AffixIO AFFIXIO

YES NO

Sandbox Contact Us

AffixIO Technical Paper · WP-003
June 2026
affix-io.com

AFFIXIO WHITE PAPER · WP-003

The Proof-Not-Log Paradigm: Replacing Mutable AI Audit Logs with Cryptographic Proofs

Logs tell you what someone said happened. Proofs show the maths.

AffixIO | United Kingdom | affix-io.com | June 2026

ABSTRACT

Application logs are cheap and editable. For AI decisions that may be disputed years later, that is a problem. We argue for proof-not-log: ZK proofs in Merkle trees, signed with ML-DSA-65, as the primary governance artefact at a production deployment.

CONTENTS

SECTION 1

Introduction

Every regulated AI deployment generates a requirement for evidence. When an AI system informs or makes decisions affecting individuals, those decisions need to be auditable: who made the decision, under what policy, at what time, and with what outcome. Regulators, courts, and data subjects all have legitimate claims on this evidence. The question is what form that evidence should take.

The default answer is a log. The AI system writes structured records to a database. The database is retained. When a regulator asks for evidence of a decision, the log record is retrieved and presented. This approach is so deeply embedded in software practice that it is rarely examined critically. Logs are how systems produce evidence. The assumption is universal and largely unquestioned.

The proof-not-log paradigm questions it. A log record is a mutable database entry whose integrity depends entirely on the trustworthiness of the infrastructure operator and the access controls around the database. A ZK proof anchored in a Merkle tree and signed with a post-quantum key is a cryptographic object whose integrity can be verified by any third party using public information, regardless of the infrastructure operator's trustworthiness. These are not equivalent forms of evidence. In regulated contexts where AI decisions carry legal weight, the difference is material.

AffixIO operates a proof-not-log system in production. Every AI response served through the platform generates a ZK proof before the user sees the response. The proof is anchored in a Merkle tree. The root is signed with ML-DSA-65. The audit record consists of the proof digest, the leaf hash, and the signed root. No AI response content is retained. No user input is stored. Any third party with the published public key can verify any proof in the system without contacting AffixIO. We explain why that architecture is structurally superior to log-based governance, and how to build it.

SECTION 2

What Logs Cannot Prove

Log-based AI audit infrastructure has four structural weaknesses that matter specifically in regulatory and legal contexts. These weaknesses are not failures of implementation: they are properties of the log-based approach that no amount of access control, encryption, or procedural safeguard can fully eliminate.

Mutability

A log record in a relational database can be altered by anyone with sufficient database privileges. Append-only log stores, write-once object storage, and log integrity services address this partially, but they replace one trust assumption with another. The question moves from "can the record be altered?" to "can the log integrity service be trusted?" A third party presented with a log record cannot independently verify that the record has not been altered since it was written without access to the log integrity infrastructure. In legal proceedings, this dependency on the infrastructure operator's trustworthiness is a weakness that adversaries routinely exploit.

No cryptographic binding to policy version

A log record notes what decision was made. It typically does not cryptographically bind that decision to the specific version of the policy that governed it at the time. If an organisation updates its AI policy, records from before and after the update are structurally indistinguishable in the log. When a regulatory inquiry asks specifically which policy governed a decision made on a given date, the answer depends on administrative records and version control history, not on any cryptographic property of the decision record itself. A ZK proof embeds the circuit identifier, which encodes the policy version, as part of the proof structure.

Verifiability requires data access

An auditor who wants to verify that a logged AI decision was made correctly needs access to the log record and typically to the AI system configuration at the time of the decision. There is no way to provide evidence of a specific decision to a third party without also providing access to the underlying data infrastructure. ZK proofs change this: the proof digest, the Merkle path, and the signed root are sufficient for a third-party verifier. No access to the original AI response, user input, or system configuration is required.

Data retention liability

A log record that includes the AI response and the user input retains personal data. Under GDPR Article 5, personal data must not be kept longer than necessary for the purpose for which it was collected. The longer the log retention period required for audit purposes, the more significant the data minimisation exposure. A log database breach may expose user communications at scale. A proof record that contains only cryptographic hashes of content retains no personal data and carries no retention liability beyond the storage of hash values.

SECTION 3

The Zero-Knowledge Proof as Primary Governance Artefact

A zero-knowledge proof is a cryptographic object that demonstrates a computation reached a specific result without revealing the inputs to that computation. Applied to AI governance, a ZK proof demonstrates that a given AI response was evaluated by a specific circuit, that the circuit produced a specific output (typically "yes" or "no" relative to a policy condition), and that this evaluation occurred at a specific time. The proof does not reveal the content of the AI response. The proof does not reveal the user's input. The proof is bound to the circuit version that was active when it was generated, which encodes the policy version cryptographically.

Three properties make the ZK proof superior to a log entry as a governance artefact in regulated contexts. First, it is computationally impossible to alter the proof after generation without invalidating it: the proof is either valid or it is not, and validity is verifiable by any party with the verification key. Second, it proves a specific computation was performed correctly, not merely that a record claims it was performed: the mathematical structure of the proof guarantees the computation, whereas a log record only asserts it. Third, it reveals nothing about the inputs beyond the fact that they satisfied the circuit's constraints, providing data minimisation that a log entry cannot achieve without losing evidentiary value.

The circuit as policy record

In AffixIO's production system, the constraint-based circuits function as machine-readable policy documents. The primary policy gate circuit encodes a three-condition AND policy: all three binary conditions must be satisfied for the circuit to output 1 (YES). The identity eligibility circuit encodes identity verification rules. The threshold eligibility circuit encodes age and health threshold conditions. When a proof is generated using a specific circuit version, the circuit identifier is embedded in the proof structure. A proof generated under the current primary policy gate circuit cannot be presented

as having been generated under a previous version. The policy version is not an administrative claim appended to the record: it is a cryptographic commitment within the proof.

Proof digests as stable identifiers

The proof digest, computed as `a composite digest over circuit identity, outcome, and proof material`, serves as the stable identifier for any governance event in the system. All downstream records, including the Merkle leaf hash, the record service audit entry, the audit reference in the verification disclosure, and the webhook event payload, reference the proof digest as their primary key. This creates a consistent, cryptographically derived identifier that does not depend on database sequence numbers or UUIDs generated by infrastructure that could be compromised.

SECTION 4

Merkle Anchoring and Tamper Evidence

A ZK proof without anchoring is a cryptographic object that proves a computation occurred, but which exists in isolation: it does not prove when it was generated, or that it has not been fabricated after the fact by someone with access to the circuit. Merkle tree anchoring addresses both problems. When a proof digest is inserted as a leaf in an append-only Merkle tree, its position in the tree is determined by all the leaves that preceded it. Any alteration to any earlier leaf changes the root. Any attempt to insert a proof at an earlier position than its actual insertion changes the root. The root is a cryptographic summary of the complete ordered history of all proofs inserted to date.

AffixIO's Merkle tree uses SHA-256 sorted-pair hashing: for any two sibling nodes A and B, the parent is `SHA-256(min(A,B) + max(A,B))` where ordering is lexicographic on the hex representation. This is the same scheme used in Bitcoin transaction Merkle trees and RFC 9162 Certificate Transparency, both of which have undergone extensive cryptanalytic scrutiny. The sorted-pair property means that the same set of leaves always produces the same root regardless of insertion order, which simplifies inclusion proof verification and allows auditors to reconstruct subtrees without knowing the exact insertion sequence.

The append-only roots log

After each proof insertion, the new Merkle root is published to an append-only roots log. Roots in the log are never deleted or overwritten. Each root is timestamped and corresponds to a specific set of anchored proofs. Any retroactive insertion or deletion of a proof would produce a different root at the relevant timestamp, conflicting with the signed attestation record for that root. The tamper-evidence property of the complete audit trail is derivable from the signed attestation records for all published roots, without requiring a dedicated log integrity service. This is a key structural advantage over log-based approaches: the integrity guarantee is derived from cryptography, not from the trustworthiness of a separate integrity infrastructure.

SECTION 5

Post-Quantum Signing and Long-Lived Integrity

A signed Merkle root is the point at which the cryptographic audit trail becomes legally meaningful evidence. The root encodes the complete proof history to a given point in time. The signature binds that root to the signing entity's key at a specific timestamp. Any party who holds the signed root and the public key can verify that the root was produced by the holder of the corresponding private key at the stated time.

AffixIO signs Merkle roots with ML-DSA-65, standardised as NIST FIPS 204 in August 2024. The choice of ML-DSA-65 rather than a classical signature scheme (ECDSA, RSA) reflects the long-lived nature of AI governance records. A governance record generated today may be presented as evidence in regulatory proceedings a decade from now. Classical signature schemes are vulnerable to store-now-decrypt-later attacks: an adversary who captures signed roots today can store them and, when a sufficiently capable quantum computer exists, forge alternative roots that appear to carry the same signature. ML-DSA-65 is based on Module Learning With Errors, for which no quantum algorithm provides a meaningful speedup. Governance records signed with ML-DSA-65 today remain resistant to signature forgery for any foreseeable timeline.

The HSM root of trust

The ML-DSA-65 private key is generated and stored inside an a FIPS-validated hardware security module cluster in a designated region. The private key never leaves the HSM hardware boundary. Physical tamper attempts trigger automatic key zeroisation. Signing requests are authenticated via AWS KMS and executed inside the HSM. The only output is the signature. This means that a complete compromise of the application server does not enable an attacker to forge attestation signatures for past or future proofs: the private key operation is physically isolated from the application layer.

SECTION 6

Third-Party Verifiability Without Vendor Access

The single most important structural difference between a log entry and a ZK proof in a signed Merkle tree is third-party verifiability. To verify a log entry, a third party needs access to the log database, trust in the log infrastructure operator, and confidence that the access controls preventing alteration have not been breached. To verify a ZK proof in AffixIO's system, a third party needs four pieces of public information: the leaf hash, the Merkle sibling path, the signed root, and the ML-DSA-65 public key. All four can be held in a self-contained verification bundle retained alongside the governance record. None of the four requires querying AffixIO's infrastructure.

This property is not a convenience feature. It is the foundation of genuine accountability. A governance system whose audit records can only be verified by the operator of that system does not provide independent accountability: it provides the operator's representation of what happened. A governance system whose audit records can be verified by any technically capable third party, using open standards and public information, provides accountability that is independent of the operator's continued cooperation, continued operation, and continued trustworthiness. For AI decisions that may be challenged in courts or regulatory proceedings years after they were made, this distinction is decisive.

> **Verification without AffixIO:** Any party holding a self-contained verification bundle can confirm a proof's inclusion by (1) recomputing the Merkle root from the leaf hash and sibling path, (2) verifying the ML-DSA-65 signature on the root using the published public key, and (3) confirming the recomputed root matches the signed root. No network request to any AffixIO endpoint is required.

SECTION 7

Data Minimisation as a Structural Property

Under GDPR Article 5(1)(c), personal data must be adequate, relevant, and limited to what is necessary. Under Article 25, data minimisation must be implemented by design and by default. Log-based AI audit systems typically retain the AI response and the user input as part of the audit record, because retaining the underlying content is what makes the record interpretable. This creates a structural tension: the more complete the audit record, the more personal data it retains; the less personal data it retains, the less interpretable it becomes.

The proof-not-log approach resolves this tension. The ZK proof proves that a computation occurred and produced a specific result without retaining the inputs. The audit record in AffixIO's system contains the proof digest (a hash of the proof), the leaf hash (a hash of the audit payload), the Merkle root (a hash of all proofs to that point), the signed attestation, and the audit reference (which encodes hashed identifiers, not plaintext). No AI response content is stored. No user input is stored. The audit record is complete as an evidentiary object, it proves that a specific policy evaluation occurred with a specific outcome at a specific time, while retaining no personal data at all.

This is not achieved through careful configuration or access control. It is a structural property of the data schema: the compliance record service's record format has no field for plaintext response content or user input. Operators cannot accidentally configure the system to retain personal data in the audit record because there is no field to put it in. Data minimisation is enforced by the schema, not by policy. This distinction matters for GDPR Article 25 compliance: "by design" requires that the privacy-protective property is built into the system, not merely configured on top of it.

SECTION 8

Regulatory Case for Proof-Based Governance

Three regulatory frameworks create direct demand for proof-based AI governance: the EU AI Act, the NIST AI Risk Management Framework, and ISO 42001:2023. Each framework's requirements map more naturally onto cryptographic proofs than onto mutable log entries.

EU AI Act

Article 11 of the EU AI Act requires that high-risk AI system providers maintain technical documentation throughout the system lifecycle. Documentation that includes mutable log records is documentation whose integrity depends on the infrastructure operator. Documentation that includes signed ZK proofs in a Merkle tree is documentation whose integrity is cryptographically verifiable by the regulator independently. Article 15 requires appropriate levels of accuracy, robustness, and cybersecurity. The post-quantum signing layer and HSM key custody of the proof-not-log system provide a cybersecurity baseline that log-based systems cannot match without significant additional infrastructure. Article 17 requires quality management systems. A proof pipeline that generates a cryptographic governance record for every AI decision, automatically and without manual intervention, is a quality management mechanism with mathematical enforcement properties.

NIST AI RMF and ISO 42001

Both frameworks require systematic measurement of AI governance performance. The proof-not-log system generates quantified measurements automatically: proof counts, source verification rates, circuit outcomes, Merkle root histories. These are produced as a side effect of the governance operation, not as a separate reporting activity. ISO 42001 Clause 9 requires performance evaluation including monitoring, measurement, analysis, and evaluation. The audit trail produced by the proof-not-log system satisfies Clause 9 requirements continuously rather than at periodic reporting intervals.

SECTION 9

Implementation Architecture

The proof-not-log architecture has five layers, each of which can be adopted independently or in combination, depending on an organisation's starting point and compliance requirements.

LAYER	COMPONENT	ROLE	REPLACES
Proof generation	policy circuit + proving backend	Generates ZK proof of policy evaluation	Log write
Proof anchoring	SHA-256 Merkle tree	Binds proof to ordered history	Log integrity service
Attestation signing	ML-DSA-65 via CloudHSM	Provides post-quantum tamper evidence	Log certification
Audit persistence	compliance record service	Multi-tenant record storage and export	Log database
Verification	Self-contained bundles	Third-party verifiable without vendor	Auditor database access

The five layers interact but are designed to be independently defensible. A failure at the proof persistence layer (compliance record service unavailable) does not invalidate proofs already generated and anchored: their signed roots remain valid. A failure at the signing layer (HSM unavailable) suspends new attestations but does not affect the validity of existing ones. The system degrades gracefully because the cryptographic validity of each proof is established at generation time and does not depend on ongoing infrastructure availability.

SECTION 10

Migration from Log-Based Systems

Organisations transitioning from log-based AI audit to proof-based governance have three practical options, which are not mutually exclusive and can be adopted in phases.

Parallel operation

The proof pipeline can operate alongside an existing log infrastructure during a transition period. Every AI decision generates both a log record (for backward compatibility with existing audit processes) and a ZK proof (for forward-looking compliance). The proof pipeline is additive: it does not require changes to existing log infrastructure. Over time, as reliance on the log records is replaced by reliance on the proof records, the log infrastructure can be decommissioned or significantly reduced in scope.

Log anchoring

For historical log records generated before the proof pipeline was operational, a log anchoring approach can provide partial tamper evidence retroactively. Historical log records are hashed and their hashes are inserted as leaves in the Merkle tree. The signed root provides a post-quantum-secure attestation that the complete set of historical records existed at the time of anchoring. Individual records can be verified via their Merkle paths. The original records are not made cryptographically governed in the full proof-not-log sense, but the completeness of the historical set is now cryptographically attested from the anchoring date forward.

Greenfield deployment

For new AI deployments, the proof-not-log approach can be adopted from day one without a migration step. The policy enforcement layer generates proofs for every AI response from the first interaction. The audit trail begins with the first proof, not with the first log record. There is no gap period and no

log-to-proof boundary to manage in future audits. This is the cleanest architecture and the one AffixIO recommends for any AI governance infrastructure being built or substantially rebuilt in 2026.

SECTION 11

Known Limitations

The proof-not-log paradigm has genuine limitations that should be assessed honestly by any organisation evaluating it.

Proof generation adds latency

Generating a production SNARK backend ZK proof takes hundreds of milliseconds. This is acceptable in the AffixIO production pipeline because the overall AI response latency is measured in seconds, but it is non-trivial and may be prohibitive for AI systems with very tight latency requirements. Log writes are essentially instantaneous by comparison.

Proofs prove computation, not input accuracy

A ZK proof proves the circuit was correctly evaluated on the supplied inputs. It does not prove the inputs accurately represent the real world. If the policy enforcement layer supplies incorrect inputs to the circuit (for example, if source verification incorrectly classifies a URL), the proof certifies the evaluation of those incorrect inputs. The proof system cannot detect errors in input preparation. This is a limitation shared with log-based systems, where a log record also cannot self-certify that the logged information accurately represents what actually happened.

Specialist infrastructure requirements

The proof-not-log system requires ZK proof infrastructure (policy circuit compiler, production proving system), an HSM for post-quantum signing, and a multi-tenant record service for proof persistence. These components require more specialist expertise to deploy and operate than a log database. This is a real barrier for smaller organisations and should be weighed against the compliance benefits.

SECTION 12

Conclusion

The proof-not-log paradigm is not a marginal improvement on log-based AI governance. It is a structural change in what kind of evidence is produced and what properties that evidence has. Log entries are mutable database records whose integrity depends on trust in the infrastructure operator. ZK proofs anchored in a Merkle tree and signed with ML-DSA-65 are cryptographic objects whose integrity is mathematically verifiable by any third party using public information, independently of the operator.

The regulatory trajectory is clear. The EU AI Act, the NIST AI RMF, and ISO 42001 all point toward AI governance evidence that is independently verifiable, tamper-evident, and maintained throughout the system lifecycle. Log-based approaches satisfy these requirements procedurally: an operator claims the logs are complete and unaltered, and provides evidence of access controls and log integrity procedures. Proof-based approaches satisfy them cryptographically: the evidence itself carries the integrity guarantee, and any technically capable third party can verify it without the operator's cooperation.

AffixIO operates the proof-not-log system in production. The audit trail is growing continuously. Any proof generated by the system can be verified against the published Merkle roots and public key without contacting AffixIO. The system demonstrates that proof-based AI governance is not a future aspiration: it is operational infrastructure available today.

Related reading

- WP-011: Merkle Tree Audit Architecture for AI Decision Systems
- WP-002: Post-Quantum Attestation in Production with ML-DSA-65
- WP-004: Real-Time Zero-Knowledge Governance in the AI Response Pipeline

Frequently asked questions

Why are audit logs weak evidence?

Anyone with database access can alter rows; third parties cannot verify integrity without trusting the operator.

What replaces a log entry?

A proof digest, Merkle inclusion path, and signed root that any party verifies with published keys.

Can we migrate gradually?

Yes. Run proofs parallel to logs, then anchor historical log hashes into the Merkle tree before cutover.