Position Paper

From Declaration to Proof

Cryptographic Governance Evidence
for Autonomous AI Agents

A T T E S T E D I N T E L L I G E N C E

Jack Brennan
admin@attestedintelligence.com

March 28, 2026

AttestedIntelligence.com

TABLE OF CONTENTS

Abstract
1. The Declaration-Enforcement Gap
2. Why Logging Is Not Proof
3. Three Properties of Governance Evidence
4. Architecture: Seal, Enforce, Prove
5. Standards Alignment
6. The MCP Governance Gap
7. Case Study: When Configuration Replaces Enforcement
8. Implementation Path
Conclusion

A B S T R A C T

Every major AI governance framework declares what autonomous agents should do. None prove what they actually did. This paper identifies the structural gap between policy declaration and runtime enforcement evidence, explains why conventional logging and monitoring cannot close it, and describes a concrete architecture - Seal, Enforce, Prove - that produces cryptographic proof of governance enforcement verifiable by any third party on an air-gapped machine. We map this architecture to five active standards efforts (MSSS, CoSAI WS4, OWASP MCP Top 10, CSA/CSAI, and Sigstore/OpenSSF) and demonstrate its application to the Model Context Protocol, where the gap between declared policy and provable enforcement is most acute. Declaration without enforcement is aspiration. Enforcement without evidence is a trust relationship. The standards that govern autonomous AI agents need all three.

1. The Declaration-Enforcement Gap

Governance frameworks for autonomous AI agents share a common structure. They define roles, specify authorized behaviors, establish access boundaries, and prescribe monitoring requirements. NIST AI RMF prescribes Govern, Map, Measure, and Manage functions. The EU AI Act mandates quality management systems (Article 17), risk assessments (Article 9), and automatic recording of events (Article 12) - with enforcement beginning August 2026 for high-risk AI systems, requiring organizations to demonstrate not just that governance policies exist but that they were applied. OWASP's Top 10 for Agentic Applications catalogs risks and recommends mitigations. The MCP Server Security Standard defines four compliance levels with 24 controls spanning authentication, authorization, input validation, and logging.

Each of these frameworks answers the question: *what should an AI agent be allowed to do?*

None of them answer the question that follows: *can anyone prove, after the fact, that the agent did only what it was allowed to do?*

This is the declaration-enforcement gap. A governance declaration states intent. A governance enforcement mechanism constrains behavior. Governance evidence proves the constraint was applied. The gap exists because the industry has invested heavily in the first two categories and almost nothing in the third.

The consequences are documented. When Palo Alto Networks' Unit 42 analyzed a leading autonomous agent framework against the OWASP Top 10 for Agentic Applications, every category exhibited significant governance gaps - not because policies were absent, but because no enforcement boundary existed between the agent's decisions and the systems it controlled. When SecurityScorecard's STRIKE team surveyed MCP server deployments, they identified widespread exposure across dozens of countries with the common finding that governance controls were declared but not enforced at runtime. When an autonomous trading agent destroyed over $400,000 in a single transaction due to a parsing error after a session crash, the failure was not a missing policy. It was a missing enforcement boundary that would have constrained the agent's transaction authority regardless of its internal state.

The gap extends beyond software. As physical AI enters commercial deployment - Boston Dynamics Atlas integrating Gemini, Agile Robots deploying thousands of units with DeepMind integration, warehouse and manufacturing robots operating with increasing autonomy - the governance evidence problem compounds. A software agent that exceeds its authority costs money. A physical agent that exceeds its authority costs safety.

Declaration without enforcement is aspiration. Enforcement without evidence is a trust relationship. The industry needs all three.

2. Why Logging Is Not Proof

The default response to governance evidence requirements is logging - every framework recommends it. The EU AI Act Article 12 requires "automatic recording of events" over the lifetime of the system. NIST AI RMF's Measure function implies continuous data collection. The OWASP MCP Top 10 identifies insufficient audit and telemetry as a critical risk. The instinct is reasonable: if every event is captured, compliance can be demonstrated by reviewing what happened.

This instinct is wrong for three reasons that are architectural, not operational.

Logs are mutable. A log file is data stored on a filesystem. Anyone with sufficient system access can alter it, delete entries, or selectively omit events. Database-backed logs offer no structural advantage: an administrator with write access can modify records without leaving a trace in the log itself. In incident response, the first question a forensic examiner asks is whether the logs can be trusted. The answer, absent cryptographic controls, is always conditional. When the system under investigation controls its own audit trail, that condition is rarely satisfied.

Logs are passive. A log records what happened. It does not prevent what should not happen. A log entry stating "tool invocation denied" tells you the system reported a denial. It does not prove that a mandatory enforcement boundary existed, that the boundary was active at the time of the request, or that the denial could not have been bypassed. Passive recording and active enforcement are fundamentally different capabilities. Conflating them is the root of the problem.

Logs are producer-controlled. The system generating the log controls what goes into it. If an AI agent has write access to its own log infrastructure, it determines the completeness and accuracy of the record. A compromised or misaligned agent can report compliant behavior while acting otherwise. Producer-controlled evidence is not independently verifiable evidence. It is a self-assessment.

These three properties are not bugs in logging implementations. They are inherent characteristics of the logging model. No amount of log aggregation, SIEM integration, or retention policy changes the fundamental structure: a log is a claim made by the system about itself. It is not proof.

3. Three Properties of Governance Evidence

Proof, in the governance context, requires three properties that logging cannot provide.

Sealed. Before an agent begins operating, its authorized scope must be cryptographically committed into an immutable artifact. Which tools it can invoke, which operations are permitted, what rate limits apply, what temporal bounds constrain its authority - all bound by an Ed25519 digital signature over the SHA-256 hash of the JCS-canonicalized (RFC 8785) policy document. Modification of any field invalidates the signature. The constraints are no longer advisory or configurable at runtime. They are cryptographic commitments that the governed agent cannot alter because it does not hold the signing key.

Signed. Every enforcement decision - whether an action was permitted, denied, or flagged - must produce a cryptographic receipt. The receipt contains the action, the policy artifact hash, the decision, the timestamp, a monotonic sequence number, and a SHA-256 hash linking it to the previous receipt's structural metadata. The receipt chain is append-only. Modification of any receipt invalidates every subsequent link. The governed subject cannot forge receipts because it operates in a separate execution context with no access to the Ed25519 signing key held by the enforcement boundary. Gaps in sequence numbers indicate receipt suppression. Timestamp inversions indicate clock manipulation. Both are mathematically detectable.

Portable. At the end of an operation, an evidence bundle containing the sealed policy artifact, all signed receipts, and Merkle inclusion proofs must enable any third party to verify the complete governance history. No network access required. No trust in the vendor. No trust in the operator. No trust in the platform. Standard cryptographic verification using Ed25519 signature checking and SHA-256 hash computation on an air-gapped machine. An independent verifier using zero framework imports - standard cryptographic libraries only - confirms that verification is not implementation-dependent.

These three properties transform the governance question from "who do we trust?" to "what can we verify?"

4. Architecture: Seal, Enforce, Prove

The architecture implementing these properties uses standard cryptographic primitives: Ed25519 for digital signatures (RFC 8032), SHA-256 (FIPS 180-4) and BLAKE2b-256 (RFC 7693) for hashing, RFC 8785 for JSON canonicalization, and Merkle trees (RFC 6962) for batch integrity proofs. No trusted execution environments required. No zero-knowledge proof circuits. The architecture runs on commodity hardware.

Phase 1: Seal

Before an agent operates, its complete governance scope is attested and locked into an immutable Policy Artifact (Claim 1(a)). The artifact encodes agent identity through the Subject Identifier (Claim 1(b)), comprising cryptographic hashes of the agent's normalized bytes and canonicalized metadata. It specifies authorized behavior: permitted tools, denied tools, rate limits, required parameters, and forbidden sequences. It defines enforcement parameters: measurement cadence, enforcement actions upon violation, and time-to-live (Claim 6). The complete artifact is signed with Ed25519 over its SHA-256 hash. Any modification invalidates the signature.

Phase 2: Enforce

A separate process - the Portal - sits between the agent and everything it interacts with (Claim 1(e)). The Portal operates as a Policy Enforcement Point in NIST SP 800-207 (Zero Trust Architecture) terms. The agent holds no cryptographic keys, no OAuth tokens, no API credentials. It can only reach MCP servers, APIs, and external resources through the Portal.

Before authorizing any operation, the Portal reads the agent's current binary, computes its SHA-256 hash, and compares it to the sealed reference (Claim 1(f)). It evaluates the requested operation against the sealed behavioral constraints. It makes the enforcement decision: PERMIT, DENY, or DRIFT. It generates a signed receipt and appends it to the continuity chain. This check happens on every request.

> When behavioral drift is detected - an agent invoking tools in anomalous patterns suggesting prompt injection or context poisoning - the Portal can transition the agent to phantom execution (Claim 11). All connections to protected resources are severed. The agent continues executing in a sandboxed environment, receiving no signal that it has been transitioned. Every unauthorized action is captured as signed forensic receipts. This transforms a compromise into an intelligence-gathering opportunity without exposing production systems.

The two-process boundary is the critical architectural property. The governed subject executes. The Portal enforces and records. That boundary cannot be collapsed by the vendor, the operator, or the model itself.

Phase 3: Prove

Every enforcement decision produces a signed Enforcement Receipt appended to a Continuity Chain (Claim 3). Each receipt contains the event type (PERMIT, DENY, DRIFT, MEASUREMENT), monotonic sequence number, timestamp, the action evaluated, the policy artifact hash that governed the decision, and a SHA-256 hash of the previous receipt's structural metadata for chain linking.

The chain's structural integrity is computed from metadata only: event type, sequence number, timestamp, and previous link (Claims 3(d-f)). Payload data is deliberately excluded. This enables privacy-preserving verification (Claim 3, independent claim): a third-party auditor can verify the complete integrity of the governance chain without ever seeing the contents of any agent interaction. An auditor can confirm that governance was maintained throughout a classified operation without accessing classified information.

Evidence Bundles package the sealed Policy Artifact, signed Enforcement Receipts, Merkle inclusion proofs, and the Portal's public key into a portable verification unit (Claim 9). Any party with the bundle can verify the

complete governance record offline using standard Ed25519 and SHA-256. No network callback. No proprietary tooling. No trust in the producing system.

This architecture completes the supply chain attestation lifecycle:

Stage	Attestation	Proves
Build	SLSA Provenance	Built from expected source by expected builder
Package	OMS Model Signing	Model integrity from build to distribution
Deploy	Container Signing	Deployed artifact matches signed reference
Operate	**Governance Predicate**	**Agent actions evaluated against sealed policy; enforcement decisions signed and chain-linked**

5. Standards Alignment

Every standards body working on AI agent security has independently identified some version of the declaration-enforcement gap. The consistency of this pattern across organizations with different mandates, different constituencies, and different technical approaches confirms that the gap is structural, not incidental.

MCP Server Security Standard (MSSS). MSSS v0.1.0 defines four compliance levels with 24 controls across 8 domains. At L3 (production) and L4 (regulated/critical), the standard requires controls for access management, input validation, and audit logging but provides no mechanism for continuous cryptographic evidence that controls remained enforced between assessments. Sealed Policy Artifacts address L3 pre-commitment. Signed Enforcement Receipts address L4 continuous proof. Offline-Verifiable Evidence Bundles address assessor verification.

CoSAI Workstream 4. CoSAI WS4's MCP security analysis identifies 12 threat categories across approximately 40 distinct threats. The white paper explicitly states that "follow-on papers will provide reference implementations and recommendations for specific mitigation controls." Across all 12 categories, mitigations are recommended but no mechanism exists to prove they were applied. A detailed enforcement evidence mapping is published at attestedintelligence.com/diligence/cosai-ws4-mcp-threat-analysis.

OWASP MCP Top 10. Cryptographic enforcement evidence addresses at least five of the ten risks directly: scope enforcement via sealed policies, tamper-evident tool invocation history, continuous authentication via TTL-based re-attestation, cryptographic audit trails replacing mutable logs, and sealed upstream server identity validation against shadow MCP servers.

CSA / CSAI Foundation. Launched March 23, 2026 with the mission of "Securing the Agentic Control Plane." Evidence Bundles provide structured telemetry for their AI Risk Observatory. Sealed policies and signed receipts implement Agentic Best Practices for runtime authorization. Offline-verifiable bundles support STAR-style assessment workflows where auditors verify governance evidence without network access to the assessed system.

Sigstore / OpenSSF. Sigstore covers supply-chain provenance: was this model built from the expected source? Nothing in the current attestation chain covers what happens after deployment. Sigstore proves "this is the right model." The Seal/Enforce/Prove architecture proves "this model obeyed its policy." Together, they complete the build-to-deploy-to-operate attestation chain.

6. The MCP Governance Gap

The Model Context Protocol creates a specific governance challenge that general-purpose frameworks do not address. MCP mediates the interaction between AI agents and external tools via JSON-RPC 2.0. When

an agent calls a tool through MCP, it exercises delegated authority: the tool executes with permissions the agent was granted, performing actions the agent decided to take.

This delegation creates an accountability gap. The agent decides. The tool executes. The MCP transport carries the instruction. But no component in the standard MCP architecture produces cryptographic proof that the decision was authorized by a governing policy, that the policy was immutable at the time of the decision, or that the complete history of decisions is tamper-evident and independently verifiable.

MCP's JSON-RPC 2.0 transport is the natural interception point. A governance proxy sitting between the agent and the MCP server intercepts every `tools/call` request, evaluates it against the sealed policy artifact, blocks unauthorized calls before they reach the server, and generates a signed receipt for every decision. The proxy holds the Ed25519 signing keys. The agent holds none. The enforcement boundary is architectural - a separate process with separate key material - not a configuration setting that can be toggled.

> The reference implementation demonstrates the complete lifecycle. Seal a policy governing an MCP agent's tool access. Start an enforcement proxy. Route tool calls through it. Observe permitted calls passing through and denied calls blocked. Export an evidence bundle. Verify the bundle offline. Seven tool calls. Seven signed receipts. Three PERMIT, three DENY, one DRIFT. One evidence bundle. Independent verification on a disconnected machine using zero framework imports.

7. Case Study: When Configuration Replaces Enforcement

On March 27, 2026, it was reported that a major AI lab's most capable model - described internally as "currently far ahead of any other AI model in cyber capabilities" - was discovered not through an announcement but through a misconfigured content management system that exposed approximately 3,000 unpublished assets. The exposure was attributed to "human error."

The incident illustrates the declaration-enforcement gap precisely. The organization undoubtedly had a governance policy restricting access to pre-release model information. The policy existed. The intent was clear. But enforcement depended on correct CMS configuration - a mutable, human-managed control that failed silently. No cryptographic mechanism constrained the system to behave according to the declared policy. No signed receipts recorded access control enforcement decisions. No evidence bundle could prove, after the fact, that the access controls were active during the period in question.

This is not an isolated pattern. In February 2026, the same dynamic played out at geopolitical scale: governance policies for the use of force existed, authorization chains were defined, but the enforcement mechanisms were configuration-dependent and the evidence trail was mutable. The structural problem is identical whether the governed system is a CMS, a weapons authorization chain, or an autonomous AI agent: mutable configuration cannot substitute for cryptographic enforcement.

Cryptographic enforcement changes the failure mode fundamentally. A sealed artifact cannot be misconfigured without breaking its Ed25519 signature. An enforcement boundary that intercepts every action cannot be silently bypassed because the governed subject holds no keys and has no alternative path to external resources. A signed receipt chain cannot be altered without invalidating every subsequent SHA-256 hash link. The system either enforces the policy and produces proof, or it fails closed and produces proof of the failure. There is no silent middle ground.

8. Implementation Path

For an organization operating MCP-based AI agents, the path from declaration to proof follows four steps.

Step 1: Define the policy. Specify which tools the agent may invoke, which are denied, what rate limits apply, what required parameters must be present for sensitive operations, and what enforcement actions trigger upon violation. This is the governance declaration that most organizations already have in some form.

Step 2: Seal the policy. Sign the policy document with Ed25519 to produce a sealed artifact. Any subsequent modification invalidates the signature. The artifact becomes a cryptographic commitment: the governance parameters are locked before the agent begins operating, and the agent cannot alter them because it does not hold the signing key.

Step 3: Route through enforcement. Configure the AI agent to reach MCP servers through the governance proxy rather than directly. The proxy validates the sealed artifact signature at startup, then evaluates every `tools/call` request against the sealed constraints. Permitted calls are forwarded with a signed PERMIT receipt. Denied calls return a JSON-RPC error without forwarding, with a signed DENY receipt. Every decision is cryptographically recorded regardless of outcome.

Step 4: Export and verify. At any point, export an evidence bundle containing the sealed artifact, all signed receipts, Merkle inclusion proofs, and the public key. Transfer the bundle to any machine. Verify using standard Ed25519 signature checking and SHA-256 hash computation. The verification is a deterministic mathematical operation producing the same PASS/FAIL result regardless of where or when it is executed.

The reference implementation is available as an open-source MCP server (npm: @attested-intelligence/aga-mcp-server) and Python SDK (PyPI: aga-governance). An independent verifier using zero framework imports is available at attestedintelligence.com/verify.

C O N C L U S I O N

The AI governance industry has built comprehensive declaration frameworks. It has not built comprehensive evidence frameworks. The gap between "we declared a policy" and "we can prove the policy was enforced" is where governance fails in practice.

Closing this gap does not require novel cryptography. It requires applying standard cryptographic primitives - Ed25519 digital signatures, SHA-256 hash chains, Merkle trees with inclusion proofs - to the specific problem of runtime governance enforcement. The architecture described here produces evidence that is sealed (immutable policy reference bound by Ed25519 signature), signed (hash-linked enforcement receipts with per-decision cryptographic attribution), and portable (offline-verifiable evidence bundles requiring only standard cryptographic operations).

Every standards body working on AI agent security has identified some version of this gap. MSSS needs continuous compliance evidence between assessments. CoSAI WS4 needs enforcement proof for its 12 categories of MCP mitigations. OWASP needs tamper-evident audit trails for at least five of its ten MCP risks. CSA/CSAI needs assessment artifacts for its STAR-style governance certification. Sigstore needs runtime attestation to complete the build-to-deploy-to-operate chain. The standards contributions are underway - engagement details and technical documentation are at attestedintelligence.com/standards.

The architecture exists. The reference implementation has 231+ automated tests and 26 cross-language test vectors. The evidence bundles verify. The independent verifier uses zero proprietary imports. What remains is integration: embedding cryptographic governance evidence into the standards that will define how autonomous AI agents are governed.

Attested Intelligence Holdings LLC

USPTO App. No. 19/433,835 (Patent Pending)

USPTO Trademark Serial No. 99677085

attestedintelligence.com