**WP-015** | Agentic AI Governance | June 2026

# Cryptographic Audit Trails for Autonomous AI Agents: Zero-Knowledge Proof Chains for Agentic Governance

AffixIO Research June 2026 WP-015 Download PDF

Agents plan, call tools, and delegate to other agents with little supervision. Eighty-four percent of teams fail agent compliance audits using ordinary logs. We describe per-action zero-knowledge proof chains that bind identity, policy scope, and authorisation without exposing prompts or weights.

CONTENTS



## 1. The Agentic Governance Gap

The enterprise AI landscape shifted decisively in 2025 and 2026. Where 2023 deployments were primarily chatbots and retrieval assistants, 2026 deployments are largely agentic: systems where language models plan sequences of actions, invoke external tools, write and execute code, browse the web, and coordinate with other agents, all with minimal human supervision. A survey of 285 IT and security professionals published in 2026 found that 84% cannot pass a compliance audit focused on agent behaviour, only 23% have a formal agent identity strategy, and just 18% are confident their identity and access management infrastructure can handle agent identities at all.

The scale of this deployment wave makes the governance gap acute. In April 2026, six cybersecurity agencies from five governments, including CISA, the NSA, the UK National Cyber Security Centre, Australia's ASD, Canada's Centre for Cyber Security, and New Zealand's NCSC, issued joint guidance on securing agentic AI. Microsoft released the Agent Governance Toolkit the same month. Google began assigning cryptographic identities to every agent built on their platform. These responses signal that regulators and hyperscalers alike recognise the problem, even if no comprehensive solution yet exists.

We think the governance gap is architectural rather than procedural. Organisations are attempting to govern agentic behaviour using audit infrastructure designed for static software: mutable logs, human-readable execution traces, and retrospective reviews. None of these mechanisms are adequate for autonomous agents that act faster than humans can review, modify their own behaviour based on context, and produce outputs that are difficult to replay or verify after the fact.

Our approach a different foundation: per-action zero-knowledge proof chains that produce cryptographic evidence of what an agent did, under what policy constraints, in what context, and with what authorisation, without requiring that the underlying model weights, proprietary system prompts, or user data be disclosed to the auditor.

---

## 2. What Autonomous AI Agents Actually Do

To design effective agentic governance, it is necessary to understand the anatomy of agentic action. A modern autonomous AI agent operating in an enterprise pipeline typically follows a planning loop: observe context, form a plan, select a tool, invoke the tool, observe the result, and repeat. At each step, the agent is making decisions that may have significant downstream consequences.

### Tool Invocations

Agents invoke tools, which may include web search, code execution sandboxes, database queries, API calls to third-party services, file system operations, and sub-agent spawning. Each tool invocation is an action with a real-world effect. A tool that sends an email, modifies a database record, or calls a payment API cannot be undone by simply restarting the agent.

### Multi-Hop Reasoning and Memory

Agents maintain working memory across a context window and may also use external persistent memory stores. Information retrieved in one step influences decisions in subsequent steps, creating reasoning chains that are difficult to decompose retrospectively. Multi-hop reasoning introduces compound risk: an error or adversarial input at any step propagates through the chain.

### Sub-Agent Delegation

Orchestrator agents delegate tasks to specialised worker agents. A research orchestrator may spawn a retrieval agent, a synthesis agent, and a citation-verification agent. Each delegation introduces a new identity, a new scope, and a new opportunity for policy violation. The orchestrator cannot verify what the worker did; it can only observe the output.

### Context Injection and Prompt Construction

At each turn, the agent's context window is constructed from retrieved documents, tool outputs, and conversation history. This construction process is itself a governance surface: adversarial content injected via a retrieved document, a technique known as

prompt injection, can redirect the agent's behaviour without the orchestrator or user being aware.

---

## 3. OWASP Agentic Top 10 and the Audit Problem

In December 2025, OWASP published the Top 10 for Agentic Applications for 2026, the first formal taxonomy of risks specific to autonomous AI agents. Each risk category has direct implications for audit trail design.

| OWASP Agentic Risk | Description | Audit Trail Requirement |
|---|---|---|
| Goal Hijacking | Agent's objective is redirected by adversarial input | Proof of original goal and policy scope at task initialisation |
| Tool Misuse | Agent invokes tools outside its authorised scope | Proof of tool authorisation at each invocation |
| Identity Abuse | Agent impersonates another agent or user | Cryptographic agent identity bound to each action proof |
| Memory Poisoning | Persistent memory store is corrupted with adversarial data | Proof of memory read integrity at each retrieval |
| Cascading Failures | Error in one agent propagates to others | Per-step proof chain enabling failure localisation |
| Rogue Agents | Agent acts outside policy without detection | Continuous policy compliance proof at each action |
| Privilege Escalation | Agent acquires capabilities beyond its initial grant | Immutable capability scope bound to agent session proof |
| Data Exfiltration | Agent leaks sensitive data via tool outputs | Output data classification proof before each tool invocation |

| OWASP Agentic Risk | Description | Audit Trail Requirement |
|---|---|---|
| Unverified Delegation | Orchestrator delegates to sub-agent without authorisation record | Delegation chain proof linking parent and child agent identities |
| Uncontrolled Autonomy | Agent takes consequential actions without human-in-the-loop checkpoints | Proof of human authorisation for high-consequence actions |

Conventional audit logs address none of these risks at the cryptographic level. A log entry stating "agent invoked tool X at timestamp T" does not prove that the tool invocation was within the agent's authorised scope, that the agent identity had not been compromised, or that the tool output was not adversarially modified before being returned to the agent. These proofs require cryptographic commitments, not text records.

---

## 4. Why Mutable Logs Fail Agentic Systems

Traditional audit logging has three fundamental properties that make it inadequate for agentic AI governance. It is mutable, it is correlatable, and it disappears when the system changes.

### Mutability

Standard application logs are mutable. A compromise of the logging infrastructure, a misconfiguration, a software bug, or a deliberate cover-up can alter or delete log entries. In regulated contexts, a log that can be silently modified has the same evidentiary value as no log at all. SIEM platforms introduce some integrity controls, but the underlying log data remains writable at the storage layer.

### Model Churn and Log Evaporation

Agent behaviour is determined by the model, the system prompt, the tool definitions, and the retrieval context. When any of these change, the agent behaves differently. If an audit requires evidence about a specific agent action taken six months ago, the original model, system prompt, and retrieval corpus may no longer exist. The log says what happened, but there is no way to verify that the log reflects what the agent actually would have done under the conditions recorded. A 2026 survey found that 33% of organisations lack evidence-quality audit trails and 61% run fragmented logs that are not actionable.

### Correlation Attacks on Agent Logs

Even when agent logs are retained, correlating log entries with specific individuals or decisions creates a privacy surface. Logs that record user queries, retrieved documents, and agent reasoning chains may expose PII, proprietary business logic, or sensitive commercial information. There is no principle of data minimisation in a log that records everything.

### Non-Repudiation Without Cryptography

Non-repudiation, the property that a party cannot deny having taken an action, is not achievable through application logs alone. A log entry does not bind the agent identity to the action in a way that can be verified by a third party without trusting the log infrastructure. Only a cryptographic proof, signed by a key provably held by the agent at the time of action, achieves genuine non-repudiation.

---

## 5. Zero-Knowledge Proof Chains for Agent Actions

AffixIO's agentic governance architecture replaces mutable execution logs with a chain of zero-knowledge proofs, one per significant agent action, that together constitute a cryptographically verifiable record of the agent's behaviour throughout a task session.

### The Action Proof Primitive

For each agent action, the proof system generates a ZK proof asserting the following statements simultaneously:

- The agent's identity commitment matches the identity registered at session initialisation
- The action type is within the set of actions authorised by the agent's policy
- The policy was not modified after session initialisation
- The tool invocation parameters satisfy the schema defined in the tool's authorisation record
- The session nonce for this action is unique and has not been previously seen

None of these statements require disclosing the agent's system prompt, the specific content of the action, or the identity of the user on whose behalf the agent is acting. The proof is a cryptographic commitment: a verifier can confirm all five properties without learning any of the sensitive inputs.

### Proof Chaining

Individual action proofs are chained using a hash-linked structure. Each proof includes a commitment to the previous proof's root, creating a chain in which any retrospective modification of an earlier proof would invalidate all subsequent proofs. This property is analogous to a blockchain but requires no distributed consensus: the chain is maintained by the AffixIO proof service and anchored periodically to an external append-only log.

> **Proof Chain Integrity Property**
>
> Given a chain of N action proofs P1 … PN, an adversary who modifies Pi must also recompute all proofs Pi+1 through PN. Because each proof requires knowledge of the agent's session signing key, which is held in a hardware security module, retroactive modification is computationally infeasible without access to the HSM.

### Session Root Anchoring

At the end of each agent session, the root of the proof chain is anchored to an external immutable log, such as a transparency ledger or an append-only S3 bucket with

object lock enabled. The anchor record contains the session identifier, the number of actions in the session, the final proof chain root, and a timestamp. This anchor serves as the audit reference: any dispute about agent behaviour can be resolved by reconstructing the proof chain from the session and verifying against the anchor.

---

## 6. Intent-Verified Delegation Architecture

The most significant governance challenge in multi-agent systems is delegation: when an orchestrator agent creates or invokes a sub-agent, how is the sub-agent's authorisation bound to the parent's intent and to the user's original request?

### Delegation Chains as Cryptographic Records

In AffixIO's architecture, every delegation event produces a delegation proof. This proof records:

- The parent agent's identity commitment
- The sub-agent's identity commitment
- The capability scope granted to the sub-agent (as a policy commitment)
- The task context commitment (a hash of the sub-task description)
- A session nonce preventing replay of the delegation record

The sub-agent's subsequent action proofs include a commitment to the delegation proof, binding every action the sub-agent takes back to the explicit authorisation granted by the parent. An auditor can verify that every tool invocation in a multi-agent pipeline traces back to a chain of delegation records that ultimately connect to the user's original request.

### Scope Inheritance and Attenuation

A fundamental security principle in delegation is attenuation: a sub-agent may be granted at most the capabilities held by the parent, never more. AffixIO enforces this at the proof layer. The delegation proof circuit verifies that the capability set

committed in the sub-agent's grant is a strict subset of the capability set committed in the parent's session record. Attempts to grant capabilities the parent does not hold produce an invalid proof that fails verification.

### Human-in-the-Loop Integration

For high-consequence actions, such as sending external communications, modifying financial records, or invoking APIs with irreversible effects, the AffixIO architecture requires a human authorisation proof. This proof is generated when a human operator explicitly approves the action via a signed interaction, and it is included in the action proof chain. An auditor can verify not only that the action was taken but that a specific human operator authorised it at a specific time.

---

## 7. The AffixIO Agentic Proof Pipeline

The AffixIO agentic proof pipeline integrates with existing agent frameworks without requiring modifications to the model itself. It operates as a sidecar service that intercepts tool invocations, generates proofs, and maintains the session proof chain.

### Integration Architecture

```
User Request
    |
    v
Orchestrator Agent (LLM)
    |--- Plan Step 1 ---> AffixIO Proof Sidecar ---> Action Proof 1
    |--- Tool Invoke ---> AffixIO Proof Sidecar ---> Action Proof 2
    |--- Sub-Agent Spawn -> AffixIO Proof Sidecar ---> Delegation Proof
    |--- Tool Invoke ---> AffixIO Proof Sidecar ---> Action Proof 3
    v
Session Complete
    |
    v
AffixIO Proof Aggregator --> Session Root --> Anchor Record
```

### Proof Generation Latency

Using current proving scheme with proving backend, per-action proof generation takes between 8 and 45 milliseconds depending on circuit complexity. This is well within the latency budget of typical agentic pipelines, where tool invocations themselves take hundreds of milliseconds to seconds. The attestation companion runs asynchronously relative to tool execution, so it does not add to the critical path latency.

### Compatibility with Major Agent Frameworks

The attestation companion exposes a standard middleware interface compatible with common agent orchestration frameworks, and custom agentic frameworks built on the OpenAI function-calling API. Integration requires adding the sidecar as an interceptor in the tool invocation chain, a change of fewer than twenty lines in most frameworks.

---

## 8. Multi-Agent Coordination and Proof Aggregation

In complex multi-agent pipelines, hundreds of actions may occur across dozens of agents within a single user session. Maintaining and verifying hundreds of individual action proofs is impractical for routine audit purposes. AffixIO addresses this through proof aggregation.

### Session-Level Merkle Trees

All action proofs within a session are leaves of a session-level Merkle tree. The session root commits to all agent actions in the session without requiring an auditor to verify each proof individually. To verify a specific action, the auditor requests an inclusion proof for that leaf: a path of sibling hashes from the leaf to the root, which can be verified in logarithmic time.

### Cross-Agent Proof Consistency

When sub-agents return results to an orchestrator, the orchestrator's subsequent reasoning is influenced by those results. AffixIO captures this dependency by

including, in the orchestrator's next action proof, a commitment to the sub-agent's last proof root. This creates a directed acyclic graph of proof dependencies that an auditor can traverse to understand how information flowed across agents and how each agent's actions depended on others.

### Batch Verification

For compliance reporting and retrospective audit, AffixIO provides a batch verification endpoint that accepts a session identifier and returns a structured report: total actions, policy violations detected, delegation chain depth, and anchoring status. The report is itself cryptographically signed, enabling it to be submitted to regulators as evidence without requiring regulator access to the underlying proof system.

---

## 9. Regulatory Mapping: EU AI Act, NIST, ISO 42001

Agentic AI systems are subject to several overlapping regulatory frameworks, each with distinct audit trail requirements.

### EU AI Act

The EU AI Act's high-risk classification covers AI systems used in employment, critical infrastructure, education, law enforcement, and access to essential services. Agentic AI systems operating in these domains must comply with Articles 11 (technical documentation), 12 (record-keeping), 13 (transparency), 14 (human oversight), and 15 (accuracy, robustness, cybersecurity). Article 12 specifically requires that high-risk AI systems be designed to automatically log events to the extent necessary to identify risks. AffixIO's proof chains provide a log that is not merely retained but cryptographically verified, exceeding Article 12's minimum requirements.

### NIST AI Agent Standards

The NIST AI Agent Standards Initiative, launched in February 2026, explicitly identifies agent security and identity as core pillars. The initiative builds on the NIST AI Risk Management Framework (AI RMF) and ISO/IEC 42001:2023, adding agent-specific

requirements around identity verification, capability scoping, and action attribution. AffixIO's cryptographic agent identity commitments and capability scope proofs directly satisfy these requirements.

### ISO 42001 and ISO 42005

ISO/IEC 42001:2023 is the management system standard for AI, analogous to ISO 27001 for information security. ISO/IEC 42005:2025 extends this to AI system impact assessment. Both standards require documented governance processes for AI systems, including evidence of control effectiveness. AffixIO's proof chains provide machine-verifiable evidence of governance control operation, enabling automated compliance reporting against both standards.

### TRiSM Alignment

Gartner's AI Trust, Risk, and Security Management (TRiSM) framework emphasises four pillars: explainability, ModelOps, data anomaly detection, and adversarial attack resistance. AffixIO's agentic proof chain addresses explainability (every action is attributed and bounded), adversarial attack resistance (goal hijacking and prompt injection are detectable through policy scope violations in proofs), and operational governance (proof chain provides the operational record required by ModelOps processes).

---

## 10. TRiSM for Agentic AI: Trust, Risk, and Security

Governance of autonomous AI agents requires a framework that goes beyond point-in-time compliance checks to provide continuous, runtime trust assurance. AffixIO applies the TRiSM principles at the infrastructure layer rather than the application layer.

### Trust: Cryptographic Agent Identity

Trust in a multi-agent system begins with identity. AffixIO assigns every agent a session keypair generated in a hardware security module. The public key is registered

in an agent identity ledger at session creation. All action proofs are signed with this key. An auditor verifying an action proof can confirm that it was produced by the specific agent instance registered in the identity ledger, not by a substitute or impersonator.

### Risk: Continuous Policy Verification

Risk management in agentic systems requires continuous verification, not periodic review. Because AffixIO generates a proof for every action, policy violations are detectable in real time. The attestation companion can be configured to block actions that would produce an invalid proof, effectively enforcing policy at the infrastructure layer before the action is executed. This converts governance from a post-hoc audit function to a runtime enforcement mechanism.

### Security: Adversarial Input Detection

Prompt injection attacks, where adversarial content in a retrieved document attempts to redirect the agent's behaviour, manifest in AffixIO's system as sudden changes in the agent's action scope. If an agent that has been operating within a narrow tool set suddenly attempts to invoke a tool outside its authorised scope, the resulting proof fails validation. The failure is flagged in real time, and the action is blocked pending human review. This does not require detecting the prompt injection itself; it requires only that the injected instruction causes the agent to attempt an out-of-scope action.

---

## 11. Implementation Considerations

### Defining the Policy Commitment

The most important implementation decision is how to define the agent's policy commitment. A policy that is too broad, permitting any tool invocation, produces proofs that are technically valid but governance-worthless. A policy that is too narrow produces excessive false-positive violations that disrupt legitimate agent behaviour. AffixIO recommends a graduated approach: begin with a permissive policy during the

agent development phase, tighten based on observed tool usage patterns, and lock the policy before production deployment.

### Key Management for Agent Session Keys

Session keypairs must be generated in an HSM or secure enclave and must not be accessible to the agent process itself. If the agent could read its own private key, a compromised agent could forge proofs. The attestation companion holds the key and signs proofs on behalf of the agent, receiving only the public commitments from the agent process.

### Storage and Retention of Proof Chains

Proof chains are compact: a typical action proof is between 400 and 800 bytes. A session with 1,000 agent actions produces approximately 500 kilobytes of proof data. Retention periods should align with regulatory requirements, typically seven years for financial services, five years for healthcare, and three years for general commercial purposes under the EU AI Act.

### Incident Response Integration

Proof chain data should be integrated with the organisation's SIEM and incident response workflows. When a proof validation failure is detected, the session identifier, the specific action proof, and the violation type should be forwarded to the SIEM as a high-priority event. The immutable proof chain then provides the forensic record needed to reconstruct the incident.

---

## 12. Conclusions

Autonomous AI agents represent the most significant governance challenge in enterprise AI since the deployment of the first production language models. The combination of autonomous action, tool access, multi-agent coordination, and persistent memory creates a risk surface that conventional audit logging cannot adequately address.

AffixIO's zero-knowledge proof chain architecture provides a fundamentally different governance foundation. Per-action cryptographic proofs bind agent identity, policy scope, and authorisation records together in a tamper-evident chain that survives model updates, system reconfigurations, and adversarial interference. Delegation proofs trace every sub-agent action back to the user's original request through a verifiable chain of authority. Session Merkle roots provide compact audit anchors that can be verified without disclosing underlying data.

As governments mandate cryptographic AI agent security, as OWASP formalises agentic risk taxonomies, and as the EU AI Act's Article 12 obligations come into force for high-risk agentic deployments, organisations that have built their governance on mutable logs will find themselves unable to meet evidentiary standards. Organisations that have built on cryptographic proof chains will be able to demonstrate compliance, isolate failures, and respond to incidents with precision.

AffixIO's agentic proof pipeline is available as an open-source middleware component compatible with common agent orchestration frameworks, and custom agent frameworks, with enterprise support and managed proof anchoring available for regulated deployments.

## Related reading

- WP-003: The Proof-Not-Log Paradigm for AI Audit Trails
- WP-004: Real-Time Zero-Knowledge Governance in the AI Response Pipeline
- WP-022: RAG Citation Integrity: Per-Chunk ZK Proofs for Agentic AI

---

## Frequently asked questions

### Why do logs fail for agentic AI?

Agents mutate state quickly, share credentials, and get updated mid-session. Logs are editable and hard to replay after the fact.

### What is an agent proof chain?

A linked sequence of compact proofs, one per action, each signed and Merkle-anchored so auditors verify behaviour without vendor access.

### Which regulations apply?

EU AI Act Articles 11 to 15, NIST AI agent guidance, ISO 42001, and OWASP Agentic Top 10 controls map directly to proof artefacts.

---



All White Papers

▶ About
▶ Solutions
▶ Legal
▶ Trust & Security

Contact

truth layer | yes | no | proof