HMAC binding security, along with caveats that limit the scope of the security claim. The CBOR encoding follows the CDDL schema specified below, providing machine-readable security bounds that Relying Parties can evaluate against their policy requirements while also offering human-readable claims suitable for non-technical stakeholders.

```
security-statement = {
    1 => tstr,                  ; claim
    2 => formal-security-bound, ; formal
    3 => [+ tstr],              ; assumptions
    4 => [* tstr],              ; caveats
}

formal-security-bound = {
    1 => uint,                  ; min-seconds (integer seconds)
    2 => entropy-decibits,      ; min-entropy (decibits, /10 for bits)
    3 => cost-microdollars,     ; min-cost (microdollars, /1000000 for USD)
}
```

26.6.1. Field Definitions

The security-statement structure, encoded in CBOR according to the CDDL schema, contains four fields that together provide both human-readable and machine-readable security claims within the RATS trust framework. The claim field (key 1) contains a human-readable security claim that MUST be phrased as a minimum bound rather than an absolute guarantee (e.g., "Forging this evidence requires at minimum 8.3 hours of sequential VDF computation, 67 bits of HMAC entropy prediction, and an estimated $42-$126 in resources"), avoiding language that implies unforgeable or irrefutable guarantees. The formal field (key 2) contains machine-readable security bounds for automated policy evaluation, enabling Relying Parties to programmatically compare evidence packets against their minimum acceptance thresholds without parsing natural language claims. The assumptions field (key 3) contains an array of assumptions under which the security claim holds, which MUST include at minimum a cryptographic assumption (e.g., "SHA-256 preimage resistance"), a hardware assumption (e.g., "TPM 2.0 calibration attestation is accurate"), and an adversary model assumption (e.g., "Adversary cannot parallelize VDF computation"), making explicit the conditions that must hold for the bounds to remain valid. The caveats field (key 4) contains an array of limitations or warnings about the security claim, with typical examples including "Cost estimates based on 2024Q4 cloud pricing", "Entropy estimate assumes timing samples are statistically independent", and "Does not protect against Attesting Environment compromise during evidence generation", enabling Relying Parties to understand the boundaries of the security guarantees.

26.6.2. Formal Security Bound

The formal-security-bound structure, encoded in CBOR according to the CDDL schema, provides three orthogonal minimum requirements for forgery that an adversary must simultaneously overcome to produce fraudulent evidence within the RATS architecture. The min-seconds field (key 1) specifies the minimum wall-clock seconds to forge the evidence, derived from time-bound.min-recompute-seconds which itself reflects the sequential VDF chain recomputation time using SHA-256 iterations that cannot be parallelized. The min-entropy-bits field (key 2) specifies the minimum entropy bits an adversary must predict or generate, derived from entropy-bound.total-entropy-bits which reflects the accumulated behavioral entropy captured through HMAC-SHA256 Jitter Seal commitments. The min-cost-usd field (key 3) specifies the minimum cost in USD to forge the evidence, conservatively derived from economic-bound.total-min-cost.usd-low to provide a lower bound that holds even if the adversary has discounted resource access.

Relying Parties within the RATS trust framework can evaluate these CBOR encoded bounds against their risk tolerance through automated policy evaluation. For example, a policy might require: accept_evidence if min-seconds >= 3600 (requiring at least 1 hour of sequential VDF computation) AND min-entropy-bits >= 64 (requiring at least 64 bits of HMAC entropy prediction) AND min-cost-usd >= 100 (requiring at least $100 in forgery resources), with all three conditions enforced simultaneously to provide defense-in-depth against different adversary capabilities.

26.7. Verification Procedure

A Verifier within the RATS architecture computes and validates forgery cost bounds through a systematic six-step procedure that ensures the claimed security guarantees are mathematically consistent with the cryptographic evidence embedded in the CBOR encoded evidence packet. First, the Verifier computes the time bound by summing VDF iterations across all checkpoints using SHA-256 hash chain verification, retrieving calibration-rate from the COSE signed calibration attestation backed by TPM 2.0 or similar hardware, and computing min-recompute-seconds = total-iterations / calibration-rate to establish the temporal forgery barrier. Second, the Verifier computes the entropy bound by aggregating Min-Entropy (H_min) estimates from all Jitter Seals with their HMAC-SHA256 commitments, verifying VDF entanglement for each seal to confirm replay prevention, and computing brute-force probability as 2^(-total-entropy-bits) to quantify the prediction difficulty. Third, the Verifier computes the economic bound by applying the cost model to the time bound, computing confidence intervals based on assumed adversary resource access, and normalizing by claimed duration derived from RFC 3339 timestamps to enable fair comparison across evidence packets.

Fourth, the Verifier constructs the security statement by generating a human-readable claim that describes the minimum VDF recomputation time, HMAC entropy bits, and USD cost required for forgery, populating the formal-security-bound fields for automated policy evaluation, listing applicable cryptographic assumptions about SHA-256 and VDF security, and adding any relevant caveats about cost model staleness or entropy estimation limitations. Fifth, the Verifier validates claimed bounds by comparing the computed bounds against those claimed in the CBOR encoded evidence packet and flagging discrepancies exceeding tolerance, which may indicate either computational errors or potential manipulation of the forgery cost claims.

The Verifier MAY recompute bounds using its own cost model rather than accepting the Attester's claimed bounds encoded in the CDDL schema, and independent recomputation is RECOMMENDED for high-stakes verification within the RATS trust framework where the consequences of accepting forged evidence are significant.

26.8. Security Considerations

26.8.1. Assumed Adversary Capabilities

Forgery cost bounds within the RATS architecture assume an adversary with specific capabilities that bound the security guarantees provided by VDF chains and HMAC entropy commitments. The assumed adversary has access to commodity hardware at market prices for computing SHA-256 hash iterations, can execute VDF algorithms correctly following the published specifications, cannot parallelize inherently sequential VDFs due to the data dependency between iterations, cannot predict behavioral entropy in advance because human input timing exhibits genuine behavioral randomness, and has not compromised the Attesting Environment during evidence generation such that key material or intermediate state remains protected.

The forgery cost bounds encoded in CBOR may not hold against adversaries who exceed these assumed capabilities within the RATS threat model. Adversaries with access to specialized SHA-256 ASICs at below-market cost may achieve lower compute costs than the economic bound assumes, reducing the effective forgery barrier. Adversaries who can compromise the Attesting Environment during evidence generation may extract key material or manipulate VDF computations, bypassing the sequential computation requirement entirely. Adversaries who discover novel cryptanalytic attacks on VDF constructions or hash function security may reduce the effective security below what the bounds indicate. Adversaries with access to quantum computers capable of breaking the cryptographic assumptions underlying SHA-256 preimage resistance may invalidate the security guarantees, though such computers do not currently exist at the scale required for this attack.

26.8.2. Limitations of Cost Bounds

Forgery cost bounds within the RATS architecture provide lower bounds rather than absolute guarantees, with several fundamental limitations that Relying Parties must understand when evaluating CBOR encoded evidence packets. The bounds assume current best-known attacks on VDF constructions and SHA-256 hash functions, meaning future cryptanalytic advances may reduce actual forgery costs below what the security-statement claims, requiring periodic reassessment of evidence security as the cryptographic landscape evolves. The

economic estimates depend entirely on cost model assumptions encoded in the CDDL schema, and actual adversary costs may differ significantly based on their specific resource access, geographic location affecting electricity costs, or existing computational infrastructure that reduces marginal costs. The Min-Entropy (H_min) estimates from Jitter Seals assume statistically independent timing samples, but correlations in human input timing data (such as rhythmic typing patterns or predictable pause structures) may reduce effective entropy below the claimed HMAC commitment strength. The time bounds depend critically on calibration accuracy from TPM 2.0 or similar hardware attestation, and without cryptographic hardware attestation the calibration is self-reported by the Attesting Environment and may be manipulated to overstate VDF computation speed, inflating the apparent time bound.

26.8.3. What Bounds Do NOT Guarantee

Forgery cost bounds within the RATS architecture explicitly do NOT provide certain guarantees that Relying Parties might incorrectly infer from the security claims encoded in CBOR. The bounds do not provide authenticity proof: evidence meeting VDF time thresholds and HMAC entropy thresholds is proven expensive to forge rather than proven authentic, and these are fundamentally distinct claims that must not be conflated in Relying Party policy decisions. The bounds do not provide content verification: the forgery cost analysis using SHA-256 chains says nothing about document content, quality, accuracy, or truthfulness, since only the process evidence describing how the document evolved is bounded rather than the document's substantive claims. The bounds do not provide intent attribution: the COSE signatures and VDF proofs do not prove who created the evidence or why they created it, since identity and intent attribution are outside the scope of cost-asymmetric forgery analysis and require separate attestation mechanisms.

26.8.4. Policy Guidance for Relying Parties

Relying Parties within the RATS architecture should establish evidence acceptance policies based on four key considerations that translate forgery cost bounds encoded in CBOR into actionable trust decisions. First, risk assessment: What is the cost of accepting forged evidence with manipulated VDF proofs or fabricated HMAC entropy commitments? High-stakes decisions such as legal proceedings, academic credential verification, or financial attestation require proportionally higher cost thresholds in the formal-security-bound to ensure forgery is economically irrational. Second, adversary economics: Would forgery be economically rational given the costs quantified in the economic-bound structure? If VDF recomputation costs using SHA-256 iterations exceed the potential

gain from successful forgery, rational adversaries operating within standard economic models will not attempt it, though irrational or ideologically motivated adversaries may still pose risks. Third, time sensitivity: How quickly must evidence be verified given the RFC 3339 timestamps in the evidence packet? Fourth, corroborating evidence: Cost bounds derived from VDF chains and Jitter Seals are one factor among many in the trust decision, and external anchors such as RFC 3161 timestamps or blockchain anchors, TPM 2.0 hardware attestation, and contextual information about the Attesting Environment all contribute to overall confidence within the RATS trust framework.

27. Cross-Document Provenance Links

This section defines a mechanism for establishing cryptographic relationships between Evidence packets within the RATS [RFC9334] architecture, with provenance links encoded in CBOR [RFC8949] according to CDDL [RFC8610] schemas that enable cross-document attestation. Provenance links enable authors to cryptographically prove that one document evolved from, merged with, or was derived from other documented works by referencing the SHA-256 [RFC6234] chain hashes and UUID [RFC9562] identifiers of parent evidence packets, creating a verifiable derivation graph that maintains the tamper-evidence properties of the underlying VDF chains while extending attestation across document boundaries.

27.1. Motivation

Real-world authorship rarely occurs in isolation, and the RATS architecture must accommodate the complex evolution patterns that characterize genuine creative and scholarly work. Documents evolve through multiple stages where research notes with their own VDF chains and HMAC entropy commitments become draft papers with additional SHA-256 segment-based Merkle trees which in turn become published articles with final attestation, multiple contributors merge their independently-attested sections (each with distinct COSE signatures) into a collaborative work requiring unified provenance tracking, thesis chapters are extracted and expanded into standalone papers that should cryptographically reference their source material via UUID links, and codebases are forked with their evidence packets serving as the basis for derivative works that need verifiable connection to their origins.

Without provenance links, each Evidence packet encoded in CBOR is cryptographically isolated despite representing interconnected creative work. An author cannot prove that their final manuscript evolved from the lab notes they documented six months earlier, even though both have valid VDF proofs and Jitter Seal entropy commitments

using HMAC-SHA256, because the evidence packets lack cryptographic linkage. Provenance links provide this capability within the RATS framework while maintaining the privacy and security properties of the underlying evidence model, enabling Relying Parties to verify not only individual evidence packets but also the derivation relationships between them.

27.2. Provenance Section Structure

The provenance section is an optional component of the Evidence packet encoded in CBOR, identified by integer key 20 in the CDDL schema and signed via COSE as part of the overall evidence envelope. When present, it documents the cryptographic relationship between the current Evidence packet and one or more parent packets by referencing their UUID identifiers and SHA-256 chain hashes, enabling Relying Parties within the RATS architecture to verify derivation claims by fetching and validating the referenced parent evidence with its VDF proofs and HMAC entropy commitments.

```
; Provenance section for cross-document linking
; Key 20 in evidence-packet
provenance-section = {
    ? 1 => [+ provenance-link],     ; parent-links
    ? 2 => [+ derivation-claim],    ; derivation-claims
    ? 3 => provenance-metadata,     ; metadata
}

; Link to a parent Evidence packet
provenance-link = {
    1 => uuid,                       ; parent-packet-id
    2 => hash-value,                 ; parent-chain-hash
    3 => derivation-type,           ; how this document relates
    4 => pop-timestamp,             ; when derivation occurred
    ? 5 => tstr,                    ; relationship-description
    ? 6 => [+ uint],                ; inherited-checkpoints
    ? 7 => cose-signature,          ; cross-packet-attestation
}

; Type of derivation relationship
derivation-type = &(
    continuation: 1,                 ; same work, new packet
    merge: 2,                        ; from multiple sources
    split: 3,                        ; Extracted from larger work
    rewrite: 4,                      ; Substantial revision
    translation: 5,                  ; Language translation
    fork: 6,                         ; independent branch
    citation-only: 7,                ; references only
)
```

```
; Claims about what was derived and how
derivation-claim = {
    1 => derivation-aspect,          ; what-derived
    2 => derivation-extent,          ; extent
    ? 3 => tstr,                     ; description
    ? 4 => uint .le 100,             ; estimated-percentage (0-100)
}

derivation-aspect = &(
    structure: 1,                    ; Document organization
    content: 2,                      ; Textual content
    ideas: 3,                        ; Conceptual elements
    data: 4,                         ; Data or results
    methodology: 5,                  ; Methods or approach
    code: 6,                         ; Source code
)

derivation-extent = &(
    none: 0,                         ; Not derived
    minimal: 1,                      ; Less than 10%
    partial: 2,                      ; 10-50%
    substantial: 3,                  ; 50-90%
    complete: 4,                     ; More than 90%
)

; Optional metadata about provenance
provenance-metadata = {
    ? 1 => tstr,                     ; provenance-statement
    ? 2 => bool,                     ; all-parents-available
    ? 3 => [+ tstr],                 ; missing-parent-reasons
}
```

27.3. Verification of Provenance Links

Verifiers MUST perform the following checks when provenance links are present:

27.3.1. Parent Chain Hash Verification

For each provenance-link, if the parent Evidence packet is available:

1. Verify that parent-packet-id matches the packet-id field of the parent Evidence packet.

2. Verify that parent-chain-hash matches the tree-root of the final checkpoint in the parent Evidence packet.

3. Verify that the derivation timestamp is not earlier than the created timestamp of the parent packet.

If the parent Evidence packet is not available, the Verifier SHOULD note this limitation in the Attestation Result caveats. The provenance link remains valid but unverified.

27.3.2. Cross-Packet Attestation

When cross-packet-attestation is present, it provides cryptographic proof that the author of the current packet had access to the parent packet at the time of derivation:

```
cross-packet-attestation = COSE_Sign1(
    payload = CBOR_encode({
        1: current-packet-id,
        2: parent-packet-id,
        3: parent-chain-hash,
        4: derivation-timestamp,
    }),
    key = author-signing-key
)
```

This attestation prevents retroactive provenance claims where an author discovers an existing Evidence packet and falsely claims derivation after the fact.

27.4. Privacy Considerations for Provenance

Provenance links may reveal information about the author’s creative process and document history. Authors SHOULD consider:

* Parent packet IDs are disclosed to anyone with access to the child packet.

* If parent packets use the author-salted hash mode, the salt MUST be shared for full verification.

* Derivation claims may reveal collaboration patterns or research relationships.

Authors MAY choose to omit provenance links for privacy while still maintaining independent Evidence for each document.

27.5. Provenance Link Examples

27.5.1. Continuation Example

A dissertation written over 18 months with monthly Evidence exports:

```
        {1: 1, 2: 3, 3: "Structure from Alice's draft"},
        {1: 2, 2: 2, 3: "Content merged from all three"},
        {1: 4, 2: 4, 3: "Data primarily from Bob"}
      ]
    }
```

28. Incremental Evidence with Continuation Tokens

This section defines a mechanism for producing Evidence packets incrementally over extended authoring periods. Continuation tokens allow a single logical authorship effort to be documented across multiple Evidence packets without losing cryptographic continuity.

28.1. Motivation for Continuation Tokens

Long-form works such as novels, dissertations, or technical books may span months or years of active authorship. Capturing all Evidence in a single packet presents practical challenges:

* Unbounded segment-based Merkle trees consume storage and increase verification time.

* Authors may need to share partial Evidence before work completion (e.g., chapter submissions, progress reports).

* System failures or device changes could result in loss of accumulated Evidence.

* Privacy requirements may dictate periodic Evidence export and local data deletion.

Continuation tokens address these challenges by enabling cryptographically-linked Evidence packet chains while preserving independent verifiability of each packet.

28.2. Continuation Token Structure

The continuation token is an optional component of the Evidence packet, identified by integer key 21. It establishes the packet's position within a multi-packet Evidence series.

```
; Continuation token for multi-packet Evidence series
; Key 21 in evidence-packet
continuation-section = {
    1 => uuid,                      ; series-id
    2 => uint,                      ; packet-sequence
    ? 3 => hash-value,              ; prev-packet-chain-hash
    ? 4 => uuid,                    ; prev-packet-id
    5 => continuation-summary,      ; cumulative-summary
    ? 6 => cose-signature,          ; series-binding-signature
}

; Cumulative statistics across the series
continuation-summary = {
    1 => uint,                      ; total-checkpoints-so-far
    2 => uint,                      ; total-chars-so-far
    3 => duration,                  ; total-vdf-time-so-far
    4 => entropy-decibits,          ; total-entropy-so-far (decibits)
    5 => uint,                      ; packets-in-series
    ? 6 => pop-timestamp,           ; series-started-at
    ? 7 => duration,                ; total-elapsed-time
}
```

Key semantics:

series-id: A UUID that remains constant across all packets in the series. Generated when the first packet in the series is created.

packet-sequence: Zero-indexed sequence number. The first packet in a series has packet-sequence = 0.

prev-packet-chain-hash: The tree-root of the final checkpoint in the previous packet. MUST be present for packet-sequence > 0. MUST NOT be present for packet-sequence = 0.

prev-packet-id: The packet-id of the previous packet in the series. SHOULD be present for packet-sequence > 0 to enable packet retrieval.

cumulative-summary: Running totals across all packets in the series, enabling Verifiers to assess the full authorship effort without accessing all prior packets.

28.3. Chain Integrity Across Packets

When a new packet continues from a previous packet, the VDF chain MUST maintain cryptographic continuity:

```
Packet N (final checkpoint):
  tree-root[last] = H(checkpoint-data)
  VDF_output{last} = computed VDF result

Packet N+1 (first checkpoint):
  prev-packet-chain-hash = tree-root[last] from Packet N
  VDF_input{0} = H(
      VDF_output{last} from Packet N ||
      content-hash{0} ||
      jitter-commitment{0} ||
      series-id ||
      packet-sequence
  )
```

This construction ensures:

1. The new packet cannot be created without knowledge of the previous packet's final VDF output.

2. Backdating the new packet requires recomputing all VDF proofs in both the current and all subsequent packets.

3. The series-id and packet-sequence are bound into the VDF chain, preventing packets from being reordered or reassigned to different series.

28.4. Verification of Continuation Chains

28.4.1. Single Packet Verification

Each packet in a continuation series MUST be independently verifiable. A Verifier with access only to packet N can:

* Verify all segment chain integrity within the packet.

* Verify all VDF proofs within the packet.

* Verify jitter bindings within the packet.

* Report the cumulative-summary as claimed (not proven without prior packets).

The Attestation Result SHOULD note that the packet is part of a series and whether prior packets were verified.

28.4.2. Full Series Verification

When all packets in a series are available, a Verifier MUST:

1. Verify each packet independently.
2. Verify that series-id is consistent across all packets.
3. Verify that packet-sequence values are consecutive starting from 0.
4. For each packet N > 0, verify that prev-packet-chain-hash matches the final tree-root of packet N-1.
5. For each packet N > 0, verify that the first checkpoint's VDF_input incorporates the previous packet's final VDF_output.
6. Verify that cumulative-summary values are consistent with the sum of individual packet statistics.

28.5. Series Binding Signature

The optional series-binding-signature provides cryptographic proof that all packets in a series were produced by the same author:

```
series-binding-signature = COSE_Sign1(
    payload = CBOR_encode({
        1: series-id,
        2: packet-sequence,
        3: packet-id,
        4: prev-packet-chain-hash,  / if present /
        5: cumulative-summary,
    }),
    key = author-signing-key
)
```

When present, Verifiers can confirm that the signing key is consistent across all packets in the series, providing additional assurance of authorship continuity.

28.6. Practical Considerations

28.6.1. When to Export a Continuation Packet

Implementations SHOULD support configurable triggers for continuation packet export:

* *Checkpoint count threshold:* Export after N checkpoints (e.g., 1000).
* *Time interval:* Export weekly or monthly.
* *Document size threshold:* Export when document exceeds N characters.
* *Manual trigger:* User-initiated export.
* *Milestone events:* Export at chapter completion or version milestones.

28.6.2. Handling Gaps in Series

If a packet in a series is lost or unavailable:

* Subsequent packets remain independently verifiable.
* The cumulative-summary provides claimed totals but cannot be proven without all packets.
* Verifiers MUST note the gap in Attestation Results.
* Chain continuity verification fails at the gap but resumes for subsequent contiguous packets.

28.7. Continuation Token Example

Third monthly export of a dissertation in progress:

```
continuation-section = {
  1: h'dissertation-series-uuid...',  / series-id /
  2: 2,                            / packet-sequence (3rd) /
  3: {                                 / prev-packet-chain-hash /
    1: 1,
    2: h'feb-packet-final-hash...'
  },
  4: h'feb-packet-uuid...',            / prev-packet-id /
  5: {                                 / cumulative-summary /
    1: 847,                            / total-checkpoints-so-far /
    2: 45230,                          / total-chars-so-far /
    3: 12600.0,                        / total-vdf-time: ~3.5 hours /
    4: 156.7,                          / total-entropy-bits /
    5: 3,                              / packets-in-series /
    6: 1(1704067200),               / series-started-at /
    7: 7776000.0                       / total-elapsed: 90 days /
  },
  6: h'D28441A0...'                     / series-binding-signature /
}
```

29. Quantified Trust Policies

This section defines a framework for expressing and computing trust scores in Attestation Results. Trust policies enable Relying Parties to customize how Evidence is evaluated and to understand the basis for confidence scores.

29.1. Trust Policy Motivation

The base attestation-result structure provides a confidence-score (0.0-1.0) and a verdict enumeration, but does not explain how these values were computed. Different Relying Parties have different trust requirements:

* An academic journal may weight presence challenges heavily.

The trust policy framework addresses these limitations by making confidence computation transparent and configurable.

29.2. Trust Policy Structure

The appraisal-policy extension is added to verifier-metadata, identified by integer key 5.

```
; Extended verifier-metadata with trust policy
verifier-metadata = {
    ? 1 => tstr,                    ; verifier-version
    ? 2 => tstr,                    ; verifier-uri
    ? 3 => [+ bstr],                ; verifier-cert-chain
    ? 4 => tstr,                    ; policy-id
    ? 5 => appraisal-policy,        ; policy details
}

; Complete appraisal policy specification
appraisal-policy = {
    1 => tstr,                      ; policy-uri
    2 => tstr,                      ; policy-version
    3 => trust-computation,         ; computation-model
    4 => [+ trust-factor],          ; factors
    ? 5 => [+ trust-threshold],     ; thresholds
    ? 6 => policy-metadata,         ; metadata
}

; How the final score is computed
trust-computation = &(
    weighted-average: 1,            ; Sum of (factor * weight)
    minimum-of-factors: 2,          ; Min across all factors
    geometric-mean: 3,              ; Nth root of product
    custom-formula: 4,              ; Described in policy-uri
)

; Individual factor in trust computation
trust-factor = {
    1 => tstr,                      ; factor-name
    2 => factor-type,               ; type
    3 => ratio-millibits,           ; weight (0-1000 = 0.0-1.0)
    4 => int,                       ; observed-value (units vary by type)
    5 => ratio-millibits,           ; normalized-score (0-1000 = 0.0-1.0)
    6 => ratio-millibits,           ; contribution (weight * score)
    ? 7 => factor-evidence,         ; supporting-evidence
}

factor-type = &(
    ; Chain-verifiable factors
    vdf-duration: 1,
    checkpoint-count: 2,
    jitter-entropy: 3,
    chain-integrity: 4,
    revision-depth: 5,

    ; Presence factors
    presence-rate: 10,
```

```
        presence-response-time: 11,

        ; Hardware factors
        hardware-attestation: 20,
        calibration-attestation: 21,

        ; Behavioral factors
        edit-entropy: 30,
        monotonic-ratio: 31,
        typing-rate-consistency: 32,

        ; External factors
        anchor-confirmation: 40,
        anchor-count: 41,

        ; Collaboration factors
        collaborator-attestations: 50,
        contribution-consistency: 51,
    )

    ; Evidence supporting a factor score
    factor-evidence = {
        ? 1 => int,                     ; raw-value (units vary by factor)
        ? 2 => int,                     ; threshold-value (same units as raw)
        ? 3 => tstr,                    ; computation-notes
        ? 4 => [uint, uint],            ; checkpoint-range
    }

    ; Threshold requirements for pass/fail determination
    trust-threshold = {
        1 => tstr,                      ; threshold-name
        2 => threshold-type,            ; type
        3 => ratio-millibits,           ; required-value (0-1000 for scores)
        4 => bool,                      ; met
        ? 5 => tstr,                    ; failure-reason
    }

    threshold-type = &(
        minimum-score: 1,               ; Score must be >= value
        minimum-factor: 2,              ; factor >= value
        required-factor: 3,             ; factor present
        maximum-caveats: 4,             ; caveats <= value
    )

    policy-metadata = {
        ? 1 => tstr,                    ; policy-name
        ? 2 => tstr,                    ; policy-description
        ? 3 => tstr,                    ; policy-authority
```

```
        ? 4 => pop-timestamp,           ; policy-effective-date
        ? 5 => [+ tstr],                ; applicable-domains
    }
```

29.3. Trust Computation Models

29.3.1. Weighted Average Model

The weighted average model represents the most common computation approach within the RATS appraisal policy framework, where each trust factor derived from VDF proofs, HMAC entropy commitments, and SHA-256 chain integrity contributes proportionally to its assigned weight in the CBOR encoded policy structure defined by the CDDL schema:

```
confidence-score = sum(factor[i].weight * factor[i].normalized-score)
                   / sum(factor[i].weight)
```

Constraints:

- sum(weights) SHOULD equal 1.0 for clarity
- All normalized-scores are in [0.0, 1.0]
- Resulting confidence-score is in [0.0, 1.0]

Example:

```
vdf-duration:      weight=0.30, score=0.95, contribution=0.285
jitter-entropy:    weight=0.25, score=0.80, contribution=0.200
presence-rate:     weight=0.20, score=1.00, contribution=0.200
chain-integrity:   weight=0.15, score=1.00, contribution=0.150
hardware-attest:   weight=0.10, score=0.00, contribution=0.000

confidence-score = 0.285 + 0.200 + 0.200 + 0.150 + 0.000 = 0.835
```

29.3.2. Minimum-of-Factors Model

The minimum-of-factors model represents a conservative computation approach within the RATS appraisal framework where the overall confidence score is limited by the weakest factor, computed as: confidence-score = min(factor[i].normalized-score for all i). This model ensures that deficiencies in any single trust dimension (whether VDF duration, Jitter Seal entropy via HMAC, presence verification, SHA-256 chain integrity, or TPM 2.0 [TPM2.0] hardware attestation) will dominate the final assessment. For example, given vdf-duration score=0.95, jitter-entropy score=0.80, presence-rate score=1.00, chain-integrity score=1.00, and hardware-attest score=0.00 (the limiting factor), the resulting confidence-score equals 0.00 because the absence of hardware attestation bounds the overall trust regardless of strong VDF and HMAC evidence.

This CBOR encoded policy model is appropriate for high-security RATS deployments where any weakness in the evidence chain should disqualify the Evidence packet entirely, such as forensic investigations, legal proceedings requiring COSE signed attestations, or high-stakes academic integrity verification where the cost of accepting forged evidence exceeds the cost of false rejection.

29.3.3. Geometric Mean Model

The geometric mean model provides a balanced computation approach within the RATS appraisal framework that penalizes outliers more heavily than weighted average but less severely than the minimum-of-factors model, computed as: confidence-score = (product(factor[i].normalized-score))^(1/n) where n is the number of trust factors derived from VDF proofs, HMAC entropy, SHA-256 chain integrity, and other evidence dimensions. For example with 5 factors encoded in the CBOR policy structure: given scores = [0.95, 0.80, 1.00, 1.00, 0.60] representing vdf-duration, jitter-entropy, presence-rate, chain-integrity, and hardware-attestation respectively, the product = 0.95 * 0.80 * 1.00 * 1.00 * 0.60 = 0.456, yielding confidence-score = 0.456^(1/5) = 0.838 which maintains reasonable overall confidence despite one weak factor while still penalizing the deficiency more than a simple weighted average would according to the CDDL schema.

29.4. Factor Normalization

Raw factor values derived from VDF proofs, HMAC entropy estimates, SHA-256 chain verification, and other evidence dimensions must be normalized to the [0.0, 1.0] range for consistent computation within the RATS appraisal framework, with normalization functions depending on the factor type as encoded in the CBOR policy structure according to the CDDL schema:

29.4.1. Threshold Normalization

```
For factors with a minimum threshold:
  if raw_value >= threshold:
      normalized = 1.0
  else:
      normalized = raw_value / threshold

Example: vdf-duration with 3600s threshold
  raw_value = 2700s
  normalized = 2700 / 3600 = 0.75
```

29.4.2. Range Normalization

```
For factors with min/max range:
  normalized = (raw_value - min) / (max - min)
  normalized = clamp(normalized, 0.0, 1.0)

Example: typing-rate with acceptable range 20-200 WPM
  raw_value = 75 WPM
  normalized = (75 - 20) / (200 - 20) = 0.306
```

29.4.3. Binary Normalization

```
For pass/fail factors:
  normalized = 1.0 if present/valid else 0.0

Example: hardware-attestation
  TPM attestation present and valid: normalized = 1.0
  No hardware attestation: normalized = 0.0
```

29.5. Predefined Policy Profiles

This RATS profile specification defines several policy profiles for common use cases, with each profile encoded in CBOR according to the CDDL schema and specifying how to weight VDF duration, HMAC Jitter Seal entropy, SHA-256 chain integrity, and TPM 2.0 hardware attestation factors. Implementations MAY support these predefined profiles by URI reference:

Profile URI	Description	Key Characteristics
urn:ietf:params:pop:policy:basic	Basic verification	Chain integrity only
urn:ietf:params:pop:policy:academic	Academic submission	Weighted average, presence required
urn:ietf:params:pop:policy:legal	Legal proceedings	Minimum model, hardware required
urn:ietf:params:pop:policy:publishing	Publishing workflow	Weighted average, VDF emphasized

Table 25: Predefined Policy Profiles

29.6. Trust Policy Example

The following example demonstrates an academic policy within the RATS architecture applied to a Standard tier Evidence packet, encoded in CBOR diagnostic notation according to the CDDL schema, with trust factors derived from VDF duration, HMAC Jitter Seal entropy, presence verification, SHA-256 chain integrity, and edit entropy normalized to the [0.0, 1.0] range:

```
 verifier-metadata = {
   1: "witnessd-verifier-2.0",
   2: "https://verify.example.com",
   4: "academic-v1",
   5: {  / appraisal-policy /
     1: "urn:ietf:params:pop:policy:academic",
     2: "1.0.0",
     3: 1,  / computation: weighted-average /
     4: [   / factors (using millibits: 1000 = 1.0) /
       {
         1: "vdf-duration",
         2: 1,
         3: 250,                      / weight: 250/1000 = 0.25 /
         4: 5400,                     / observed: 90 minutes (seconds) /
         5: 1000,                     / normalized: 1000/1000 = 1.0 /
         6: 250,                      / contribution: 250 * 1000 / 1000 /
         7: {1: 5400, 2: 3600}        / raw, threshold (seconds) /
       },
       {
         1: "jitter-entropy",
         2: 3,
         3: 200,                      / weight: 0.20 /
         4: 457,                      / observed: 45.7 bits (decibits) /
         5: 1000,                     / normalized: 1.0 /
         6: 200                       / contribution: 0.20 /
       },
       {
         1: "presence-rate",
         2: 10,
         3: 250,                      / weight: 0.25 /
         4: 917,                      / observed: 11/12 = 0.917 (millibits) /
         5: 917,                      / normalized: direct ratio /
         6: 229                       / contribution: 250 * 917 / 1000 /
       },
       {
         1: "chain-integrity",
         2: 4,
         3: 200,                      / weight: 0.20 /
         4: 1000,                     / binary valid = 1.0 (millibits) /
```

```
            5: 1000,                    / normalized: 1.0 /
            6: 200                      / contribution: 0.20 /
          },
          {
            1: "edit-entropy",
            2: 30,
            3: 100,                     / weight: 0.10 /
            4: 35,                      / observed: 3.5 bits (decibits) /
            5: 863,                     / normalized: 0.863 (millibits) /
            6: 86                       / contribution: 100 * 863 / 1000 /
          }
        ],
        5: [   / thresholds (millibits) /
          {
            1: "minimum-overall",
            2: 1,
            3: 700,                     / required: 700/1000 = 0.70 /
            4: true
          },
          {
            1: "presence-required",
            2: 3,
            3: 0,                       / any presence suffices /
            4: true
          }
        ],
        6: {   / metadata /
          1: "Academic Submission Policy",
          3: "WritersLogic Academic Integrity",
          5: ["academic", "education", "research"]
        }
      }
    }

    / confidence: (250 + 200 + 229 + 200 + 86) / 1000 = 965/1000 = 0.965 /
```

30. Compact Evidence References

This section defines a compact representation of Evidence within the RATS architecture that can be embedded in document metadata or other space-constrained data structures where full CBOR encoded Evidence packets would exceed available capacity. Compact Evidence References provide a cryptographic link via SHA-256 hashes and COSE signatures to full Evidence packets containing VDF proofs and HMAC Jitter Seals, without requiring the full packet with its complete segment chain to be transmitted or stored in the constrained embedding context.

30.1. Compact Reference Motivation

Full Evidence packets encoded in CBOR according to the CDDL schema can be large (kilobytes to megabytes depending on segment count and VDF proof size), making them unsuitable for direct inclusion in space-constrained document headers or metadata fields.

A Compact Evidence Reference within the RATS architecture provides "proof at a glance" that links to the full Evidence packet containing complete VDF chains and HMAC Jitter Seals for verification. The reference is cryptographically bound to the Evidence through SHA-256 hashes of the chain and document, with a COSE signature preventing tampering with the summary claims without detection by Relying Parties.

30.2. Compact Reference Structure

The Compact Evidence Reference within the RATS architecture uses a dedicated CBOR semantic tag (1347571281 = 0x50505021 = "PPP!") to distinguish it from full Evidence packets containing complete VDF chains and HMAC Jitter Seals, enabling parsers to immediately identify the compact format and locate the referenced full packet via the evidence-uri field for complete verification with SHA-256 chain validation and COSE signature checking.

```
; Compact Evidence Reference
; Tag 1347571281 = 0x50505021 = "PPP!"
tagged-compact-ref = #6.1347571281(compact-evidence-ref)

compact-evidence-ref = {
    1 => uuid,                      ; packet-id
    2 => hash-value,                ; chain-hash
    3 => hash-value,                ; document-hash
    4 => compact-summary,           ; summary
    5 => tstr,                      ; evidence-uri
    6 => cose-signature,            ; compact-signature
    ? 7 => compact-metadata,        ; metadata
}

compact-summary = {
    1 => uint,                      ; checkpoint-count
    2 => uint,                      ; total-chars
    3 => duration,                  ; total-vdf-time
    4 => uint,                      ; evidence-tier (1-4)
    ? 5 => forensic-assessment,     ; verdict (if available)
    ? 6 => confidence-millibits,    ; confidence (0-1000 = 0.0-1.0)
}

compact-metadata = {
    ? 1 => tstr,                    ; author-name
    ? 2 => pop-timestamp,           ; created
    ? 3 => tstr,                    ; verifier-name
    ? 4 => pop-timestamp,           ; verified-at
}
```

30.3. Compact Reference Signature

The compact-signature within the RATS architecture binds all reference fields using COSE_Sign1 to prevent tampering with the summary claims without detection by Relying Parties. The signature is computed as COSE_Sign1(payload = CBOR_encode({1: packet-id (UUID [RFC9562]), 2: chain-hash (SHA-256), 3: document-hash (SHA-256), 4: compact-summary, 5: evidence-uri}), key = signing-key), cryptographically binding the reference to both the document content and the full Evidence packet containing VDF proofs and HMAC Jitter Seals.

The signing key for the COSE signature may be the author’s signing key (self-attestation where the author vouches for their own evidence), the Verifier’s signing key (third-party attestation after independent verification of the VDF chain and SHA-256 integrity), or the evidence service’s key (hosting attestation where the service vouches for packet availability and immutability). The signature

type SHOULD be indicated by the COSE key identifier (kid) header or inferred from the evidence-uri domain within the RATS trust framework.

30.4. Verification of Compact References

30.4.1. Reference-Only Verification

Without fetching the full Evidence packet containing VDF proofs and HMAC Jitter Seals, a Verifier within the RATS architecture can perform reference-only verification by: (1) verifying the COSE compact-signature is valid using the signer’s public key, (2) identifying the signer (author, third-party verifier, or evidence hosting service) from the COSE key identifier, (3) checking that evidence-uri points to a trusted source for fetching the full CBOR encoded Evidence if needed, and (4) displaying the compact-summary to the user showing segment count, total characters, VDF duration, and evidence tier.

This reference-only verification provides basic assurance within the RATS trust framework that Evidence exists and was attested by a known party whose COSE signature is valid, without requiring full verification of the SHA-256 segment chain, VDF proofs, or HMAC entropy commitments.

30.4.2. Full Verification via URI

For complete verification within the RATS architecture, the Verifier follows a six-step procedure that validates both the compact reference and the full CBOR encoded Evidence packet: (1) fetch the Evidence packet from evidence-uri using HTTPS or other secure transport, (2) verify that packet-id (UUID) matches between the compact reference and fetched packet, (3) verify that chain-hash (SHA-256) matches the final tree-root in the fetched Evidence, (4) verify that document-hash (SHA-256) matches the document-ref content-hash binding the evidence to the attested document, (5) perform full Evidence verification per this specification including VDF proof recomputation, HMAC entropy verification, and COSE signature validation, and (6) verify that compact-summary values (checkpoint-count, total-chars, total-vdf-time, evidence-tier) match the actual Evidence computed values.

Discrepancies between the compact reference and the fetched Evidence MUST cause verification to fail within the RATS trust framework, as such discrepancies indicate either tampering with the compact reference, corruption of the full Evidence packet, or a mismatch between the referenced and fetched packets.

30.5. Encoding Formats

Compact Evidence References within the RATS architecture may be encoded in several formats depending on the embedding context, with the base representation being CBOR according to the CDDL schema, but with transformations available for contexts requiring text encoding, URL-safe encoding, or human-readable representation while preserving the cryptographic binding via SHA-256 hashes and COSE signatures:

30.5.1. CBOR Encoding

The native format is CBOR with the 0x50505021 tag. This is the most compact binary representation, suitable for:

* Binary metadata fields
* Protocol messages
* Database storage

Typical size: 150-250 bytes.

30.5.2. Base64 Encoding

For text-only contexts, the CBOR bytes are base64url-encoded:

```
    pop-ref:2nQAAZD1UPAgowGQA...base64url...
```

The "pop-ref:" prefix enables detection and parsing. Typical size: 200-350 characters.

30.6. Compact Reference Example

```
/ Tagged Compact Evidence Reference (0x50505021 = "PPP!") /
1347571281({
  1: h'550e8400e29b41d4a716446655440000',  / packet-id /
  2: {                                      / chain-hash /
    1: 1,
    2: h'a7ffc6f8bf1ed76651c14756a061d662
          f580ff4de43b49fa82d80a4b80f8434a'
  },
  3: {                                      / document-hash /
    1: 1,
    2: h'e3b0c44298fc1c149afbf4c8996fb924
          27ae41e4649b934ca495991b7852b855'
  },
  4: {                                      / compact-summary /
    1: 47,                                  / checkpoints /
    2: 12500,                               / chars /
    3: 5400.0,                              / VDF time: 90 min /
    4: 2,                                   / tier: Standard /
    5: 2,                                   / verdict: manual-composition-likely /
    6: 0.87                                 / confidence /
  },
  5: "https://evidence.example.com/p/"\
     "550e8400e29b41d4a716446655440000.pop",
  6: h'D28441A0A201260442...',              / compact-signature /
  7: {                                      / metadata /
    1: "Jane Author",
    2: 1(1706745600),                       / created /
    3: "WritersLogic Verification Service",
    4: 1(1706832000)                        / verified /
  }
})
```

Encoded size: approximately 220 bytes (CBOR), 295 characters (base64url).

31. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of publication, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit."

31.1. witnessd-core (Reference Implementation)

Organization: Writerslogic Inc

Implementation Name: witnessd-core

Implementation URL: https://github.com/writerslogic/witnessd

Description: Rust library implementing the complete PPPP specification including checkpoint generation, VDF computation (Wesolowski construction), HMAC Jitter Seal entropy binding, hash chain construction, COSE signatures, and CBOR encoding. Supports all three evidence tiers (software-only, attested, hardware-bound) with TPM 2.0 and Secure Enclave integration.

Maturity Level: Production

Coverage: Full specification coverage including: checkpoint chain construction, VDF temporal proofs, jitter entropy binding, absence claims, forgery cost bounds, continuation tokens, salt modes, and all profile levels (core, enhanced, maximum).

Version Compatibility: Schema version 1.6.0

Licensing: Apache-2.0

Contact: David Condrey (david@writerslogic.com)

31.2. witnessd-cli

Organization: Writerslogic Inc

Implementation Name: witnessd-cli

Implementation URL: https://github.com/writerslogic/witnessd

Description: Command-line interface built on witnessd-core providing evidence generation, verification, and inspection capabilities. Supports batch processing, JSON output, and integration with build systems and CI/CD pipelines.

Maturity Level: Production

Coverage: Complete Evidence Packet (.pop) generation and Attestation Result (.war) verification.

Licensing: Apache-2.0

Contact: David Condrey (david@writerslogic.com)

31.3. Witnessd for macOS

Organization: Writerslogic Inc

Implementation Name: Witnessd for macOS

Description: Native macOS desktop application providing real-time evidence generation during document editing. Integrates with Secure Enclave for hardware-bound key storage and attestation (Tier T3).

Maturity Level: Production

Coverage: Full evidence generation with Secure Enclave integration, automatic checkpoint creation, and evidence export.

Platform: macOS 12.0+ (Apple Silicon and Intel)

Contact: David Condrey (david@writerslogic.com)

31.4. Witnessd for Windows

Organization: Writerslogic Inc

Implementation Name: Witnessd for Windows

Description: Native Windows desktop application providing real-time evidence generation during document editing. Integrates with TPM 2.0 for hardware-bound key storage and attestation (Tier T3).

Maturity Level: Production

Coverage: Full evidence generation with TPM 2.0 integration, automatic checkpoint creation, and evidence export.

Platform: Windows 10/11

Contact: David Condrey (david@writerslogic.com)

31.5. WritersLogic Online Verifier

Organization: Writerslogic Inc

Implementation Name: WritersLogic Verification Service

Implementation URL: https://writerslogic.com/verify

Description: Web-based independent Verifier implementation for Attestation Result generation. Accepts Evidence Packets (.pop), performs complete verification including chain integrity, VDF validation, entropy threshold checks, and optional external anchor verification, then produces Attestation Results (.war).

Maturity Level: Production

Coverage: Full Verifier role implementation per RATS architecture including all verification checks defined in this specification.

Contact: David Condrey (david@writerslogic.com)

31.6. Interoperability Testing

The implementations listed above have been tested for interoperability. Evidence Packets generated by witnessd-core, witnessd-cli, the macOS application, and the Windows application are successfully verified by the WritersLogic Online Verifier, demonstrating cross-implementation compatibility.

Test vectors from [I-D.condrey-rats-pop-examples] have been validated against all implementations.

32. Security Considerations

This section consolidates security analysis for the witnessd Proof of Process specification within the RATS [RFC9334] architecture, referencing and extending the per-section security considerations defined in Section 23 for HMAC Jitter Seal entropy, Section 24.7 for VDF chain temporal guarantees, Section 25.6.5 for gap detection in segment-based Merkle trees, Section 26.8 for forgery cost bound quantification, and Section 9.17 for overall CBOR evidence integrity with COSE signatures.

The specification adopts a quantified security approach consistent with the RATS philosophy: rather than claiming evidence is "secure" or "insecure" in absolute terms, security is expressed as cost asymmetries in VDF recomputation, entropy prediction barriers in HMAC-SHA256 Jitter Seals, and tamper-evidence properties through

SHA-256 hash chains with COSE signatures. This framing reflects the fundamental reality that sufficiently resourced adversaries can eventually forge any evidence; the goal is to make forgery economically irrational for most scenarios by ensuring that the VDF time cost, HMAC entropy prediction cost, and computational resource cost exceed the potential gain from successful forgery.

32.1. Research Limitations and Assumptions

The "Biology" invariant relies on psycholinguistic correlations (e.g., Cognitive Load Delays) that require further large-scale empirical validation across diverse modern input methods (e.g., mobile autocomplete). The "Pink Noise" metric assumes a simplified motor control model. Repeated evidence generation by the same author may enable cross-session timing analysis, which is mitigated through periodic key rotation and timing value clipping.

32.2. Threat Model

The witnessd threat model within the RATS architecture defines three categories relevant to VDF chain security, HMAC entropy commitment integrity, and SHA-256 segment chain tamper-evidence: adversary goals describing what attacks the protocol defends against, assumed adversary capabilities bounding the resources available to attackers, and explicitly out-of-scope adversaries whose capabilities exceed the design assumptions of this CBOR evidence format with COSE signatures.

32.2.1. Adversary Goals

The protocol defends against adversaries pursuing five primary goals: (1) Backdating: creating evidence that falsely claims earlier creation time; (2) Fabrication: generating evidence for documents never genuinely authored; (3) Transplanting: associating legitimate evidence with different content; (4) Omission: selectively removing checkpoints from an evidence chain; (5) Impersonation: attributing evidence to a different device or author. Each goal is addressed through specific cryptographic and structural properties: VDF sequential computation prevents backdating, jitter entropy prevents fabrication, content binding prevents transplanting, Merkle trees detect omission, and hardware attestation prevents impersonation.

32.2.2. Assumed Adversary Capabilities

The RATS profile specification assumes adversaries have five categories of capabilities that bound the security guarantees of VDF chains, HMAC entropy commitments, and SHA-256 checkpoint integrity. Software Control: the adversary has full control over software running on their device, including the ability to modify or replace the Attesting Environment that generates CBOR evidence, and they can intercept, modify, or fabricate any software-generated data that is not protected by TPM 2.0 [TPM2.0] hardware attestation or similar tamper-resistant hardware. Commodity Hardware Access: the adversary can acquire commodity computing hardware at market prices for computing SHA-256 iterations, and they may have access to cloud computing resources enabling them to rent substantial computational capacity for VDF recomputation attempts. Bounded Compute Resources: the adversary's computational resources are bounded by economic constraints quantified in the forgery cost bounds, meaning they cannot instantaneously compute arbitrarily large numbers of VDF iterations, and the wall-clock time required for sequential SHA-256 computation cannot be circumvented with additional parallel resources due to the inherent data dependency between iterations. Algorithm Knowledge: the adversary has complete knowledge of all algorithms including VDF constructions, CDDL schemas, COSE signatures, and CBOR encoding, since security does not depend on obscurity and the specification is public. Statistical Sophistication: the adversary can perform statistical analysis on Jitter Seal timing histograms and may attempt to generate synthetic behavioral data using HMAC that passes Min-Entropy (H_min) tests, though the commitment-before-observation model prevents adaptive synthesis.

32.2.3. Out-of-Scope Adversaries

The RATS profile specification explicitly does NOT defend against five categories of adversaries whose capabilities exceed the design assumptions of VDF chains, HMAC entropy commitments, and SHA-256 checkpoint integrity. Nation-State Adversaries with HSM Compromise: adversaries capable of extracting keys from hardware security modules (TPM 2.0, Secure Enclave) through sophisticated physical attacks, side-channel analysis, or manufacturer compromise, since hardware attestation via COSE signatures assumes HSM integrity for calibration and identity binding. Cryptographic Breakthrough: adversaries with access to novel cryptanalytic techniques that break SHA-256 collision resistance, ECDSA signature security underlying COSE, or other standard cryptographic primitives used throughout the CDDL schema, since the specification relies on established cryptographic assumptions. Quantum Adversaries: adversaries with access to fault-tolerant quantum computers capable of executing Shor's algorithm (breaking RSA/ECDSA used in COSE signatures) or providing significant

Grover speedups against SHA-256 preimage resistance, with post-quantum considerations noted in Section 24.6.2 but full quantum resistance not claimed. Time Travel: adversaries capable of creating CBOR evidence at one point in time and presenting it as if created earlier where external anchors via RFC 3161 or blockchain timestamps are not available or have been compromised, since external timestamp authorities are trusted for absolute time claims beyond VDF relative ordering. Coerced Authors: adversaries who coerce legitimate authors into producing evidence with valid VDF proofs and HMAC Jitter Seals under duress, since the specification documents process rather than intent or consent.

The exclusion of these adversaries is not a weakness but a recognition of practical threat modeling within the RATS framework, since evidence systems appropriate for defending against nation-state actors with HSM compromise or quantum computational capabilities would impose costs and constraints (such as post-quantum COSE algorithms or hardware-isolated attestation environments) unsuitable for general authoring scenarios where the goal is making forgery economically irrational rather than theoretically impossible.

32.3. Cryptographic Security

The specification relies on established cryptographic primitives with well-understood security properties. This section documents the security assumptions and requirements for each cryptographic component.

32.3.1. Hash Function Security

Hash functions are used throughout the specification for content binding, chain construction, entropy commitment, and VDF computation.

Required Properties:

* Collision Resistance:

  It must be computationally infeasible to find two distinct inputs that produce the same hash output. This property ensures that different document states produce different content-hash values.

* Preimage Resistance:

  Given a hash output, it must be computationally infeasible to find any input that produces that output. This property prevents adversaries from constructing documents that match a predetermined hash.

* Second Preimage Resistance:

  Given an input and its hash, it must be computationally infeasible to find a different input with the same hash. This property prevents document substitution attacks.

Algorithm Requirements: SHA-256 is RECOMMENDED and MUST be supported by all implementations. SHA-3-256 SHOULD be supported for algorithm agility. Hash functions with known weaknesses (MD5, SHA-1) MUST NOT be used.

Security Margin: SHA-256 provides 128-bit security against collision attacks and 256-bit security against preimage attacks under classical assumptions. Grover's algorithm reduces these to 85-bit and 128-bit respectively under quantum assumptions. This margin is considered adequate for the specification's threat model.

32.3.2. Signature Security

Digital signatures are used for segment chain authentication, hardware attestation, calibration binding, and Attestation Result integrity.

COSE Algorithm Requirements: Implementations MUST support COSE algorithm identifiers from the COSE registry [IANA.cose]:

* ES256 (ECDSA with P-256 and SHA-256): MUST support

* ES384 (ECDSA with P-384 and SHA-384): SHOULD support

* EdDSA (Ed25519): SHOULD support

* ML-DSA (Dilithium): REQUIRED for Maximum Tier evidence to ensure post-quantum signature security

RSA-based algorithms (PS256, RS256) MAY be supported for compatibility with legacy systems but are not recommended for new implementations due to larger signature sizes and post-quantum vulnerability.

Key Size Requirements: Minimum key sizes for 128-bit security:

* ECDSA: P-256 curve or larger

* EdDSA: Ed25519 or Ed448

* RSA: 3072 bits or larger

Signature Binding: Signatures MUST bind to the complete payload being signed. Partial payload signatures (signing a subset of fields) create opportunities for field substitution attacks. The chain-mac field provides additional binding beyond the checkpoint signature.

32.3.3. VDF Security

Verifiable Delay Functions provide the temporal security foundation of the specification. VDF security rests on the sequential computation requirement.

Sequential Computation: The VDF output cannot be computed significantly faster than the specified number of sequential operations. For iterated hash VDFs, this reduces to the assumption that no algorithm computes H^n(x) faster than n sequential hash evaluations. No such algorithm is known for cryptographic hash functions.

Parallelization Resistance: Additional computational resources (more processors, GPUs, ASICs) cannot reduce the wall-clock time required for VDF computation. The iterated hash construction is inherently sequential: each iteration depends on the previous output.

See Section 24.7.2 for detailed analysis.

Verification Soundness: For iterated hash VDFs, verification is by recomputation. The Verifier executes the same computation and compares results. This provides perfect soundness: a claimed output that differs from the actual computation will always be detected.

For succinct VDFs (Pietrzak, Wesolowski), verification relies on the cryptographic hardness of the underlying problem (RSA group or class group). Soundness is computational rather than perfect.

32.3.4. VDF Entanglement Attack Vectors

The VDF Entanglement mechanism binds each checkpoint to the previous VDF output, current content-hash, jitter-commitment, and sequence number. This section analyzes three specific attack vectors against this construction and documents their mitigations and cost bounds.

32.3.4.1. Grinding Attacks

A grinding attack attempts to influence the VDF output by iteratively selecting different jitter-commitment values until the resulting VDF input produces a favorable output.

Attack Mechanism: The attacker generates candidate raw-interval sequences, computes SHA-256 for each to produce candidate jitter-commitments, computes the full VDF for each input, and evaluates whether the output satisfies some "favorable" criterion.

Cost Bound: With T=2^25 sequential SHA-256 iterations (~10 seconds minimum wall-clock time per VDF evaluation), grinding N candidates requires N× 10 seconds of sequential work. Parallelization reduces wall-clock time but increases hardware cost proportionally. For N=1000 candidates, the attacker requires either ~2.8 hours sequential or 1000× hardware investment for 10-second parallel grinding.

Mitigation: The VDF’s inherent sequential computation requirement ensures grinding cost scales linearly with attempts. Verifiers SHOULD treat evidence with implausibly favorable VDF outputs (e.g., outputs matching specific bit patterns) with increased skepticism. The economic irrationality of grinding depends on the value of achieving a "favorable" output being less than the computational cost; for most authorship scenarios, no particular VDF output provides exploitable advantage.

Residual Risk: If the "favorable output" criterion is loose (e.g., any output in a large set), grinding becomes more feasible. Implementations SHOULD NOT rely on VDF outputs having any particular statistical properties beyond unpredictability.

32.3.4.2. Pre-computation Attacks

A pre-computation attack attempts to compute VDF chains offline when the content-hash is predictable (e.g., template documents, boilerplate), then present them as evidence of real-time work.

Attack Mechanism: For pre-computation to succeed, the attacker must know VDF_output{N-1} (requiring a valid prior chain), predict content-hash{N} (achievable for templates), forge jitter-commitment{N} (requiring synthetic behavioral data), and correctly predict sequence{N} (deterministic from chain state).

Primary Defense: The jitter-commitment acts as a cryptographic

nonce. Because entropy-commitment = SHA-256(raw-intervals) is computed BEFORE histogram aggregation, the attacker must commit to specific interval sequences, not merely plausible histogram shapes. The cardinality of valid interval sequences vastly exceeds histogram space, preventing pre-computation of the commitment.

Binding MAC Defense: The binding-mac field includes prev-tree-root, preventing transplantation of jitter data from unrelated checkpoint chains. An attacker cannot pre-compute jitter for chain A and graft it onto chain B.

Residual Risk: An attacker who records legitimate typing sessions on their own device can replay those intervals with pre-computed content. The jitter-commitment is "real" but temporally decoupled from the content work. This attack requires the attacker to have produced genuine behavioral data at some point; it enables temporal displacement but not fabrication ex nihilo. Implementations requiring stronger guarantees SHOULD require external timestamp anchors (RFC 3161 or blockchain) to bind evidence to absolute time.

32.3.4.3. Statistical Modeling Attacks

A statistical modeling attack trains a machine learning model on legitimate jitter data to generate synthetic patterns that pass entropy validation, enabling fake checkpoints without real behavioral input.

Attack Mechanism: The attacker collects legitimate jitter histograms, trains a generative model (VAE, GAN, or similar) on the distribution, and samples synthetic histograms matching learned statistical properties.

Primary Defense: The commitment-before-observation model is the critical defense. Because entropy-commitment = SHA-256(raw-intervals) is computed from raw intervals, not histogram buckets, the attacker must generate plausible raw interval sequences. The space of valid interval sequences (millisecond-precision timings across hundreds or thousands of events) is orders of magnitude larger than the histogram summary space. Training a generative model on histograms provides no information about which specific interval sequences produced those histograms.

Entropy Validation: Verifiers compute Min-Entropy (H_min) on the

declared histogram. Synthetic histograms that are "too uniform" (high entropy) or "too concentrated" (low entropy) fail validation. Hurst exponent analysis (valid range $H \in [0.55, 0.85]$) further distinguishes genuine behavioral data exhibiting long-range temporal dependence from synthetic generation attempts.

Residual Risk: An attacker with access to large corpora of raw interval sequences (not just histograms) could train a generative model on intervals directly. Timing value clipping bounds information leakage; however, an attacker observing many histograms from the same source could infer distributional properties. Implementations requiring defense against well-resourced statistical attackers SHOULD require hardware attestation (T3/T4 tiers) binding jitter capture to trusted execution environments where the raw intervals cannot be intercepted pre-commitment.

32.3.4.4. Combined Attack Cost Analysis

An adversary attempting to forge evidence must overcome multiple independent barriers simultaneously:

* VDF recomputation: ~10 seconds wall-clock minimum per checkpoint, non-parallelizable
* Jitter synthesis: Must generate raw intervals (not histograms) that pass entropy validation and match behavioral plausibility tests
* Chain binding: Must possess valid previous VDF output and tree root, preventing ex nihilo fabrication
* Temporal binding: External anchors (when present) constrain absolute timing claims

The composition of these barriers means that practical forgery requires either (a) legitimate prior chain access plus VDF computation time plus synthetic but plausible jitter, or (b) compromise of the Attesting Environment itself. Cost-asymmetry is maintained: generating genuine evidence requires only normal authoring activity, while forgery requires computational investment, behavioral data acquisition, and chain access.

32.3.5. Key Management

Proper key management is essential for maintaining evidence integrity.

Hardware-Bound Keys: When available, signing keys SHOULD be bound to hardware security modules (TPM, Secure Enclave). Hardware binding provides:

* Key non-exportability: Private keys cannot be extracted from the device

* Device binding: Evidence can be tied to a specific physical device

* Tamper resistance: Key compromise requires physical attack

Session Keys: The checkpoint-chain-key used for chain-mac computation SHOULD be derived uniquely for each session. Key derivation SHOULD use HKDF (RFC 5869) with domain separation:

```
# Root Credential (from Enrollment)
# RC = HKDF-SHA256(PUF_Seed, "witnessd-root-v1")
#
# Session Key (derived from Root Credential)
# SK = HKDF-SHA256(RC, session-nonce || "witnessd-session-v1")
#
# Checkpoint Chain Key (derived from Session Key)
# CCK = HKDF-SHA256(SK, "witnessd-chain-v1")

chain-key = HKDF-SHA256(
    salt = session-entropy,
    ikm = device-master-key,
    info = "witnessd-chain-v1" || session-id
)
```

Key Rotation: Device keys SHOULD be rotated periodically (RECOMMENDED: annually) or upon suspected compromise. Evidence packets created with revoked keys SHOULD be flagged during verification.

32.4. Attesting Environment Trust

The Attesting Environment (AE) is the witnessd-core software running on the author’s device. Understanding what the AE is trusted for, and what it is NOT trusted for, is essential for correct interpretation of evidence.

32.4.1. What the AE Is Trusted For

The AE is trusted to perform accurate observation and honest reporting of the specific data it captures:

Accurate Timing Measurement: The AE is trusted to accurately measure inter-keystroke intervals and other timing data. This does not require trusting the content of keystrokes, only the timing between events.

Correct Hash Computation: The AE is trusted to correctly compute cryptographic hashes of document content. Verification can detect incorrect hashes, but cannot detect if the AE computed a hash of different content than claimed.

VDF Execution: The AE is trusted to actually execute VDF iterations rather than fabricating outputs. This trust is partially verifiable: VDF outputs can be recomputed, but the claimed timing cannot be independently verified without calibration attestation.

Monitoring Events (for monitoring-dependent claims): For claims in the monitoring-dependent category (types 16-63), the AE is trusted to have actually observed and reported the events (or non-events) it claims. This trust is documented in the ae-trust-basis field.

32.4.2. What the AE Is NOT Trusted For

The specification explicitly does NOT rely on AE trust for the following:

Content Judgment: The AE makes no claims about document quality, originality, accuracy, or appropriateness. Evidence documents process, not content merit.

Intent Inference: The AE makes no claims about why the author performed specific actions, what the author was thinking, or whether the author intended to deceive. Evidence documents observable behavior, not mental states.

Authorship Attribution: The AE makes no claims about who was operating the device. The evidence shows that input events occurred on a device; it does not prove that a specific individual produced those events.

Cognitive Process: Behavioral patterns consistent with human typing do not prove human cognition. An adversary could theoretically program input patterns that mimic human timing while the content originates elsewhere. The Jitter Seal makes this costly, not impossible.

32.4.3. Hardware Attestation Role

Hardware attestation increases AE trust by binding evidence to verified hardware:

[TPM2.0] (Linux, Windows): Provides platform integrity measurement (PCRs), key sealing to platform state, and hardware-bound signing keys. TPM attestation proves that the AE was running on a specific device in a specific configuration.

Secure Enclave (macOS, iOS): Provides hardware-bound key generation and signing operations. Keys generated in the Secure Enclave cannot be exported, binding signatures to the specific device.

Attestation Limitations: Hardware attestation proves the signing key is hardware-bound; it does not prove the AE software is unmodified. Full AE integrity would require secure boot attestation and runtime integrity measurement, which are platform-specific and not universally available.

32.4.4. Compromised AE Scenarios

Understanding the impact of AE compromise is essential for risk assessment:

Modified AE Software: An adversary running modified AE software can fabricate any monitoring-dependent claims (types 16-63). Chain-verifiable claims (types 1-15) remain bound by VDF computational requirements even with modified software.

Fake Calibration: Modified software could report artificially slow calibration rates, making subsequent VDF computations appear to take longer than they actually did. This attack is mitigated by:

* Hardware-signed calibration attestation (when available)
* Plausibility checks based on device class
* External anchor cross-validation

Fabricated Jitter Data: Modified software could generate synthetic timing data that mimics human patterns. The cost of this attack is bounded by:

* Real-time generation requirement (VDF entanglement)
* Statistical consistency across checkpoints

* Entropy threshold requirements

See Section 23.2 for quantified bounds on simulation attacks.

Mitigation Summary: AE compromise cannot reduce the VDF computational requirement or bypass the sequential execution constraint. Compromise enables fabrication of monitoring data but does not eliminate the time cost of forgery. The forgery-cost-section quantifies the minimum resources required even with full software control.

32.5. Verification Security

The verification process must be secure against both malicious Evidence and malicious Verifiers.

32.5.1. Verifier Independence

Evidence verification is designed to be independent of the Attester:

No Shared State: Verification requires no communication with or data from the Attester beyond the Evidence packet itself. A Verifier with only the .pop file can perform complete verification.

Adversarial Verification: A skeptical Verifier can appraise Evidence without trusting any claims made by the Attester. All cryptographic proofs are included and can be recomputed independently.

Multiple Independent Verifiers: Multiple Verifiers appraising the same Evidence should reach consistent results for computationally-bound claims. Monitoring- dependent claims may receive different confidence assessments based on Verifier policies.

32.5.2. Sampling Strategies for Large Evidence Packets

Evidence packets may contain thousands of checkpoints. Full verification of all VDF proofs may be impractical. Verifiers MAY use sampling strategies:

Boundary Verification: Always verify the first and last checkpoints fully. This confirms the chain endpoints.

Random Sampling: Randomly select checkpoints for full VDF verification. If any sampled checkpoint fails, reject the entire Evidence. Probability of detecting a single invalid checkpoint with k samples from n checkpoints: $1 - (1 - 1/n)^k$.

Chain Linkage Verification: Verify prev-hash linkage for ALL checkpoints (computationally cheap). This ensures no checkpoints were removed or reordered.

Anchor-Bounded Verification: If external anchors are present, prioritize verification of checkpoints adjacent to anchors. External timestamps bound the timeline at anchor points.

Sampling Disclosure: Attestation Results SHOULD disclose the sampling strategy used and the number of checkpoints fully verified. Relying Parties can assess whether the sampling provides adequate confidence for their use case.

32.5.3. External Anchor Verification

External anchors (RFC 3161 timestamps, blockchain proofs) provide absolute time binding but introduce additional trust requirements:

Timestamp Authority Trust: Timestamps per RFC 3161 require trust in the Time Stamping Authority (TSA). Verifiers SHOULD use TSAs with published policies and audit records. Multiple TSAs MAY be used for redundancy.

Blockchain Anchor Verification: Blockchain-based anchors require access to blockchain data (directly or via APIs). Verifiers SHOULD verify:

* The transaction containing the anchor is confirmed
* Sufficient confirmations for the security level required
* The anchor commitment matches the expected segment data

Anchor Freshness: Anchors prove that Evidence existed at the anchor time; they do not prove Evidence was created at that time. An adversary could create Evidence, wait, then obtain an anchor. This is mitigated by anchor coverage requirements (multiple anchors throughout the session).

32.6. Protocol Security

This section addresses protocol-level attacks and mitigations, drawing on the per-section security analyses.

32.6.1. Replay Attack Prevention

Replay attacks attempt to reuse valid evidence components in invalid contexts. Multiple mechanisms prevent replay:

Nonce Binding: Session entropy (random 256-bit seed) is incorporated into the genesis checkpoint VDF input. This prevents precomputation of VDF outputs before a session begins.

Chain Binding: Each checkpoint includes prev-hash, binding it to the specific chain history. Checkpoints cannot be transplanted between chains without invalidating the hash linkage.

See Section 23.1 for jitter-specific replay prevention.

Sequence Binding: Checkpoint sequence numbers MUST be strictly monotonic. Duplicate or out-of-order sequence numbers indicate manipulation.

Content Binding: VDF inputs incorporate content-hash, binding temporal proofs to specific document states. Evidence for one document cannot be transferred to another without VDF recomputation.

32.6.2. Transplant Attack Prevention

Transplant attacks attempt to associate legitimate evidence from one context with content from another context:

Content-VDF Binding: The VDF input includes content-hash:

```
VDF_input{N} = H(
    VDF_output{N-1} ||
    content-hash{N} ||
    jitter-commitment{N} ||
    sequence{N}
)
```

Changing the document content requires recomputing all subsequent VDF proofs.

Jitter-VDF Binding: The jitter-commitment is entangled with VDF input. Transplanting jitter data from another session is infeasible because it would require the original VDF output (which depends on different content) or recomputing the entire VDF chain with new jitter (which requires capturing new behavioral entropy in real time).

Chain MAC: The chain-mac field HMAC-binds checkpoints to the session's chain-key:

```
chain-mac = HMAC-SHA256(
    key = chain-key,
    message = tree-root || sequence || session-id
)
```

Without the chain-key, an adversary cannot construct valid chain-mac values for transplanted checkpoints.

32.6.3. Backdating Attack Costs

Backdating creates evidence claiming a process occurred earlier than it actually did. The cost of backdating is quantified by the VDF recomputation requirement:

VDF Recomputation: To backdate evidence by inserting or modifying checkpoints at position P, the adversary must recompute all VDF proofs from position P forward. This requires:

```
backdate_time >= sum(iterations[i]) / adversary_vdf_rate
                 for i = P to N
```

where N is the final checkpoint. Backdating by a significant amount (hours or days) requires proportional wall-clock time.

External Anchor Constraints: If external anchors exist in the chain, backdating is constrained to the interval between anchors. An adversary cannot backdate before an anchor without also forging the external timestamp.

Cost Quantification: The forgery-cost-section provides explicit cost bounds for backdating attacks, including compute costs, time costs, and economic estimates.

32.6.4. Omission Attack Prevention

Omission attacks selectively remove checkpoints to hide unfavorable evidence:

Sequence Verification: Checkpoint sequence numbers MUST be consecutive. Missing sequence numbers indicate omission. Verifiers MUST reject chains with non-consecutive sequences.

Hash Chain Integrity: Removing a checkpoint breaks the hash chain (subsequent checkpoint's prev-hash will not match). Repairing the chain requires recomputing all subsequent segment hashes and VDF proofs.

Completeness Claims: The checkpoint-chain-complete absence claim

(type 6) explicitly asserts that no checkpoints were omitted. This claim is computationally-bound.

32.7. Operational Security

Security of the overall system depends on proper operational practices beyond the protocol specification.

32.7.1. Key Lifecycle Management

Key Generation: Device keys SHOULD be generated within hardware security modules when available. Software-generated keys MUST use cryptographically secure random number generators.

Key Storage: Private keys SHOULD be stored in platform-appropriate secure storage:

* macOS: Secure Enclave or Keychain
* Linux: TPM or system keyring
* Windows: TPM or DPAPI

Keys MUST NOT be stored in plaintext in the filesystem.

Key Rotation: Organizations SHOULD establish key rotation policies. RECOMMENDED rotation interval: annually or upon personnel changes. Evidence packets created with revoked keys SHOULD receive reduced confidence scores.

Key Revocation: Mechanisms for key revocation are outside the scope of this specification but SHOULD be considered for deployment. Certificate revocation lists (CRLs) or OCSP may be appropriate for managed environments.

32.7.2. Evidence Packet Storage and Transmission

Integrity Protection: Evidence packets are self-protecting through cryptographic binding. Additional encryption is not required for integrity but MAY be applied for confidentiality.

Confidentiality Considerations: Evidence packets contain document hashes and behavioral data. While content is not included, statistical information about the authoring process is present. Transmission over untrusted networks SHOULD use TLS 1.3 or equivalent.

Archival Storage: Evidence packets intended for long-term storage

SHOULD be:

* Stored with redundancy (multiple copies, geographic distribution)

* Protected against bit rot (checksums, error-correcting codes)

* Associated with necessary verification materials (public keys, anchor confirmations)

Retention Policies: Organizations SHOULD establish retention policies balancing evidentiary value against privacy considerations. Jitter data has privacy implications; retention beyond the verification period may not be necessary or desirable.

32.7.3. Verifier Policy Considerations

Minimum Requirements: Verifiers SHOULD establish minimum requirements for acceptable Evidence:

* Minimum evidence tier (Basic, Standard, Enhanced, Maximum)

* Minimum VDF duration relative to claimed authoring time

* Minimum entropy threshold

* Required absence claims for specific use cases

Confidence Thresholds: Verifiers SHOULD define confidence thresholds for acceptance:

* Low-stakes: confidence >= 0.3 may be acceptable

* Standard: confidence >= 0.5 typical requirement

* High-stakes: confidence >= 0.7 recommended

* Litigation: confidence >= 0.8 with Maximum tier

Caveat Handling: Verifiers SHOULD define how caveats affect acceptance decisions. Some caveats may be disqualifying for specific use cases (e.g., "no hardware attestation" may be unacceptable for high-stakes verification).

32.8. Limitations and Non-Goals

This section explicitly documents what the specification does NOT protect against and what it does NOT claim to achieve.

32.8.1. Attacks Not Protected Against

Collusion: If the author and a third party collude (e.g., the author provides their device credentials to another person who types while the author is credited), the Evidence will show a legitimate-looking process. The specification documents observable behavior, not identity.

Pre-Prepared Content: An author could slowly type pre-prepared content, creating Evidence of a gradual process for content that already existed. The specification documents that typing occurred, not that thinking occurred during typing.

External Input Devices: Input from devices not monitored by the AE (e.g., hardware keystroke injectors, remote desktop from unmonitored machines) may not be distinguishable from local input. Hardware-level input verification is outside scope.

Social Engineering: Attacks that manipulate Relying Parties into accepting inappropriate Evidence (e.g., convincing a reviewer that weak Evidence is sufficient) are outside scope.

32.8.2. The Honest Author Assumption

The specification fundamentally documents PROCESS, not INTENT:

Evidence Shows What Happened: Evidence shows that input events occurred with specific timing patterns, that VDF computation required certain time, that document states changed in sequence. Evidence does not show why any of this happened.

Process != Cognition: Evidence that an author typed content gradually does not prove the author thought of that content. The author could have been transcribing, copying from memory, or following dictation.

Behavioral Consistency: The correct interpretation of Evidence is "behavioral consistency": the observable process was consistent with the claimed process. This is weaker than "authorship proof" but is verifiable and falsifiable.

32.8.3. Content-Agnostic By Design

The specification is deliberately content-agnostic:

No Semantic Analysis: Evidence contains document hashes, not content. The specification makes no claims about what was written, only how it was written.

No Quality Assessment: Evidence does not indicate whether content is good, original, accurate, or valuable. Strong Evidence can accompany poor content; excellent content can have weak Evidence.

No AI Detection: The specification explicitly does NOT claim to detect whether content was "written by AI" or "written by a human" in terms of content origin. It documents the observable INPUT process, which is distinct from content generation.

Privacy Benefit: Content-agnosticism is a privacy feature. Evidence can be verified without accessing the document content, enabling verification of confidential documents.

32.9. Comparison to Related Work

This section compares the security model of witnessd Proof of Process to related attestation and timestamping systems.

32.9.1. Comparison to Traditional Timestamping

Traditional timestamping (RFC 3161) proves that a document existed at a point in time. Proof of Process provides additional properties:

Property	RFC 3161	Proof of Process
Existence proof	Yes (point in time)	Yes (continuous)
Process documentation	No	Yes
Behavioral evidence	No	Yes (jitter)
Temporal ordering	No (independent timestamps)	Yes (VDF chain)
Third-party trust	Required (TSA)	Optional (anchors)
Local generation	No (requires TSA interaction)	Yes

Table 26

Proof of Process is complementary to timestamping. External anchors (including RFC 3161 timestamps) provide absolute time binding that strengthens VDF-based relative ordering.

32.9.2. Comparison to Code Signing

Code signing attests to the identity of the signer and integrity of the signed artifact. Proof of Process serves different goals:

Property	Code Signing	Proof of Process
Identity binding	Strong (PKI)	Weak (device-bound)
Artifact integrity	Yes	Yes (hash binding)
Creation process	No	Yes
Temporal properties	Timestamp only	Duration, ordering
Use case	Software distribution	Authoring documentation

Table 27

Code signing establishes "who signed this"; Proof of Process establishes "how this was created." The two could be combined for comprehensive provenance documentation.

32.9.3. Relationship to RATS Security Model

Proof of Process implements an application-specific profile of the RATS architecture. Key security model alignments:

Evidence vs. Attestation Results: The separation between .pop (Evidence) and .war (Attestation Result) files follows the RATS distinction. Evidence is produced by the Attester; Attestation Results by the Verifier.

Appraisal Policy: RATS defines Appraisal Policy for Evidence as the Verifier’s rules for evaluating Evidence. The absence-claim thresholds and confidence-level requirements serve this role in Proof of Process.

Background Check vs. Passport Model: Proof of Process supports both RATS models. The "passport model" applies when the author obtains a .war file and presents it to Relying Parties. The "background check model" applies when the Relying Party verifies the .pop file directly or through a trusted Verifier.

Freshness: RATS freshness mechanisms (nonces, timestamps) align with

the session-entropy and external-anchor mechanisms in Proof of Process. VDF proofs provide an additional freshness dimension: evidence of elapsed time.

Endorsements and Reference Values: Hardware attestation in the hardware-section corresponds to RATS Endorsements. Calibration data serves as Reference Values for VDF timing verification.

For RATS-specific security guidance, implementers should also consult the RATS security considerations in RFC 9334 Section 11.

32.10. Process Score Construction

The Verifier evaluates Evidence across three dimensions, each producing a component score in the range [0.0, 1.0]:

1. Residency (R): Strength of hardware binding, from software-only (0.0-0.7) through TPM attestation (0.7-0.9) to TEE-captured input events (0.9-1.0).

2. Sequence (S): VDF chain integrity and temporal plausibility, including monotonic ordering, calibration consistency, and external anchor corroboration.

3. Behavioral Consistency (B): Whether the behavioral metrics in the evidence chain reflect a consistent generative process, derived from spectral analysis, edit operation distributions, and temporal evolution of per-checkpoint measurements.

The Process Score combines these components:

```
PS = w_R * R + w_S * S + w_B * B

Default weights: w_R = 0.3, w_S = 0.3, w_B = 0.4
Verifier-configurable; weights MUST sum to 1.0
```

The Process Score is a measurement of evidence chain strength. It does not classify content origin, determine authorship identity, or render a verdict. Verifiers include the Process Score in the Attestation Result; Relying Parties apply their own acceptance thresholds.

Evidence satisfying source consistency constraints provides high-confidence assessment. The Process Score reflects the strength of the evidence chain, not a verdict on authorship. Relying Parties apply domain-specific policies to determine what Process Score is acceptable for their use case.

32.10.1. Source Consistency Verification

When ZK proof mechanisms are employed (T3-T4), the proof attests to the following properties without exporting behavioral data:

> "The evidence chain exhibits:
> (1) unbroken VDF temporal ordering across all checkpoints,
> (2) valid entropy commitments bound to content hashes,
> (3) behavioral metrics consistent with interactive editing, and
> (4) no source consistency transitions exceeding threshold."

The ZK proof allows a Verifier to confirm these properties without access to the underlying timing data, preserving author privacy while enabling high-confidence source consistency evaluation.

32.11. Security Properties Summary

This section summarizes the security properties provided by the specification:

32.11.1. Properties Provided

Tamper-Evidence: Modifications to Evidence packets are detectable through cryptographic verification. The hash chain, VDF entanglement, and MAC bindings ensure that alteration invalidates the Evidence.

Cost-Asymmetric Forgery: Producing counterfeit Evidence requires resources (time, compute, entropy generation) disproportionate to legitimate Evidence creation. The forgery-cost-section quantifies these requirements.

Independent Verifiability: Evidence can be verified by any party without access to the original device, without trust in the Attester's infrastructure, and without network connectivity (except for external anchors).

Privacy by Construction: Document content is never stored in Evidence. Behavioral data is aggregated before inclusion. The specification enforces privacy through structural constraints, not policy.

Temporal Ordering: VDF chain construction provides tamper-evident relative ordering of checkpoints with forgery costs bounded by VDF recomputation time. External anchors provide absolute time binding.

Behavioral Binding: Jitter Seal entanglement binds captured

behavioral entropy to the segment chain, making Evidence transplantation infeasible.

32.11.2. Properties NOT Provided

Tamper-Proof: Evidence CAN be forged given sufficient resources. The specification makes forgery costly, not impossible.

Identity Proof: Evidence does NOT prove who operated the device. It proves that input events occurred on a device, not that a specific person produced them.

Intent Proof: Evidence does NOT prove why actions occurred. Observable behavior is documented; mental states are not.

Content Origin Proof: Evidence does NOT prove where ideas came from. The input process is documented; the cognitive source is not.

Absolute Certainty: All security properties are bounded by explicit assumptions. No claim is made to be absolute, irrefutable, or guaranteed.

33. Privacy Considerations

This section consolidates privacy analysis for the witnessd Proof of Process specification. It references and extends the per-section privacy considerations defined in Section 22, Section 25.6.6, and Section 9.17.3.

Privacy is a core design goal of this specification, not an afterthought. The protocol implements privacy-by-construction: structural constraints that make privacy violations architecturally impossible, rather than relying on policy or trust. This approach follows the guidance of [RFC6973] (Privacy Considerations for Internet Protocols).

33.1. Privacy by Construction

The witnessd evidence model enforces privacy through architectural constraints that cannot be circumvented without fundamentally modifying the protocol.

33.1.1. No Document Content Storage

Evidence packets contain cryptographic hashes of document states, never the document content itself. This is a structural invariant:

* Content Hash Binding:

The document-ref structure (CDDL key 5 in evidence-packet) contains only a hash-value of the final document content, the byte-length, and character count. The content itself is never included in the Evidence packet.

* Checkpoint Content Hashes:

  Each checkpoint (key 4: content-hash) contains a hash of the document state at that point. An adversary with the Evidence packet but not the document cannot recover content from these hashes.

* Edit Deltas Without Content:

  The edit-delta structure (key 7 in checkpoint) records chars-added, chars-deleted, insertions, deletions, and replacements as counts only. No information about what characters were added or deleted is included.

This design enables verification of process without revealing what was written, supporting confidential document workflows where the evidence must be verifiable but the content must remain private.

33.1.2. No Keystroke Capture

The specification captures inter-event timing intervals without recording which keys were pressed:

* Timing-Only Measurement:

  Jitter-binding captures millisecond intervals between input events. The interval "127ms" carries no information about whether the interval was between ’a’ and ’b’ or between ’x’ and ’y’.

* No Character Mapping:

  Timing intervals are stored in observation order without any association to specific characters, words, or semantic content.

* No Keyboard Event Codes:

  Scan codes, virtual key codes, and other keyboard identifiers are not recorded. The specification treats all input events uniformly as timing sources.

This architecture ensures that even with complete access to an Evidence packet, no information about what was typed can be reconstructed.

33.1.3. No Screenshots or Screen Recording

The specification explicitly excludes visual capture mechanisms:

* No screenshot capture at checkpoints or any other time
* No screen recording or video capture
* No window title or application name logging
* No clipboard content capture (only timing of clipboard events for monitoring-dependent absence claims, and only event counts, not content)

Visual content capture would fundamentally violate the content-agnostic design and is architecturally excluded.

33.1.4. Local Evidence Generation

Evidence is generated entirely on the Attester device with no network dependency:

* No Telemetry:

  The Attesting Environment does not transmit telemetry, analytics, or any behavioral data to external services.

* No Cloud Processing:

  All cryptographic computations (hashing, VDF, signatures) occur locally. No document content or behavioral data is sent to cloud services for processing.

* Optional External Anchors:

  The only network communication is optional: external anchors (RFC 3161 [RFC3161], [OpenTimestamps], blockchain) transmit only cryptographic hashes, never document content or behavioral data.

Users can generate and verify Evidence in fully air-gapped environments. External anchors enhance evidence strength but are not required.

33.2. Data Minimization

Following RFC 6973 Section 6.1, the specification minimizes data collection to what is strictly necessary for evidence generation and verification.

33.2.1. Data Collected

The following data IS collected and included in Evidence packets:

Timing Histograms: Inter-event timing intervals aggregated into histogram buckets (jitter-summary, key 3 in jitter-binding). Bucket boundaries are coarse (RECOMMENDED: 0, 50, 100, 200, 500, 1000, 2000, 5000ms) to prevent precise interval reconstruction.

Edit Statistics: Character counts for additions, deletions, and edit operations (edit-delta structure). These are aggregate counts, not positional data.

Checkpoint Hashes: Cryptographic hashes of document states at each checkpoint. One-way functions; content cannot be recovered.

VDF Proofs: Verifiable Delay Function outputs proving minimum elapsed time. These are computational proofs, not behavioral data.

Optional: Raw Timing Intervals: The raw-intervals field (key 5 in jitter-binding) MAY be included for enhanced verification. This is OPTIONAL and user-controlled. When omitted, only histogram aggregates are included.

33.2.2. Data NOT Collected

The following data is explicitly NOT collected:

* Document content (text, images, formatting)
* Individual characters or words typed
* Keyboard scan codes or key identifiers
* Screenshots or visual captures
* Screen recordings or video
* Clipboard content (only event timing)
* Window titles or application names
* User names, email addresses, or identifiers (optional: author declaration is user-controlled)
* IP addresses or network identifiers

* Location data

33.2.3. Disclosure Levels

The specification supports tiered disclosure through optional fields:

Level	Data Included	Privacy Impact
Minimal	Hashes, VDF proofs, histogram summaries only	Lowest
Standard	+ Presence challenges, forensics section	Low-Moderate
Enhanced	+ Raw timing intervals, keystroke section	Moderate
Maximum	+ Hardware attestation, absence claims	Higher

Table 28

Users SHOULD select the minimum disclosure level that meets their verification requirements. Higher tiers provide stronger evidence at the cost of revealing more behavioral data.

33.3. Biometric-Adjacent Data

Keystroke timing data, while not traditionally classified as biometric, has biometric-adjacent properties that warrant special consideration. This section addresses regulatory considerations and mitigation measures.

33.3.1. Identification Risks

Research has demonstrated that keystroke dynamics can serve as a behavioral biometric:

* Individual Identification:

  Detailed timing patterns can theoretically distinguish individuals with high accuracy across sessions.

* State Detection:

Timing variations may correlate with cognitive state, fatigue, stress, or physical condition.

* Re-identification Risk:

  If an adversary has access to multiple Evidence packets from the same author, timing patterns might enable linkage across sessions even without explicit identity.

33.3.2. Re-identification Risk Mitigation

To mitigate re-identification risk while preserving correlation utility, the protocol implements multiple layered defenses:

* *Timing Value Clipping:* All timing values are clipped to the range [0, 5000ms], bounding the sensitivity of timing data and preventing outlier values from leaking behavioral information.

* *Histogram Bucketing:* Raw intervals are aggregated into coarse histogram buckets before commitment, reducing temporal resolution below the threshold required for biometric fingerprinting while preserving sufficient fidelity for the Spearman rho 0.7 correlation check.

* *Hurst Exponent Validation:* Only intervals exhibiting valid long-range temporal dependence ($H \in [0.55, 0.85]$) are accepted, filtering synthetic sequences that lack genuine behavioral dynamics.

33.3.3. Isochronous Data Release (Heartbeat Quantization)

To prevent side-channel leakage via packet arrival timing, the Attesting Environment MUST implement "Isochronous Emission." Rather than transmitting jitter metrics as they are captured, the system buffers the data and releases it in fixed-interval beats (e.g., every 5000ms).

This rigid quantization eliminates the information leakage inherent in the burstiness of the user’s typing. An adversary observing the network traffic sees only a constant heartbeat, forcing them to rely entirely on the clipped and bucketed histogram content with zero metadata about the temporal structure of the input stream.

33.3.4. Key Rotation for Privacy

Timing data accumulated across multiple sessions from the same signing key provides more information for cross-session linkage attacks. Periodic key rotation limits the temporal window available for such analysis.

33.3.4.1. Key Rotation Requirements

Signing key rotation policies limit the accumulation of timing data under a single key identity:

Monthly Rotation (REQUIRED): Implementations MUST rotate signing keys at least monthly. Key metadata SHOULD track the next rotation date and session count since key generation.

Weekly Rotation (RECOMMENDED): For high-frequency evidence generation scenarios (more than 4 sessions per week), weekly key rotation is RECOMMENDED to further limit cross-session analysis windows.

Session-Based Rotation: Implementations MAY implement automatic key rotation after a configurable number of sessions (e.g., 20 sessions) rather than purely time-based rotation.

33.3.4.2. Rotation Verification

Verifiers SHOULD validate key rotation compliance:

1. Verify that key-valid-from is before the evidence packet creation timestamp and key-valid-until is after.

2. Verify that the key validity period does not exceed the maximum allowed rotation interval (e.g., 31 days for monthly).

3. If session counts are tracked, verify they are within recommended limits for the rotation policy.

Rotation policy violations SHOULD be reported as caveats in the attestation result but do not invalidate the evidence packet. The primary evidence (VDF, jitter, content binding) remains valid.

33.3.5. Regulatory Considerations

Implementations and deployments should consider applicable privacy regulations:

GDPR (EU/EEA): Keystroke dynamics may constitute "special categories

of personal data" under Article 9 if used for identification purposes. Implementations should document whether timing data is used for identification (prohibited without explicit consent) or solely for process evidence (may fall under different legal basis).

CCPA (California): Biometric information is covered under CCPA Section 1798.140(b). Users have rights to know, delete, and opt-out. The local-only processing model simplifies compliance.

BIPA (Illinois): Illinois Biometric Information Privacy Act has strict requirements for biometric data collection, including written policies and consent. Deployments in Illinois should consult legal counsel.

The specification’s local-only processing model and user control over data disclosure support compliance, but legal interpretation varies by jurisdiction.

33.3.6. User Disclosure Requirements

Implementations MUST inform users about behavioral data collection:

1. Clear notification that timing data is captured during authoring
2. Explanation of what timing data reveals and does not reveal
3. Disclosure of where Evidence packets may be transmitted
4. User control over disclosure levels (histogram-only vs. raw)
5. Instructions for disabling timing capture if desired
6. Process for reviewing and deleting captured data

These disclosures SHOULD be presented before Evidence generation begins, not buried in terms of service.

33.4. Salt Modes for Content Privacy

The hash-salt-mode field (CDDL lines 164-168) enables privacy-preserving verification scenarios where document binding should not be globally verifiable.

33.4.1. Unsalted Mode (Value 0)

```
content-hash = H(document-content)
```

Properties:

* Anyone with the document can verify the binding
* No additional secret required for verification
* Document existence can be confirmed by any party with content

Use cases:

* Public documents where verification should be open
* Academic submissions where verifiers have document access
* Published works where authorship claims should be checkable

Privacy implications: Anyone who obtains both the document and the Evidence packet can confirm the binding. If document confidentiality matters, consider salted modes.

33.4.2. Author-Salted Mode (Value 1)

```
content-hash = H(salt || document-content)
salt-commitment = H(salt)
```

Properties:

* Author generates and retains the salt
* Evidence packet contains salt-commitment, not salt
* Author selectively reveals salt to chosen verifiers
* Without salt, document-hash relationship cannot be verified

Use cases:

* Confidential documents where author controls verification
* Selective disclosure to specific reviewers or institutions
* Manuscripts under review before publication

Privacy implications: The author has exclusive control over who can verify the document binding. The salt should be stored securely; loss of salt means verification becomes impossible.

33.4.3. Salt Requirements

* Salts MUST be cryptographically random (minimum 256 bits)
* Salts MUST NOT be derived from predictable values
* Salt-commitment prevents brute-force guessing for short documents
* Salt loss makes verification impossible; backup appropriately
* Salt transmission should use secure channels

33.5. Identity and Pseudonymity

The specification supports multiple identity postures, from fully anonymous to strongly identified, with user control over disclosure.

33.5.1. Anonymous Evidence Generation

Evidence packets CAN be generated without any identity disclosure:

* The declaration field (key 17 in evidence-packet) is OPTIONAL
* Within declaration, author-name (key 3) and author-id (key 4) are both OPTIONAL
* Device keys can be ephemeral, not linked to identity
* Evidence proves process characteristics without revealing who

Anonymous evidence is suitable for contexts where process documentation matters but author identity is irrelevant or should remain confidential.

33.5.2. Pseudonymous Evidence

Pseudonymous use links evidence to a consistent identifier without revealing real-world identity:

* author-id can be a pseudonymous identifier
* Device key provides cryptographic continuity without identity
* Multiple works can be linked to same pseudonym if desired
* Real identity can remain undisclosed

Pseudonymous evidence enables reputation building without identity exposure.

33.5.3. Identified Evidence

For contexts requiring identity binding:

* author-name and author-id can be populated with real identity
* Declaration signature (key 6) binds identity claim to evidence
* Hardware attestation can strengthen device-to-person binding
* External identity verification is outside specification scope

Identity strength depends on the verification context, not the specification. The specification provides the mechanism for identity claims; verification of those claims is a deployment concern.

33.5.4. Device Binding Without User Identification

Hardware attestation (hardware-section) binds evidence to a specific device without necessarily identifying the user:

* Device keys are bound to hardware (TPM, Secure Enclave)
* Evidence proves generation on a specific device
* Device ownership is a separate question from evidence generation
* Multiple users of same device produce device-linked evidence

Device binding strengthens evidence integrity without requiring user identification. It proves "this device" without proving "this person."

33.6. Data Retention and Deletion

Following RFC 6973 Section 6.2, this section addresses data lifecycle considerations.

33.6.1. Evidence Packet Lifecycle

Evidence packets are designed as archival artifacts:

Creation: Evidence accumulates during authoring session(s). Packet is finalized when authoring is complete.

Distribution: Packet may be transmitted to Verifiers, stored alongside documents, or archived for future verification needs.

Retention: Retention period depends on use case. Legal documents may require indefinite retention; other contexts may allow shorter periods.

Deletion: Once distributed, deletion from all recipients may be impractical. Authors should consider disclosure scope before distribution.

33.6.2. User Rights to Deletion

Users have the following deletion capabilities:

* Local Data:

  Evidence stored locally can be deleted at any time by the author. Implementations SHOULD provide clear deletion mechanisms.

* Distributed Evidence:

  Once Evidence is transmitted to Verifiers or Relying Parties, deletion depends on those parties' policies. The specification cannot enforce deletion of distributed data.

* Attestation Results:

  .war files produced by Verifiers are controlled by Verifiers. Authors may request deletion under applicable privacy laws.

Authors should understand that distributing Evidence creates copies outside their control. Privacy-sensitive authors should limit distribution scope.

33.6.3. External Anchor Permanence

External anchors have special retention characteristics:

RFC 3161 Timestamps: TSA records may be retained by the timestamp authority per their policies. Typically includes the hash committed, not any document or behavioral data.

Blockchain Anchors: Blockchain records are permanent and immutable by design. The anchored hash cannot be deleted from the blockchain. This is a feature for evidence permanence but has privacy implications.

OpenTimestamps: OTS proofs reference Bitcoin transactions, which are permanent. The proof structure can be deleted locally, but the Bitcoin transaction remains.

Users concerned about data permanence should carefully consider whether to use blockchain-based external anchors. RFC 3161 timestamps offer similar evidentiary value with more conventional retention policies.

IMPORTANT: Only cryptographic hashes are anchored, never document content or behavioral data. The permanent record is a hash, not the underlying information.

33.7. Third-Party Disclosure

This section addresses what information is disclosed to various parties in the verification workflow, following RFC 6973 Section 5.2 on disclosure.

33.7.1. Information Disclosed to Verifiers

When an Evidence packet (.pop) is submitted for verification, the Verifier learns:

* Document hash (content-hash) - NOT the content itself
* Document size (byte-length, char-count)
* Authoring timeline (checkpoint timestamps, VDF durations)
* Behavioral statistics (timing histograms, entropy estimates)
* Edit patterns (aggregate counts, not content)
* Optional: Raw timing intervals if disclosed
* Optional: Author identity if declared
* Optional: Device attestation if included

Verifiers SHOULD NOT:

* Retain Evidence packets beyond verification needs
* Use behavioral data for purposes beyond verification
* Attempt to re-identify anonymous authors from behavioral patterns

* Share Evidence data with unauthorized parties

Implementations MAY define Verifier privacy policies that authors can review before submitting Evidence.

33.7.2. Information Disclosed to Relying Parties

Relying Parties consuming Attestation Results (.war) learn:

* Verification verdict (forensic-assessment)
* Confidence score
* Verified claims (specific thresholds met)
* Caveats and limitations
* Verifier identity
* Reference to the original Evidence packet (packet-id)

The .war file is designed to provide necessary trust information without full Evidence disclosure. Relying Parties needing more detail can request the original .pop file.

33.7.3. Minimizing Disclosure

Authors concerned about disclosure can:

1. Use minimal disclosure tier (histogram-only, no raw intervals)
2. Omit optional sections (keystroke-section, absence-section)
3. Use author-salted mode to control verification access
4. Omit declaration or use pseudonymous identity
5. Select Verifiers with strong privacy policies
6. Limit distribution to necessary Relying Parties

33.8. Cross-Session Correlation

This section addresses risks of behavioral fingerprinting across sessions and mitigation measures.

33.8.1. Correlation Risks

Multiple Evidence packets from the same author may enable linkage:

Behavioral Fingerprinting: Keystroke timing patterns exhibit individual characteristics that persist across sessions. An adversary with multiple Evidence packets could potentially link them to the same author even without explicit identity.

Device Fingerprinting: If device keys are reused across sessions, Evidence packets are cryptographically linkable. Hardware attestation makes this linkage explicit.

Stylometric Correlation: Edit pattern statistics (though not content) may correlate with writing style. Combined with timing data, this could strengthen cross-session linkage.

33.8.2. Device Key Rotation

To limit cross-session correlation via device keys:

* Session Keys:

  Use per-session derived keys rather than a single device key. HKDF with session-specific info prevents direct linkage.

* Periodic Rotation:

  Rotate device keys periodically (RECOMMENDED: annually). Evidence packets signed with different keys are not cryptographically linked.

* Context-Specific Keys:

  Use different keys for different contexts (e.g., work vs. personal) to prevent cross-context linkage.

33.8.3. Session Isolation Properties

The specification provides inherent session isolation:

* Each Evidence packet has a unique packet-id (UUID)

* VDF chains are session-specific (session entropy in genesis)

* No protocol mechanism links sessions together

* Jitter data is bound to specific segment-based Merkle trees

Cross-session linkage requires external analysis, not protocol features. The specification does not provide linkage mechanisms.

33.8.4. Additional Mitigations

Authors concerned about cross-session correlation can:

1. Use coarser histogram buckets to reduce timing precision
2. Omit raw-intervals field
3. Vary devices for different document contexts
4. Use different pseudonyms for different contexts
5. Limit Evidence distribution to minimize adversary access to multiple packets

33.9. Privacy Threat Analysis

Following RFC 6973 Section 5, this section analyzes specific privacy threats.

33.9.1. Surveillance

The specification is designed to resist surveillance:

* No content transmission prevents content-based surveillance
* Local-only processing prevents network monitoring
* Optional external anchors transmit only hashes
* No telemetry or analytics collection

The primary surveillance risk is through Evidence packet distribution. Authors control this distribution.

33.9.2. Stored Data Compromise

If Evidence packets are compromised:

* Document content is NOT exposed (hash-only)
* Behavioral patterns MAY be exposed (timing data)
* Authoring timeline is exposed (timestamps)

* If identity declared, author identity is exposed

Mitigation: Encrypt Evidence packets at rest. Use access controls for stored Evidence. Limit retention period where appropriate.

33.9.3. Correlation

Correlation threats are addressed in Section 33.8. Key mitigations include key rotation, histogram aggregation, and distribution limiting.

33.9.4. Identification

Re-identification threats:

* Anonymous Evidence MAY be re-identifiable through behavioral patterns

* Histogram aggregation significantly reduces this risk

* Raw interval disclosure increases re-identification risk

* Device attestation explicitly identifies devices

Authors requiring strong anonymity should use minimal disclosure tier without raw intervals and without device attestation.

33.9.5. Secondary Use

Evidence data could theoretically be used for purposes beyond verification:

* Behavioral analysis for profiling

* Productivity monitoring

* Training data for machine learning

Mitigation: The specification does not prevent secondary use by data recipients. Authors should consider Verifier and Relying Party policies before disclosure. Implementations MAY include usage restrictions in Evidence packet metadata.

33.9.6. Disclosure

Unauthorized disclosure of Evidence packets:

* Authors control initial distribution

* Recipients may further distribute; specification cannot prevent

* Salted modes limit utility of leaked Evidence

* Anonymous Evidence limits identity exposure on leak

Authors should treat Evidence packets as potentially sensitive and limit distribution to trusted parties.

33.9.7. Exclusion

The risk that authors cannot participate in systems if they decline Evidence generation:

* Evidence generation is voluntary

* Disclosure levels are user-controlled

* Relying Parties may require Evidence for certain contexts

* The specification does not mandate deployment contexts

Deployments should consider whether Evidence requirements create exclusionary effects and provide alternatives where appropriate.

33.10. Privacy Properties Summary

This section summarizes the privacy properties provided and not provided by the specification.

33.10.1. Privacy Properties Provided

Content Confidentiality: Document content is never stored in Evidence. Verification can occur without content access (using salted modes).

Keystroke Privacy: Individual keystrokes are never recorded. Only timing intervals between events are captured, without character association.

Local Control: All data processing occurs locally. No external services required for Evidence generation.

Disclosure Control: Authors control Evidence distribution, disclosure level, and identity exposure.

Pseudonymity Support: Evidence can be generated and verified without real-world identity disclosure.

Selective Verification: Salted modes enable author-controlled verification access.

33.10.2. Privacy Limitations

Behavioral Data Exposure: Timing data reveals behavioral patterns. While aggregated, this data has biometric-adjacent properties.

Distribution Not Controlled: Once Evidence is distributed, the specification cannot control further dissemination or use.

Cross-Session Linkage Risk: Multiple Evidence packets may be linkable through behavioral analysis, even with different identities.

External Anchor Permanence: Blockchain anchors create permanent records that cannot be deleted.

Metadata Disclosure: Evidence packets reveal document size, authoring timeline, and edit statistics even without content.

33.10.3. Recommendations for Privacy-Sensitive Deployments

1. Use minimal disclosure tier (histogram-only, no raw intervals)
2. Consider coarser histogram buckets for enhanced privacy
3. Use author-salted mode for confidential documents
4. Avoid blockchain anchors if deletion rights are important
5. Rotate device keys periodically
6. Limit Evidence distribution to necessary parties
7. Review Verifier privacy policies before submission
8. Consider pseudonymous identities where appropriate
9. Provide clear user disclosures about data collection
10. Implement data retention policies aligned with use case

34. Error Handling and Recovery

Implementations MUST handle verification failures and evidence deficiencies according to the following taxonomy. Errors are classified by their impact on the forensic-assessment verdict and the required recovery actions.

Error Code	Description	Impact	Recovery Action
ERR_VDF_MISMATCH	VDF output recomputation failed	FATAL (Evidence Invalid)	Reject Evidence packet
ERR_ENTROPY_LOW	Jitter entropy below tier threshold	WARNING (Reduced Confidence)	Flag in Attestation Result caveats
ERR_CALIB_GAPPED	Missing or untrusted calibration	MAJOR (Tier downgrade)	Treat as Basic tier evidence
ERR_CHAIN_GAP	Non-consecutive sequence numbers	FATAL (Evidence Tampered)	Reject Evidence packet

Table 29: Error Taxonomy

35. Protocol Versioning and Migration

PPPP uses semantic versioning. Version 1.1 introduced mandatory VDF-jitter entanglement. Verifiers MUST support backwards compatibility for v1.0 (non-entangled) packets but SHOULD flag them with a "Legacy" warning. Future versions involving breaking changes to the VDF iteration function will increment the major version and require a new CBOR tag.

36. Normative Error Handling

Verifiers MUST implement the following error handling procedures:

* *ERR_VDF_MISMATCH*: If the recomputed VDF output does not match the reported segment output, the entire evidence chain MUST be rejected as fraudulent.

* *ERR_RESIDENCY_STALE*: If the hardware attestation quote (TPM/SE) is older than 24 hours, the Residency (R) score MUST be degraded to 0.5.

* *ERR_ENTROPY_LOW*: If the Behavioral Consistency (B) metric is below 0.7, the result MUST flag "Unverifiable Interactive Process" but SHOULD NOT invalidate the Duo (R, S) proofs.

37. IANA Considerations

This document requests creation of the "Proof of Process VDF Algorithms" registry. This registry contains identifiers for Verifiable Delay Function algorithms used in Evidence packets.

This document utilizes the following registered CBOR tags and SMI Private Enterprise Number (PEN) registry, similar to the structured identity anchoring found in RFC 9334.

37.1. CBOR Tags Registration

This document requests the registration of the following tags in the "CBOR Tags" registry [IANA.cbor-tags]:

* Tag: 1347571280, Data Item: PPPP Evidence, Description: Proof of Process Provenance Evidence Packet

* Tag: 1463894560, Data Item: PPPP Result, Description: Proof of Process Provenance Verification Result

* Tag: 1347571281, Data Item: PPPP Compact Ref, Description: Compact Reference to PPPP Evidence

37.2. CBOR Tags Registry

The coordination of tags mirrors the suite of identifiers used in COSE architectures.

Tag	Hex	ASCII	Data Item	Semantics
1347571280	0x50505020	"PPPP"	map	PPPP Evidence Packet (.pppp)
1463894560	0x57415220	"WAR "	map	PPPP Result (.war)
1347571281	0x50505021	"PPP!"	map	PPPP Supplemental Data

Table 30: CBOR Tags Summary

37.3. Private Enterprise Number (PEN) Registry

This specification utilizes SMI Private Enterprise Number **65074**, similar to the vendor-specific OID trees used in X.509 PKI.

```
    1.3.6.1.4.1.65074
      .1 - PPPP Protocol Core
        .1 - Invariant: Residency (R)
        .2 - Invariant: Sequence (S)
        .3 - Invariant: Behavioral Consistency (B)
      .2 - PPPP Evidence Tiers
      .3 - PPPP Revocation Reasons
```

37.4. Tag for Writers Authenticity Report (0x57415220)

The tag value 1463894560 (hexadecimal 0x57415220) corresponds to the ASCII encoding of "WAR " and identifies Writers Authenticity Report structures. This tag encapsulates an Attestation Result produced by Verifiers after appraising Proof of Process Evidence Packets.

The WAR format conveys verification verdicts, confidence scores, and forensic assessments following the IETF RATS (Remote ATtestation procedureS) architecture. A dedicated tag enables zero-configuration identification of attestation results, allowing Relying Parties to distinguish verification outcomes from raw evidence without content-type negotiation.

The tagged data item is a CBOR map conforming to the attestation-result structure defined in Section 9.14 .

37.5. Tag for Compact Evidence Reference (0x50505021)

The tag value 1347571281 (hexadecimal 0x50505021) corresponds to the ASCII encoding of "PPP!" and identifies Compact Evidence Reference structures. This tag encapsulates a cryptographic pointer to a full Proof of Process Evidence Packet.

Compact Evidence References are designed for embedding in space-constrained contexts such as document metadata (PDF XMP, EXIF), QR codes, NFC tags, git commit messages, and protocol headers. The compact reference contains the packet-id, chain-hash, document-hash, and a summary with a cryptographic signature binding all fields. A dedicated tag enables zero-configuration detection and verification of authorship claims without transmitting full evidence packets.

The tagged data item is a CBOR map conforming to the compact-evidence-ref structure defined in Section 30 .

37.6. Justification for Dedicated Tags

The four-byte tag values were chosen for the following reasons:

* *Self-describing format:* The ASCII-based mnemonics ("PPPP", "WAR", "PPP!") enable immediate visual identification in hex dumps and debugging contexts.

* *Zero-configuration detection:* Applications can identify Proof of Process data without prior context or content-type negotiation.

* *Interoperability:* Standardized tags enable diverse implementations (academic systems, publishing platforms, verification services) to recognize and process data without coordination.

* *Compact encoding:* Despite being 4-byte tags, CBOR’s efficient encoding minimizes overhead for these application-specific semantic markers.

38. Entity Attestation Token Profiles Registry

This document requests registration of an EAT profile in the "Entity Attestation Token Profiles" registry established by [RFC9711].

Profile URI	Description	Reference
https://example.com/rats/eat/profile/pop/1.0	witnessd Proof of Process Evidence Profile	[this document]

Table 31: EAT Profile Registration

Note: The URI https://example.com/rats/eat/profile/pop/1.0 is provisional during individual submission. Upon working group adoption, registration of an IANA-hosted profile URI will be requested (e.g., urn:ietf:params:rats:eat:profile:pop:1.0).

The profile defines the following characteristics:

Profile Version: 1.0

Applicable Claims: All standard EAT claims per RFC 9711, plus the custom claims defined in Section 39.

Evidence Format: CBOR-encoded evidence-packet structure with semantic tag 1347571280.

Attestation Result Format: CBOR-encoded attestation-result structure with semantic tag 1463894560.

Domain: Document authorship process attestation, behavioral evidence for content provenance.

39. CBOR Web Token Claims Registry

This document requests registration of custom claims in the "CBOR Web Token (CWT) Claims" registry [IANA.cwt]. These claims are used within EAT Attestation Results to convey witnessd-specific assessment data.

Initial registration is requested in the private-use range (-70000 to -70010) to enable early implementation. Upon standards track advancement, permanent positive claim keys will be requested.

Claim Name	Claim Key	Claim Value Type	Claim Description	Reference
pop-forensic-assessment	-70000	unsigned integer	Forensic assessment enumeration value (0-5) indicating the Verifier's assessment of behavioral evidence consistency with human authorship patterns.	[this document]
pop-presence-score	-70001	unsigned integer (millibits)	Presence challenge response score in range 0-1000 (millibits, divide by 1000 for 0.0-1.0) representing the ratio of successfully completed human presence challenges.	[this document]
pop-evidence-tier	-70002	unsigned integer	Evidence tier classification (1-4) indicating the comprehensiveness of evidence collected: 1=Basic, 2=Standard, 3=Enhanced, 4=Maximum.	[this document]
pop-ai-composite-score	-70003	unsigned integer (millibits)	AI indicator composite score in range 0-1000 (millibits, divide by 1000 for 0.0-1.0) derived from behavioral	[this document]

			forensic analysis. Lower values indicate patterns more consistent with human authorship.		

Table 32: Custom CWT Claims Registration

The forensic-assessment enumeration values for pop-forensic-assessment are defined as:

Value	Name	Description
0	not-assessed	Verification incomplete or not attempted
1	manual-composition-consistent	Evidence strongly consistent with manual composition patterns patterns
2	manual-composition-likely	Evidence suggests manual composition patterns patterns
3	inconclusive	Evidence neither confirms nor refutes claims
4	automated-assisted-likely	Evidence consistent with automated assistance patterns in authorship
5	automated-insertion-consistent	Evidence strongly consistent with bulk automated insertion patterns

Table 33: Forensic Assessment Enumeration Values

40. New Registries

This document requests IANA to create three new registries under a new "witnessd Proof of Process" registry group.

40.1. Proof of Process Claim Types Registry

This document requests creation of the "Proof of Process Claim Types" registry. This registry contains the identifiers for absence claims that can be asserted and verified in Evidence packets.

40.1.1. Registration Procedures

The registration procedures for this registry depend on the claim type range:

Range	Category	Registration Procedure
1-15	Chain-verifiable claims	Specification Required
16-63	Monitoring-dependent claims	Specification Required
64-127	Environmental claims	Expert Review
128-255	Private use	First Come First Served

Table 34: Claim Types Registration Procedures

Chain-verifiable claims (1-15) are claims that can be proven solely from the Evidence packet without trusting the Attesting Environment beyond data integrity. These claims require a published specification demonstrating verifiability.

Monitoring-dependent claims (16-63) require trust in the Attesting Environment's accurate reporting of monitored events. Specifications MUST document the trust assumptions.

Environmental claims (64-127) relate to the execution environment or external conditions. Expert review ensures claims are well-defined and implementable.

Private use claims (128-255) are available for implementation-specific extensions without coordination.

40.1.2. Registration Template

Registrations MUST include the following fields:

Claim Type Value: Integer identifier in the appropriate range

Claim Name: Human-readable name (lowercase with hyphens)

Category: One of: computationally-bound, monitoring-dependent, environmental, or private-use

Description: Brief description of what the claim asserts

Verification Method: How the claim is verified (for non-private-use claims)

Reference: Document defining the claim

40.1.3. Initial Registry Contents

The initial contents of the "Proof of Process Claim Types" registry are as follows:

40.1.3.1. Computationally Bound Claims (1-15)

Value	Name	Description	Reference
1	max-single-delta-chars	Maximum characters added in any single checkpoint delta	[this document]
2	max-single-delta-bytes	Maximum bytes added in any single checkpoint delta	[this document]
3	max-net-delta-chars	Maximum net character change across the entire chain	[this document]
4	min-vdf-duration-seconds	Minimum total VDF-proven elapsed time in seconds	[this document]
5	min-vdf-duration-per-kchar	Minimum VDF-proven time per thousand characters	[this document]
6	checkpoint-chain-complete	Segment chain has no gaps (all sequence numbers	[this document]

		present)	
7	checkpoint-chain-consistent	All segment hashes and VDF linkages verify correctly	[this document]
8	jitter-entropy-above-threshold	Captured jitter entropy exceeds specified bits threshold	[this document]
9	jitter-samples-above-count	Number of jitter samples exceeds specified count	[this document]
10	revision-points-above-count	Number of revision points (non-monotonic edits) exceeds threshold	[this document]
11	session-count-above-threshold	Number of distinct editing sessions exceeds threshold	[this document]
12	cognitive-load-integrity	Complexity-timing correlation exceeds threshold	[this document]
13	intra-session-consistency	Behavioral timing remains in stable cluster	[this document]
14	complexity-timing-correlation	Information density correlates with timing density	[this document]
15	Unassigned	Reserved for future computationally-bound claims	[this document]

Table 35: Computationally Bound Claim Types

40.1.3.2. Monitoring-Dependent Claims (16-20)

Value	Name	Description	Reference
16	max-paste-event-chars	Maximum characters in any single paste event	[this document]

17	max-clipboard-access-chars	Maximum total characters accessed from clipboard	[this document]
18	no-paste-from-ai-tool	No paste operations from known AI tool applications	[this document]
19	Unassigned	Reserved	[this document]
20	max-insertion-rate-wpm	Maximum sustained insertion rate in words per minute	[this document]
21	no-automated-input-pattern	No detected automated or scripted input patterns	[this document]
22	no-macro-replay-detected	No keyboard macro replay patterns detected	[this document]
23-63	Unassigned	Reserved for future monitoring-dependent claims	[this document]

Table 36: Monitoring-Dependent Claim Types

40.1.3.3. Registration Procedures

Range	Category	Registration Procedure
1-15	Iterated hash VDFs	Standards Action
16-31	Succinct VDFs	Standards Action
32-63	Experimental	Expert Review

Table 37: VDF Algorithms Registration Procedures

Iterated hash VDFs (1-15) are algorithms where verification requires recomputation. Standards Action ensures thorough security analysis.

Succinct VDFs (16-31) are algorithms with efficient verification (e.g., [Pietrzak2019], [Wesolowski2019]). Standards Action ensures cryptographic soundness.

Experimental algorithms (32-63) may be registered with Expert Review for research and interoperability testing. Production use requires promotion to Standards Action ranges.

40.1.3.4. Registration Template

Registrations MUST include the following fields:

Algorithm Value: Integer identifier in the appropriate range

Algorithm Name: Human-readable name

Category: One of: iterated-hash, succinct, or experimental

Parameters: Required CDDL structure for algorithm parameters

Verification Complexity: Asymptotic verification complexity

Security Assumptions: Cryptographic assumptions for security

Reference: Document specifying the algorithm

40.1.3.5. Initial Registry Contents

Value	Name	Category	Verification	Reference
1	iterated-sha256	iterated-hash	O(n)	[this document]
2	iterated-sha3-256	iterated-hash	O(n)	[this document]
3-15	Unassigned	iterated-hash	-	[this document]
16	pietrzak-rsa3072	succinct	O(log n)	[this document]
17	wesolowski-rsa3072	succinct	O(1)	[this document]
18	pietrzak-class-group	succinct	O(log n)	[this document]

19	wesolowski-class-group	succinct	O(1)	[this document]
20-31	Unassigned	succinct	-	[this document]
32-63	Unassigned	experimental	-	[this document]

Table 38: VDF Algorithms Initial Values

The iterated hash algorithms use the iterated-hash-params CDDL structure (keys 1-2). The succinct algorithms use the succinct-vdf-params CDDL structure (keys 10-11). See Section 24 for detailed specifications.

40.1.4. Proof of Process Entropy Sources Registry

This document requests creation of the "Proof of Process Entropy Sources" registry. This registry contains identifiers for behavioral entropy sources used in Jitter Seal bindings.

40.1.4.1. Registration Procedures

The registration procedure for this registry is Specification Required.

Registrations MUST include a specification describing:

* The input modality or behavioral signal being captured
* The method for converting the signal to timing intervals
* Privacy implications of capturing this entropy source
* Expected entropy density (bits per sample) under typical conditions

40.1.4.2. Registration Template

Registrations MUST include the following fields:

Source Value: Integer identifier

Source Name: Human-readable name (lowercase with hyphens)

Description: Brief description of the entropy source

Privacy Impact: One of: minimal, low, moderate, high

Reference: Document specifying the entropy source

40.1.4.3. Initial Registry Contents

Value	Name	Description	Privacy Impact	Reference
1	keystroke-timing	Inter-key intervals from keyboard input	moderate	[this document]
2	pause-patterns	Gaps between editing bursts (>2 seconds)	low	[this document]
3	edit-cadence	Rhythm of insertions/ deletions over time	low	[this document]
4	cursor-movement	Navigation timing within document	low	[this document]
5	scroll-behavior	Document scrolling patterns	minimal	[this document]
6	focus-changes	Application focus gain/ loss events	low	[this document]

Table 39: Entropy Sources Initial Values

40.2. Media Types Registry

This document requests registration of two media types in the "Media Types" registry [IANA.media-types].

40.2.1. application/vnd.example-pop+cbor Media Type

Type name: application

Subtype name: vnd.example-pop+cbor

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary. As a CBOR format, it contains NUL octets and non-line-oriented data.

Security considerations: This media type contains cryptographically anchored evidence of authorship process. It does not contain active or executable content. Integrity is ensured via a HMAC-SHA256 segment chain and Verifiable Delay Functions (VDFs). Privacy is maintained through author-controlled salting of content hashes as defined in Section 9.10.2. Security considerations of CBOR [RFC8949] apply. See also Section 32 of this document.

Interoperability considerations: While the +cbor suffix allows generic parsing, full semantic validation and behavioral forensic analysis require a witnessd-compatible processor as defined in this specification. The content is a CBOR-encoded evidence-packet structure with semantic tag 1347571280.

Published specification: [this document]

Applications that use this media type: Generation of digital authorship evidence by the witnessd suite and WritersLogic integrated editors. Verification services, document provenance systems, academic integrity platforms.

Fragment identifier considerations: N/A

Additional information: Deprecated alias names for this type: N/A
Magic number(s): 0xD950505020 (CBOR tag encoding at offset 0)
File extension(s): .pop
Macintosh file type code(s): N/A

Person and email address to contact for further information: David Condrey <david@writerslogic.com>

Intended usage: COMMON

Restrictions on usage: N/A

Author: David Condrey

Change controller: WritersLogic Inc.

Provisional registration: No

40.2.2. application/vnd.example-war+cbor Media Type

Type name: application

Subtype name: vnd.example-war+cbor

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary. As a CBOR-encoded format, it contains NUL octets and non-line-oriented data.

Security considerations: This media type conveys the final appraisal result (verdict) of an authorship attestation. (1) It does not contain active or executable content. (2) Integrity and authenticity are provided via a COSE signature [RFC9052] that MUST be verified against the Verifier's public key. (3) The information identifies a specific document by its content hash; privacy is managed through the hash-salting protocols defined in Section 9.10.2. (4) The security considerations for CBOR (RFC 8949) and COSE (RFC 9052) apply. Users are cautioned not to rely on unsigned or unverified .war files for high-stakes authenticity claims. See also Section 32 of this document.

Interoperability considerations: The +cbor suffix allows generic CBOR tools to identify the underlying encoding. This format is a specific profile of the RATS Attestation Result and references a Proof of Process (.pop) evidence packet by UUID as defined in this specification. The content is a CBOR-encoded attestation-result structure with semantic tag 1463894560.

Published specification: [this document]

Applications that use this media type: Verification and display of authorship scores by publishers, academic repositories, literary journals, and the WritersLogic verification suite.

Fragment identifier considerations: N/A

Additional information: Deprecated alias names for this type: N/A
Magic number(s): 0xD957415220 (CBOR tag

encoding at offset 0)
File extension(s): .war
Macintosh file type code(s): N/A

Person and email address to contact for further information: David Condrey <david@writerslogic.com>

Intended usage: COMMON

Restrictions on usage: N/A

Author: David Condrey

Change controller: WritersLogic Inc.

Provisional registration: No

40.3. Designated Expert Instructions

The designated experts for the registries created by this document should apply the following criteria when evaluating registration requests:

40.3.1. Proof of Process Claim Types Registry

For claim types requiring Specification Required:

* The specification MUST clearly define what the claim asserts
* For computationally-bound claims, the specification MUST demonstrate that the claim can be verified solely from the Evidence packet
* For monitoring-dependent claims, the specification MUST document the Attesting Environment trust assumptions
* The claim name SHOULD be descriptive and follow existing naming conventions

For environmental claims requiring Expert Review:

* The specification SHOULD describe implementation considerations
* The claim SHOULD NOT duplicate existing claims
* Privacy implications SHOULD be documented

40.3.2. Proof of Process VDF Algorithms Registry

For experimental algorithms requiring Expert Review:

* The algorithm MUST be documented with sufficient detail for independent implementation
* Security analysis SHOULD be provided, even if preliminary
* The algorithm SHOULD NOT be a minor variant of an existing registered algorithm
* Implementation availability is encouraged but not required

40.3.3. Proof of Process Entropy Sources Registry

For entropy sources requiring Specification Required:

* The specification MUST describe how timing intervals are derived from the entropy source
* Expected entropy density under typical conditions SHOULD be documented
* Privacy implications MUST be clearly stated
* The entropy source SHOULD provide meaningful behavioral signal that cannot be trivially simulated

41. References

41.1. Normative References

[IANA.cbor-tags]
IANA, "CBOR Tags",
<https://www.iana.org/assignments/cbor-tags>.

[IANA.cose]
IANA, "CBOR Object Signing and Encryption (COSE)",
<https://www.iana.org/assignments/cose>.

[IANA.cwt] IANA, "CBOR Web Token (CWT) Claims",
<https://www.iana.org/assignments/cwt>.

[IANA.media-types]
IANA, "Media Types",
<https://www.iana.org/assignments/media-types>.

[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>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[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>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2024, <https://www.rfc-editor.org/info/rfc9334>.

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, December 2024, <https://www.rfc-editor.org/info/rfc9711>.

41.2. Informative References

[Grudin1983]
Grudin, J., "Error patterns in novice and skilled transcription typing", 1983.

[I-D.condrey-rats-pop-examples]
Condrey, D., "Examples of Proof of Process Provenance (PPPP) Evidence Packets and Attestation Results", Work in Progress, Internet-Draft, draft-condrey-rats-pop-examples-01, February 2024, <https://datatracker.ietf.org/doc/html/draft-condrey-rats-pop-examples-01>.

[I-D.condrey-rats-pop-protocol]
Condrey, D., "Proof of Process (PoP): A Verifiable Process Transcript Format", Work in Progress, Internet-Draft, draft-condrey-rats-pop-protocol-00, February 2026, <https://datatracker.ietf.org/doc/html/draft-condrey-rats-pop-protocol-00>.

[I-D.condrey-rats-pop-schema]
Condrey, D., "PPPP CDDL Schema", Work in Progress, Internet-Draft, draft-condrey-rats-pop-schema-01, <https://datatracker.ietf.org/doc/html/draft-condrey-rats-pop-schema-01>.

[I-D.ietf-rats-ar4si]
Birkholz, H., Fossati, T., Pan, W., and E. Voit, "Attestation Results for Secure Interactions", Work in Progress, Internet-Draft, draft-ietf-rats-ar4si, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-ar4si>.

[I-D.ietf-rats-ear]
Fossati, T. and S. Frost, "EAT Attestation Results", Work in Progress, Internet-Draft, draft-ietf-rats-ear, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-ear>.

[I-D.ietf-rats-eat]
Lundblade, L., Mandyam, G., and J. O'Donoghue, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-28, February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-28>.

[I-D.ietf-rats-epoch-markers]
Birkholz, H., Fossati, T., Pan, W., and C. Bormann, "RATS Epoch Markers", Work in Progress, Internet-Draft, draft-ietf-rats-epoch-markers, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-epoch-markers>.

[Kushniruk1991]
Kushniruk, A., "Cognitive processes in the design of user interfaces", Ergonomics 34(10), 1991.

[Mandelbrot1982]
Mandelbrot, B., "The Fractal Geometry of Nature", 1982.

[OpenTimestamps]
Todd, P., "OpenTimestamps: Scalable, Trust-Minimized, Distributed Timestamping with Bitcoin", 2016, <https://opentimestamps.org>.

[Pietrzak2019]
Pietrzak, K., "Simple Verifiable Delay Functions", ITCS 2019, 2019, <https://eprint.iacr.org/2018/627>.

[Rayner1998]
Rayner, K., "Eye movements in reading and information processing: 20 years of research", Psychological Bulletin 124(3), 1998.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/info/rfc3161>.

[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>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Protocol Implementations: The Rough Guide", RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/info/rfc9562>.

[TPM2.0] Trusted Computing Group, "TPM 2.0 Library Specification", 2019, <https://trustedcomputinggroup.org/resource/tpm-library-specification/>.

[Wesolowski2019] Wesolowski, B., "Efficient Verifiable Delay Functions", EUROCRYPT 2019, 2019, <https://eprint.iacr.org/2018/623>.

Acknowledgments

The authors would like to thank the members of the IETF RATS working group for their foundational work on remote attestation architectures that this specification builds upon.

Special thanks to the reviewers and contributors who provided feedback on early drafts of this specification.

Document History

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

draft-condrey-rats-pop-01

Revision addressing working group feedback.

* Reframed core claim around source consistency analysis and decision history rather than authorship verification
* Added Evidence Flow section mapping RATS passport model
* Added Decision History framework capturing edit operation topology without content access
* Added Privacy-Preserving Document Classification
* Added Input Event Trust Boundary with tier-mapped adversary model
* Added Source Consistency transition pattern taxonomy
* Replaced interactive Vise handshake with non-interactive local behavioral analysis consistent with implementation
* Rewrote abstract and introduction for clarity
* Addressed relay, replay, and diversion attack concerns

draft-condrey-rats-pop-00

Initial submission.

* Defined Evidence Packet (.pop) and Attestation Result (.war) formats

* Specified Jitter Seal mechanism for behavioral entropy capture

* Specified VDF mechanisms for temporal ordering proofs

* Defined absence proof taxonomy with trust requirements

* Established forgery cost bounds methodology

* Documented security and privacy considerations

* Requested IANA registrations for CBOR tags, media types, and EAT claims

Appendix: Verification Constraint Summary

For interoperability, PPPP-compliant Verifiers MUST validate the following constraints on Evidence Packets:

1. *VDF Continuity:* H(out_{i-1}, content_i, jitter_i) === in_i for all checkpoints.

2. *Temporal Monotonicity:* Each checkpoint timestamp strictly exceeds its predecessor.

3. *Chain Integrity:* SHA-256 hash chain is unbroken from genesis to final checkpoint.

4. *Entropy Commitment:* HMAC binding between behavioral entropy and checkpoint content is valid.

5. *VDF Sequential Proof:* Pietrzak proof verifies for declared iteration count at each checkpoint.

6. *Source Consistency:* Edit operation distribution and timing patterns evaluated for coherence across checkpoint chain (informational, not pass/fail).

Appendix: VDF Verification Test Vectors

The following test vectors (SHA-256 Iterated Hash) are provided for interoperability testing:

```
Input (Seed): "witnessd-genesis-v1" (hex: 7769746e657373642d67656e657369732d7631)
Iterations: 10,000
Output (Expected): 7d3c9a4f... (Full 32-byte hash)

Input (Entangled): "DST_CHAIN" || H(content) || Output_n-1
Iterations: 50,000
Output (Expected): b1a2c3d4...
```

Author's Address

David Condrey
Writerslogic Inc
San Diego, California,
United States
Email: david@writerslogic.com
URI: https://writerslogic.com