RATS
Internet-Draft
Intended status: Standards Track
Expires: 7 October 2026

A. Damodaran
Sovereign AI Stack

5 April 2026

The Prove-Transform-Verify (PTV) Protocol for Attested Agent Identity
draft-anandakrishnan-rats-ptv-agent-identity-00

Abstract

This document describes the Prove-Transform-Verify (PTV) protocol for hardware-anchored attestation of AI agent identity. PTV is designed to enable an agent to prove that it is running an authorized model and policy without exposing sensitive data. The protocol is intended to compose with existing RATS attestation mechanisms and to support exercise-time re-attestation requirements.

The protocol defines a common envelope format (CBOR/CDDL), message types for attestation request/response, and a threat model that separates identity binding integrity from behavioral continuity. Behavioral continuity is treated as a separate requirement class addressed via exercise-time checks and execution receipts (e.g., SCITT composition), not by PTV alone.

Example use cases include healthcare clinical decision support systems (CDSS), critical infrastructure industrial control systems (ICS/OT), and cross-border regulatory compliance scenarios.

Status of This Memo

Table of Contents

1. Introduction

Current identity frameworks (OAuth 2.0, SPIFFE) establish workload identity but do not verify what model an agent is running or whether it follows authorized policies. This gap creates systemic risk: a compromised agent can generate unauthorized outputs while still presenting valid credentials.

For example, in a clinical decision support system, a hospital may need cryptographic evidence that a diagnosis was produced by an approved model and policy set, without exposing patient records or model internals. Similarly, in an industrial control system, an operator may need cryptographic evidence that a safety-critical control action was generated by an authorized agent configuration, without exporting plant telemetry or proprietary control logic.

The PTV protocol addresses this gap by binding agent identity to hardware roots of trust (TPM 2.0 or Secure Enclave) and using zero-knowledge proofs to attest to model integrity without exposing model weights or inference data.

1.1. Terminology and Definitions

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.

Agent	An autonomous software entity that performs tasks on behalf of a user or system.
Attester	The agent component responsible for generating cryptographic attestations about its state or actions.
Verifier	A system that validates cryptographic attestations received from Attesters without accessing underlying sensitive data.
Relying Party	A system that consumes attestation results to make authorization decisions.
PTV Edge Proxy	A trusted gateway that performs attestation on behalf of constrained devices lacking hardware roots of trust.

Nonce	A cryptographically random value used to prevent replay attacks.
Triage Band	A risk classification (ROUTINE, ZK_PROOF, BFT) that determines the attestation mechanism. Valid values are 0 (ROUTINE), 1 (ZK_PROOF), or 2 (BFT).
Governance Pack	A set of policies and compliance rules applied during attestation.
Proof	A zero-knowledge proof attesting to agent state.
Sovereign Bound Metadata	Jurisdiction and data residency metadata embedded in attestation chains that cannot be removed without invalidating the attestation. The corresponding envelope field is sovereign_bound, which contains the subfields jurisdiction, data_residency, and compliance.
Hardware Root of Trust	A tamper-resistant chip (TPM 2.0 or Secure Enclave) providing cryptographic evidence that software has not been altered since last attestation.

2. Threat Model

2.1. Identity Binding Integrity

Identity binding integrity asks: "Is this the agent that was enrolled, and is the signer of this attestation trusted to hold the corresponding key?"

PTV addresses this class of threats by:

* binding attestation evidence to a hardware root of trust (e.g., TPM AK),
* using nonce-based freshness to prevent replay, and
* allowing policy-aware verification of configuration state.

This corresponds to cryptographic identity continuity in the sense of the RATS Architecture [RFC9334]. A relying party that consumes a PTV attestation MAY treat it as an assertion about identity binding integrity only, unless additional mechanisms are in place.

2.2. Behavioral Continuity (Out of Scope for -01)

Behavioral continuity asks: "Is the enrolled agent still behaving within its attested envelope after context changes?"

This version of PTV does not fully address this requirement class. In particular, PTV does not detect or mitigate:

* behavioral drift after context compression or summarization,
* model weight replacement or fine-tuning after enrollment, and
* runtime drift from adversarial prompt injection or accumulated state corruption.

A valid PTV attestation on a behaviorally-drifted agent is therefore a false signal of behavioral identity continuity. For high-stakes use cases such as clinical decision support systems (CDSS) and ICS/OT, relying parties SHOULD treat behavioral continuity as a separate requirement. It can be partially addressed by combining PTV freshness with delegation provenance (e.g., HDP) and execution outcome verification (e.g., SCITT receipts). See Section 6.

2.3. Exercise-time Freshness Requirement

For high-stakes deployments, relying parties SHOULD be able to demand fresh attestation at exercise time (per authorization decision), via an EAT nonce challenge or a PTV challenge-response, rather than relying solely on a credential issued at enrollment time.

3. PTV Model and Roles

3.1. Protocol Overview

The PTV protocol operates through three phases:

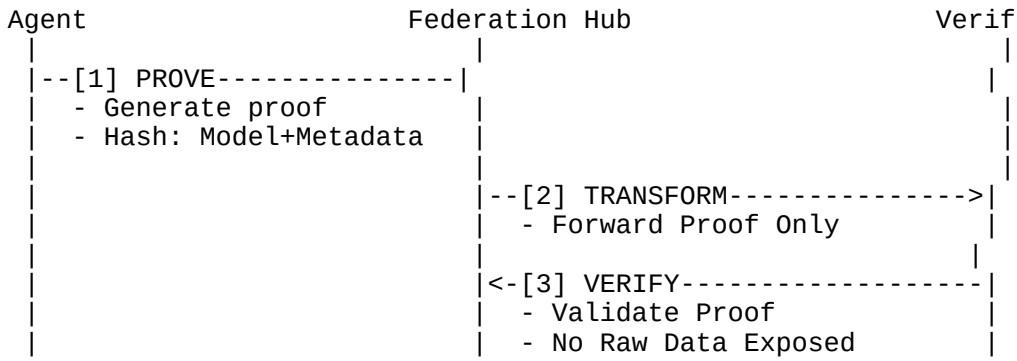

Figure 1: PTV Protocol Sequence Diagram

The three phases are as follows:

* *PROVE:* The agent generates a cryptographic proof locally that a task was executed correctly on an authorized model.
* *TRANSFORM:* Only the proof (not raw data) is transmitted to the central system.
* *VERIFY:* The system validates the proof without accessing sensitive inputs.

3.2. Roles

The PTV protocol defines four logical roles:

* *Attester:* The agent generating attestations. May be a direct TPM-bound agent or a constrained device using an Edge Proxy.
* *Verifier:* The system validating attestations.
* *Relying Party:* The system consuming attestation results for authorization decisions.
* *Edge Proxy:* An optional gateway performing attestation on behalf of constrained devices.

3.3. Deployment Models

PTV supports two deployment models:

* *Direct Mode:* The agent has a TPM 2.0 or TEE and generates proofs natively.
* *Proxy-Assisted Mode:* The agent is a constrained device (e.g., legacy PLC, sensor) without a TPM. The PTV Edge Proxy performs attestation on its behalf.

4. Message Formats

4.1. CBOR/CDDL Envelope

All PTV messages are encapsulated in a common envelope structure, serialized using CBOR [RFC8610].

The following CDDL rules define the envelope:

```
triage-band = 0..2

ptv-envelope = {
  ptv_version: tstr,
  msg_type: ptv-msg-type,
  nonce: bstr,
  ? attester_id: tstr,
  ? sovereign_bound: {
    jurisdiction: tstr,
    data_residency: tstr,
    compliance: [* tstr]
  },
  ? triage_band: triage-band,
  ? evidence: bstr,
  ? proof: bstr,
  ? policy: ptv-policy,
  ? result: ptv-result,
  ? behavior_fingerprint: bstr,
  ? timestamp: time,
  ? extensions: { * tstr => any }
}

ptv-msg-type = &(
  prove-req: 1,
  prove-resp: 2,
  transform-att: 3,
  verify-result: 4
)

ptv-policy = {
  ? audience: tstr,
  ? policy_uri: tstr,
  ? trust_domain: tstr,
  ? requirements: [* tstr]
}

ptv-result = &(
  success: 0,
  failure: 1,
  indeterminate: 2
)
```

The behavior_fingerprint field is OPTIONAL in this version. It is intended for deployments that adopt a SCITT-style execution receipt pattern. Its contents are opaque bytes with deployment-specific semantics; this version does not define normative processing rules. See Section 6.3 for discussion of related future work.

In PTV_PROVE_RESP and PTV_TRANSFORM_ATT, exactly one of evidence or proof MUST be present. A message that contains neither or both MUST be rejected by the verifier.

4.2. Message Types

4.2.1. PTV_PROVE_REQ (Message Type 1)

Sent by the relying party to request attestation. Contains a nonce and optional policy constraints.

```
PTV_PROVE_REQ = {
  ptv_version: "1.0",
  msg_type: 1,
  nonce: bstr,
  ? policy: ptv-policy
}
```

4.2.2. PTV_PROVE_RESP (Message Type 2)

Sent by the attester in response to a prove request. Contains either evidence (for ROUTINE triage) or a proof (for ZK_PROOF triage).

```
PTV_PROVE_RESP = {
  ptv_version: "1.0",
  msg_type: 2,
  nonce: bstr,
  attester_id: tstr,
  sovereign_bound: {
    jurisdiction: tstr,
    data_residency: tstr,
    compliance: [* tstr]
  },
  triage_band: triage-band,
  (evidence: bstr // ROUTINE triage
   // OR
   proof: bstr),   // ZK_PROOF triage
  ? behavior_fingerprint: bstr,
  timestamp: time
}
```

4.2.3. PTV_TRANSFORM_ATT (Message Type 3)

Sent by the Edge Proxy to the Verifier.

```
PTV_TRANSFORM_ATT = {
  ptv_version: "1.0",
  msg_type: 3,
  nonce: bstr,
  attester_id: tstr,
  sovereign_bound: {
    jurisdiction: tstr,
    data_residency: tstr,
    compliance: [* tstr]
  },
  triage_band: triage-band,
  (evidence: bstr // ROUTINE triage
   // OR
   proof: bstr),   // ZK_PROOF triage
  timestamp: time
}
```

4.2.4. PTV_VERIFY_RESULT (Message Type 4)

Sent by the verifier to the relying party.

```
PTV_VERIFY_RESULT = {
  ptv_version: "1.0",
  msg_type: 4,
  nonce: bstr,
  result: ptv-result,
  ? reason: tstr,
  ? audit_id: tstr,
  timestamp: time
}
```

4.3. Error Code Registry

The following error codes are defined for PTV attestation failures:

Code	Description	Recovery Action
PTV_ERR_001	TPM attestation failure	Verify TPM 2.0 presence and permissions
PTV_ERR_002	Proof verification failed	Regenerate proof with correct inputs
PTV_ERR_003	Jurisdiction mismatch	Check sovereign_bound against policy
PTV_ERR_004	Quorum not reached	Retry with additional consensus nodes
PTV_ERR_005	Envelope expired	Generate fresh attestation
PTV_ERR_006	Nonce replay detected	Discard and request a new nonce generated according to the cryptographic randomness guidance in RFC 4086 [RFC4086].
PTV_ERR_007	Model hash not approved	Update approved model list

Table 1: PTV Error Codes

4.4. Performance Characteristics (Informative)

Note: The following figures are preliminary observations from a prototype implementation. They are provided for illustration only and are not normative requirements of the protocol.

Metric	Observed Value	Conditions
Proof generation	~187ms	Prototype, n=10,000 (illustrative)
Proof verification	<5ms	Single proof (prototype)
Raw proof size	<300 bytes	Groth16 (BN254) in prototype

Table 2: Prototype Performance Observations

5. Security Considerations

This section discusses security and privacy considerations for the PTV protocol.

5.1. Freshness and Replay

All PTV attestations MUST include a nonce provided by the relying party or verifier. The nonce MUST be cryptographically random (see [RFC4086]) and MUST NOT be reused across attestation sessions. Timestamps SHOULD be included and checked against a deployment-specific freshness window (RECOMMENDED: less than or equal to 300 seconds).

5.2. Zero-Knowledge Proof Mechanism

PTV is designed to be compatible with zero-knowledge proof systems but does not mandate a specific construction.

The predicate being proven MUST be specified, for example: "the hash of the agent configuration matches an enrolled value".

The zero-egress property of ZK-proofs aligns with Privacy by Design principles. Deployers SHOULD assess compliance with applicable regulations; this protocol is designed to facilitate such assessments but does not constitute legal certification.

5.3. Identity Misbinding

Attestations SHOULD be bound to the specific agent instance using a hardware root of trust (TPM 2.0 or Secure Enclave). Software-only attestations are NOT RECOMMENDED for high-assurance deployments.

5.4. Denial of Service

Verifiers SHOULD implement rate limiting and proof caching to mitigate DoS attacks.

6. Interoperability and Future Work

6.1. EAT

When Entity Attestation Tokens (EAT, [RFC9711]) are present, PTV evidence and proofs SHOULD map to EAT claims such as eat_nonce, swname, and swversion. PTV evidence/proofs are also intended to be usable within SEAT/ExPAT-style exported attestation mechanisms. A future revision of this document may define a concrete mapping.

6.2. Layering and Composition

PTV is designed to compose with complementary protocols that address different parts of the agentic trust chain. Specifically:

* PTV provides attested agent identity and model/policy binding at exercise time (the "who and what is running" layer),
* The Human Delegation Provenance Protocol (HDP) [HDP] establishes signed delegation provenance anchored to a human principal (the "who authorized what" layer), and
* SCITT-based execution receipts or Execution Outcome Verification (EOV) provide evidence of what the agent actually produced (the "what was done" layer).

A natural composition point is for an EOV receipt to reference a hash of an HDP delegation record as its delegation token, rather than duplicating the full provenance chain. Future revisions of this document may describe concrete mappings once the HDP and EOV models stabilize.

6.3. SCITT and Execution Receipts

PTV is intended to compose with SCITT as follows:

* PTV/SEAT provide attested agent identity and state at exercise time, and
* SCITT COSE_Sign1 statements carry execution receipts for actions with external effects.

Three open standardization gaps remain, treated as future work:

1. A registered COSE header label for the behavioral fingerprint.
2. A standardized format for behavior probes / fingerprints.

3. A live transparency service for cross-domain execution logs.

6.4. Execution Outcome Verification

Execution outcome verification is treated as a separate concern from attestation. This version assumes that execution receipts, where needed, are carried by SCITT or a similar transparency mechanism.

Delegation provenance protocols (for example HDP [HDP]) and execution-outcome attestation patterns are complementary to PTV and out of scope for this version.

7. IANA Considerations

This document makes no requests of IANA. Future revisions may request registration of media types, OAuth parameters, or attestation method registries once the protocol design has stabilized.

8. Normative References

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

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", June 2005, <https://www.rfc-editor.org/info/rfc4086>.

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

[RFC8610] Bormann, C. and H. Birkholz, "CBOR Data Definition Language (CDDL)", June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "Entity Attestation Token (EAT)", December 2024, <https://www.rfc-editor.org/info/rfc9711>.

9. Informative References

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

[HDP] Dalugoda, S., "Human Delegation Provenance Protocol (HDP): Cryptographic Chain-of-Custody for Agentic AI Systems", March 2026, <https://datatracker.ietf.org/doc/draft-helixar-hdp-agentic-delegation-00/>.

Author's Address

Anandakrishnan Damodaran
Sovereign AI Stack
Email: ananda.krishnan@hotmail.com
URI: https://github.com/anandkrshnn