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

D. Condrey
Writerslogic Inc

11 February 2026

Proof of Process (PoP): A Verifiable Process Transcript Format
draft-condrey-rats-pop-protocol-00

Abstract

This document specifies the Proof of Process (PoP) Transcript Format, a structured data model for recording the evolutionary history of a digital document. It defines a canonical set of semantic editing events (Insertion, Deletion, Move, Replacement) and a schema for Tool Receipts, allowing distinct contributors—including human authors, AI agents, and automated editors—to cryptographically sign their specific contributions within a single provenance chain.

The protocol utilizes a hash-linked Provenance DAG (Directed Acyclic Graph) to ensure the integrity and ordering of events. This format enables Verifiers to reconstruct the document's lifecycle and apply flexible, policy-based logic to determine authorship attribution. It is designed to be content-aware but policy-agnostic, supporting diverse use cases ranging from academic integrity to software supply chain security.

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

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 4
3. Functional Scope and Protocol Claims 5
  3.1. Transcript Integrity Claims 5
  3.2. Out of Scope 5
4. Architecture Overview 5
5. Data Model 6
  5.1. Ranges and Identifiers 6
  5.2. Canonical Event Types 6
6. Event Hash Chaining 7
  6.1. Hash Algorithm 7
  6.2. Requirements 7
7. Time Anchoring 8
  7.1. External Anchors 8
  7.2. Sequential Work Anchors 8

7.3. Anchor Verification Rules 8
8. Tool Receipts and Provenance DAG 8
8.1. Receipt Model 9
8.2. Receipt Verification Rules 9
9. Envelope Structure 9
9.1. Policy and Seed Commitments 9
9.2. Claims Field 10
9.3. Optional Zero-Knowledge Bundle 10
10. Verification Procedure 10
10.1. Structural Validation 10
10.2. Hash Chain Validation 10
10.3. Anchor Validation 10
10.4. Receipt Validation 11
10.5. Artifact Binding Validation 11
10.6. Policy Evaluation 11
11. Policy-Defined Evaluation 11
11.1. Policy Commitments 11
11.2. Seeded Evaluation (Optional) 11
11.3. Evaluation Outputs 12
12. Security Considerations 12
12.1. Threat Model Summary 12
12.2. Mitigations 12
12.3. Algorithm Agility 13
13. Privacy Considerations 13
14. IANA Considerations 13
14.1. Media Type Registration 13
15. CDDL Summary (Normative) 14
16. Example Verification Flow (Informative) 16
17. References 16
17.1. Normative References 16
17.2. Informative References 17
Acknowledgments 18
Author’s Address 18

1. Introduction

As digital documents evolve through complex workflows involving human authors, automated tools, and AI assistants, stakeholders require a standardized method to record _what happened_ during the document’s lifecycle. Existing logs are often proprietary, unstructured, or easily mutated.

This document specifies the *Proof of Process (PoP) Transcript Format*, a standardized data model for recording the evolutionary history of a digital artifact. It defines a canonical set of *Semantic Events* (Insert, Delete, Replace, Move) and a mechanism for linking these events into a hash-chained *Provenance DAG (Directed Acyclic Graph)*.

Designed for interoperability, this format allows disparate editing environments (word processors, code editors, CMS platforms) to produce compatible *Process Transcripts*. These transcripts can support *Tool Receipts*, allowing AI agents and automated tools to cryptographically sign and attribute their specific contributions, thereby enabling a "compositional provenance" model where human and machine contributions are clearly demarcated within a single verifiable history.

2. Terminology

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

Artifact A digital object such as a document, code bundle, or media file.

Event A canonical semantic editing operation applied to an artifact state.

Transcript An ordered sequence of events representing a process session.

Envelope A CBOR-encoded container holding transcript commitments, anchors, receipts, and policy hooks.

Anchor A mechanism that binds transcript state to chronology, such as an external timestamp or sequential-work proof.

Receipt A signed statement from a tool indicating that content was produced or transformed, referenced by events.

Provenance DAG A directed acyclic graph constructed from receipts and references between them.

Policy A verifier-defined configuration for evaluation of process evidence.

Policy Commit A cryptographic commitment to a policy descriptor, included to identify the applied policy.

Seed Commit A cryptographic commitment to a seed used to parameterize evaluation functions and/or audits.

Deterministic Encoding CBOR encoding with a canonical ordering as

required for reproducible hashing.

3. Functional Scope and Protocol Claims

This section defines the scope of the transcript format regarding data integrity and provenance visibility.

3.1. Transcript Integrity Claims

A valid PoP Transcript ensures:

* *Event Immutability:* Once an event is recorded and covered by a subsequent hash, it cannot be altered or removed without invalidating the chain.

* *Completeness:* The final hash of the transcript deterministically reconstructs the final state of the document. Any discrepancy between the transcript-derived state and the actual document content indicates tampering.

* *Attribution:* Every event is associated with a specific Actor ID (Key) or Tool Receipt, allowing precise attribution of _who_ or _what_ performed a specific edit.

3.2. Out of Scope

* *Content Quality or Truth:* The transcript records that text was written; it does not verify the factual accuracy or quality of that text.

* *Intent:* The transcript records operations (e.g., "User deleted paragraph A"). It does not record _why_ the operation occurred.

* *Surveillance:* This format is designed for _verification_, not _surveillance_. It supports privacy-preserving disclosure modes (e.g., Zero-Knowledge proofs of statistics) where the raw content of the edits remains private, revealing only the _metrics_ of the process.

4. Architecture Overview

PoP is a portable evidence format intended to be produced by an editing environment (the Producer) and consumed by a Verifier. A third party (Anchor Service) MAY provide timestamps or other anchoring evidence. Tools (e.g., language models, grammar checkers, converters) MAY provide signed receipts.

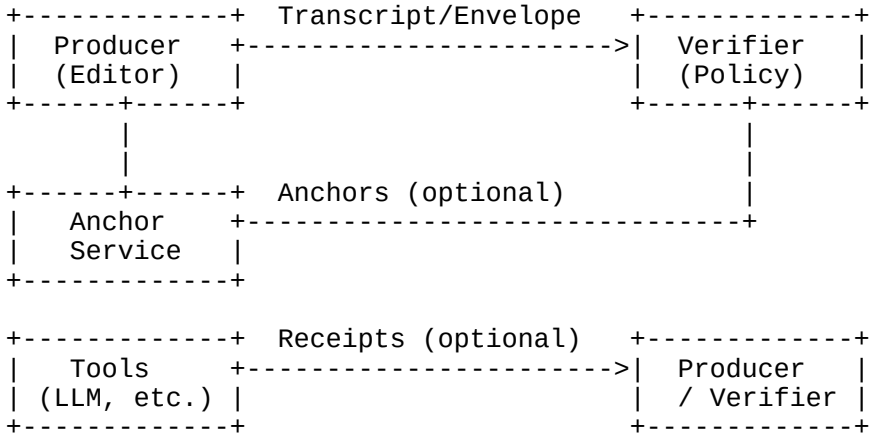

Figure 1: PoP Evidence Flow (Informative)

PoP is compatible with RATS/EAT deployments by treating the PoP envelope as evidence that can be conveyed alongside attestation tokens, referenced by claims, or embedded in artifact metadata for later validation. The overall Proof of Process architecture is defined in [I-D.condrey-rats-pop].

5. Data Model

PoP uses CBOR [RFC8949] as the encoding and CDDL [RFC8610] for data model definitions. All hashing computations depend on deterministic CBOR encoding; Producers and Verifiers MUST use a deterministic CBOR encoding profile suitable for reproducible hashing.

5.1. Ranges and Identifiers

Offsets and ranges refer to a canonical byte or character indexing scheme over the logical artifact content. The indexing scheme MUST be specified by the Producer and bound in the envelope so that Verifiers can interpret ranges consistently.

This specification defines a generic Range structure. Producers MAY use byte offsets, Unicode scalar offsets, or another deterministic scheme. The chosen scheme MUST be indicated in the envelope.

5.2. Canonical Event Types

PoP defines canonical semantic event types. Events describe changes to artifact state at a semantic operation level rather than raw keystrokes. Raw keystroke telemetry is OPTIONAL and is not required for PoP verification.

The following event types are defined:

* *INS*: insert content at a range/position
* *DEL*: delete content in a range
* *REP*: replace content in a range with new content
* *MOV*: move a range to a destination position
* *PASTE*: insert content with an origin reference (e.g., receipt-bound)
* *SNAP*: checkpoint snapshot of artifact hash and optional cursor state

Events MUST include sufficient information to validate transcript integrity and, when required by policy, to validate that the transcript corresponds to the final artifact (e.g., via checkpoints).

6. Event Hash Chaining

PoP binds transcript events in order using a hash chain. The chain depends on deterministic encoding of each Event structure.

6.1. Hash Algorithm

Let H be a cryptographic hash function. This specification RECOMMENDS SHA-256. The initial hash value h_0 MUST be defined as the hash of a domain separation string and the session identifier:

h_0 = H("PoP-Transcript" || session_id)

For each event E_i, the chain value is computed as:

h_i = H(h_{i-1} || CBOR_Deterministic(E_i))

The transcript root is defined as root = h_n for the final event E_n.

6.2. Requirements

* Producers MUST use deterministic CBOR encoding for all hashed structures.
* Verifiers MUST recompute the hash chain and compare the derived root to the envelope root.
* Mismatch of any chain computation MUST cause verification failure.

7. Time Anchoring

PoP supports optional chronology binding via Anchors. Policies MAY require specific anchor types or densities. This document defines two anchor classes: External Anchors and Sequential Work Anchors.

7.1. External Anchors

An External Anchor binds the transcript root (or an intermediate chain value) to an externally verifiable time assertion (e.g., a timestamping service, transparency log, or other notary).

The specific external anchoring system is out of scope for this document; PoP defines a generic container for including and validating anchor evidence.

7.2. Sequential Work Anchors

A Sequential Work Anchor provides evidence that a specified amount of sequential computation was performed between two transcript states. This can increase the cost of post-hoc backfilling. The concrete VDF or sequential-work scheme is policy-defined. This document defines a container to carry such evidence.

7.3. Anchor Verification Rules

* If an envelope includes anchors, verifiers MUST validate each anchor according to its type.

* If a policy requires anchors and they are absent or invalid, verification MUST fail.

* Anchor validation MUST bind to the relevant transcript state (root or intermediate chain values) as specified by the anchor.

8. Tool Receipts and Provenance DAG

PoP supports compositional provenance via signed tool receipts. Receipts enable producers to explicitly declare and bind tool-generated outputs (including AI assistance) without relying on brittle inference or detection. Receipts MAY be used for any deterministic or non-deterministic tool that emits content, patches, or transformations that contribute to the artifact.

8.1. Receipt Model

A Receipt is a signed statement that binds tool identity and an output commitment. An input commitment MAY be included to allow stronger binding, but is OPTIONAL to support privacy-preserving tools.

Receipts MAY reference other receipts to form a provenance DAG. Policies MAY restrict recursion depth.

8.2. Receipt Verification Rules

* Verifiers MUST validate each receipt signature over its canonical encoding.

* Receipt public keys MUST be represented in a verifiable format; policies MAY require certificate chains or allow bare keys.

* If an event references a receipt (via source_ref), the referenced receipt MUST be present or retrievable under policy.

* Policies MAY require validation of the provenance DAG and MAY impose recursion and size limits.

9. Envelope Structure

The PoP Envelope is the primary interchange object. It binds transcript commitments, anchors, receipts, and policy hooks. The envelope is CBOR-encoded and SHOULD be embedded into artifact metadata or conveyed alongside artifacts.

9.1. Policy and Seed Commitments

PoP defines policy hooks but does not standardize a scoring model. To enable reproducible application of a policy, the envelope includes:

* *policy_commit*: a commitment to a policy descriptor

* *seed_commit*: a commitment to a seed used to parameterize evaluation and/or audits

Commitments support commit-then-reveal workflows where the evaluation surface is unpredictable at capture time but verifiable after reveal under policy.

9.2. Claims Field

The envelope MAY include a claims field carrying derived values (e.g., integrity pass/fail, evaluation outputs). Claims MUST NOT be treated as authoritative unless verifiers recompute or validate them under policy. Policies SHOULD treat claims as advisory and derive evaluation from transcript evidence.

9.3. Optional Zero-Knowledge Bundle

The envelope MAY include a zk bundle to support privacy-preserving verification (e.g., threshold proofs for policy-defined evaluations without full transcript disclosure). The specific ZK proof system is policy-defined.

10. Verification Procedure

Verifiers MUST perform the following procedure to validate a PoP envelope. Policies MAY impose additional requirements and MAY allow partial disclosure modes.

10.1. Structural Validation

1. Parse the envelope as CBOR and validate required fields.

2. Validate version and supported algorithms.

3. Validate encoding determinism constraints where applicable.

10.2. Hash Chain Validation

1. Obtain the transcript (full disclosure or policy-authorized proofs).

2. Recompute the hash chain from h_0 through h_n.

3. Compare derived root to the envelope root.

Mismatch MUST cause verification failure.

10.3. Anchor Validation

1. Validate all included anchors.

2. Confirm anchors bind to the intended transcript state (root or specified chain values).

3. Apply policy requirements for anchor presence, density, or types.

10.4. Receipt Validation

1. Validate receipt signatures and canonical encodings.

2. Resolve and validate referenced receipts as required by policy.

3. Verify any policy constraints on receipt DAG depth or classes.

10.5. Artifact Binding Validation

Policies MAY require validation that the transcript corresponds to the final artifact. This may be performed by verifying SNAP events, checkpoints, or other commitments that bind the transcript to the final artifact hash.

10.6. Policy Evaluation

After transcript and anchoring verification, a verifier MAY compute features and apply a policy-defined evaluation. If the envelope includes a ZK bundle and the policy allows it, the verifier MAY validate a ZK proof instead of obtaining the full transcript.

11. Policy-Defined Evaluation

PoP does not standardize any scoring function, threshold, or interpretation. Policies define evaluation inputs, models, and thresholds. PoP provides hooks to identify policies and support reproducible evaluation.

11.1. Policy Commitments

The policy_commit field is a cryptographic commitment to a policy descriptor, e.g.:

policy_commit = H(policy_descriptor)

The contents and distribution of the policy descriptor are out of scope for this document. Policies MUST be uniquely identifiable by policy_commit.

11.2. Seeded Evaluation (Optional)

Policies MAY use seeds to parameterize evaluation functions (e.g., to mitigate adaptive optimization against a static evaluation surface). If used, the envelope includes seed_commit, and verifiers MAY require a reveal of the seed or a ZK proof that incorporates the committed seed.

11.3. Evaluation Outputs

Any evaluation outputs, including derived claims, are policy-defined and MUST be expressed as assessments of evidence, not assertions of cognitive origin. Policies SHOULD provide outputs that are interpretable and auditable (e.g., quantified import ratios, refinement ratios, or other measurable evidence summaries).

12. Security Considerations

PoP provides tamper-evident transcripts and optional chronology binding. It does not prevent an adversarial producer from performing arbitrary sequences of valid edits. Policies should treat PoP as evidence of recorded process, not proof of mental origin.

12.1. Threat Model Summary

* *Synthetic transcript generation*: a producer may attempt to fabricate a transcript consistent with a target artifact.

* *Receipt forgery*: a producer may attempt to forge or replay tool receipts.

* *Anchor replay or substitution*: invalid anchors may be presented or misbound.

* *Adaptive optimization*: if evaluation metrics are static and public, adversaries may optimize transcripts to pass thresholds.

* *Privacy leakage*: transcripts can reveal sensitive drafting history if disclosed improperly.

12.2. Mitigations

* Hash chaining over deterministic encodings provides tamper evidence.

* Anchors (timestamps, sequential work) can raise the cost of post-hoc fabrication and bind chronology.

* Receipt signatures and, when used, DAG validation mitigate provenance forgery.

* Seeded evaluation can mitigate adaptive optimization (policy-defined; not required by PoP).

* Policies should define disclosure requirements and support partial disclosure or ZK proofs.

12.3. Algorithm Agility

Producers and verifiers SHOULD support algorithm agility for hash functions and signature schemes. Policies SHOULD specify allowed algorithms and minimum security strengths.

13. Privacy Considerations

Transcripts can reveal sensitive intermediate drafts, editing patterns, and content that was later removed. PoP supports multiple disclosure modes, but this document does not mandate a particular privacy policy.

* *Full disclosure*: the entire transcript is revealed to a verifier.
* *Partial disclosure*: Merkle proofs or selective event disclosure under policy.
* *Zero-knowledge*: policy-defined threshold proofs without transcript disclosure.

Policies SHOULD minimize disclosure and SHOULD avoid collecting raw keystrokes or invasive telemetry unless explicitly required for a deployment and accompanied by informed consent.

14. IANA Considerations

This document requests IANA to register a media type for PoP envelopes encoded in CBOR.

14.1. Media Type Registration

Type name: application

Subtype name: pop+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: binary

Security considerations: see Section 12

Interoperability considerations: none

Published specification: this document

Applications that use this media type: editors, provenance systems, attestation verifiers

Fragment identifier considerations: none

Additional information:

Magic number(s): none

File extension(s): .pop

Macintosh file type code(s): none

Person & email address to contact for further information: David Condrey (david@writerslogic.com)

Intended usage: COMMON

Restrictions on usage: none

Author: Authors of this document

Change controller: IETF

15. CDDL Summary (Normative)

This appendix provides a consolidated CDDL summary for PoP-1.5 structures. Implementations MUST follow these definitions and MUST apply deterministic CBOR encoding for all hashed objects. The complete normative CDDL schema is defined in [I-D.condrey-rats-pop-schema].

```
; NOTE: This CDDL is a summary skeleton. Field-level constraints and algorithm
; registries should be expanded as the draft matures.

PoPEnvelope = {
  version: "PoP-1.5",
  session_id: bstr,
  root: bstr,
  transcript_commit: bstr,
  ? index_scheme: tstr,          ; e.g., "byte", "unicode_scalar"
  anchors: [* Anchor],

  seed_commit: bstr,
  policy_commit: bstr,

  receipts: [* Receipt],
  import_index: ImportSummary,
```

```
  ? claims: Claims,
  ? zk: ZKBundle
}

Range = {
  start: uint,
  len: uint
}

Event = {
  type: tstr,                    ; "INS" / "DEL" / "REP" / "MOV" / "PASTE" / "SNAP"
  ? range: Range,
  ? dst_pos: uint,
  ? data_hash: bstr,
  ? old_hash: bstr,
  ? new_hash: bstr,
  ? length: uint,
  ? doc_hash: bstr,              ; for SNAP
  ? cursor_pos: uint,            ; optional
  ? source_ref: bstr,            ; references Receipt.output_commit or receipt id
  ? aux: bstr
}

Anchor = {
  type: tstr,                    ; "external" / "seqwork" / policy-defined
  binds: bstr,                   ; root or intermediate chain value
  value: bstr,
  ? timestamp: uint
}

Receipt = {
  tool_id: tstr,
  tool_pubkey: bstr,
  input_commit: bstr / null,
  output_commit: bstr,
  flags: [* tstr],
  ? anchor: Anchor,
  sig: bstr
}

ImportSummary = {
  raw_import_bytes: uint,
  paste_events: uint,
  receipt_bound_events: uint,
  refined_pct: float
}

Claims = {
```

```
  integrity: float,
  time: float,
  liveness: float,
  emergence: float,
  topology: float,
  assistance: float,
  refinement: float,
  process_quality: float
}

ZKBundle = {
  score_threshold_proof: bstr,
  public_inputs: [* bstr]
}
```

Figure 2: PoP-1.5 CDDL Summary

16. Example Verification Flow (Informative)

This appendix illustrates a typical verification sequence.

1. Verifier parses the PoP envelope and checks required fields.
2. Verifier obtains transcript disclosure (full transcript or policy-approved proofs).
3. Verifier recomputes the transcript hash chain and validates the root.
4. Verifier validates all anchors included in the envelope and applies policy requirements.
5. Verifier validates receipt signatures and resolves any referenced receipts per policy.
6. Verifier validates binding to the final artifact hash (if required by policy).
7. Verifier applies policy-defined evaluation; if ZK is provided, verifies threshold proof.
8. Verifier outputs an evidence summary as policy-defined assessment of process evidence.

17. References

17.1. Normative References

[IANA.cbor-tags]
IANA, "CBOR Tags",
<https://www.iana.org/assignments/cbor-tags>.

[IANA.media-types]
IANA, "Media Types",
<https://www.iana.org/assignments/media-types>.

[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>.

[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)", 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>.

[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>.

17.2. Informative References

[I-D.condrey-rats-pop]
Condrey, D., "Proof of Process Provenance Protocol (PPPP): An Evidence Framework for Document Authorship Attestation", Work in Progress, Internet-Draft, draft-condrey-rats-pop, <https://datatracker.ietf.org/doc/html/draft-condrey-rats-pop>.

[I-D.condrey-rats-pop-examples]
Condrey, D., "Examples of Proof of Process Evidence Packets and Attestation Results", Work in Progress, Internet-Draft, draft-condrey-rats-pop-examples, <https://datatracker.ietf.org/doc/html/draft-condrey-rats-pop-examples>.

[I-D.condrey-rats-pop-schema]
Condrey, D., "Proof of Process CDDL Schema", Work in Progress, Internet-Draft, draft-condrey-rats-pop-schema, <https://datatracker.ietf.org/doc/html/draft-condrey-rats-pop-schema>.

[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>.

[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>.

Acknowledgments

The author thanks the RATS working group for foundational work on remote attestation architectures. Thanks also to reviewers who provided feedback on CBOR encoding determinism, COSE signing surfaces, and transport binding interoperability.

Author’s Address

David Condrey
Writerslogic Inc
United States

Email: david@writerslogic.com
URI: https://writerslogic.com