Security, Authority and Accountability

Whom a user of Lawie trusts, for which property, and what happens when that trust fails

Security paper LL-SEC-01
Version 1.0.0, 26 November 2026

Abstract

A platform that executes legal work for others asks its users to trust many parties at once: the providers of the language models it calls, the official sources of the law it cites, the company that operates it, the administrators who grant its agents their tools, the people who approve its acts, the custodians of its signing keys and the infrastructure beneath it. This paper names each of these parties, states what Lawie trusts it for, and records what fails, how the failure would be noticed, what limits the damage and what remains. It then sets out the controls that carry the weight (signed request identity, tenant scoping, default-deny tool grants, approvals bound to the exact act, hash-chained audit records and signed portable dossiers) with their measured outcomes. In the BEN-02 benchmark three of Lawie's authorization functions denied no legitimate request, refused every scenario of some attack classes and allowed every scenario of others [MEASURED]. The audit chain's verifier detected every tested edit, deletion, reordering, insertion and deleted prefix, and the dossier verifier failed every tampered dossier [MEASURED].

The question that matters to a court, a regulator or a client is whether a third party can determine who authorized an act, on what evidence and with what result, without trusting the operator. We state what Lawie records and what would make the answer verifiable without trusting the operator, and we state what this paper does not claim: Lawie is not audited or certified [ABSENT], we do not call its records tamper-proof [IMPLEMENTED], and nothing described here is in production [BLOCKED].

Prepared by Lawie Labs. *Status:* First edition. *Distribution:* Public. *Basis:* source repository `main` @ `ffa13d4bce` (2026-11-24); experiments of 26 November 2026; EUR-Lex texts retrieved 26 November 2026. *Audience:* Security researchers, auditors, general counsel and risk officers of regulated firms, regulators and diligence teams.

Evidence labels. Every material statement about Lawie carries an evidence class. [IMPLEMENTED] in source at the basis commit; [TESTED] exercised by repository tests at that commit; [MEASURED] measured by an offline experiment of this suite on the stated public or synthetic data, never a production or customer measurement; [ANALYTICAL] follows by argument from stated assumptions; [SIMULATED] output of a stated model; [EXTERNAL] checked against a cited primary source; [PROPOSED] design or protocol not built or not run; [UNVALIDATED] a hypothesis, stated as such; [ABSENT] not implemented; [BLOCKED] awaits an external owner or decision. Statements describe the source repository at the basis commit, not a deployed service: no workflow described here has a dated production acceptance, nothing is independently audited, and no statement has been reviewed by an independent human expert. This is a technical publication, not legal advice, not an offer of services and not a statement that any use of Lawie complies with any law.

Contents

Abstract **1**
1 Purpose **3**
2 The shape of trust **3**
3 Adversaries **4**
4 Trust assumption register **4**
4.1 Sources of content 5
4.2 The operator and its infrastructure 5
4.3 People inside the tenant 6
4.4 Keys and those who verify 6
5 Controls **7**
5.1 Approvals and authorization 7
5.2 Records and dossiers 7
5.3 Fail-closed behaviour 8
6 Accountability **8**
6.1 What Lawie records 9
6.2 What third-party verification requires 9
7 What the evidence establishes, and what it does not **10**
8 What this paper does not claim **10**
9 Assumptions and the properties that retire them **11**
10 Guidance for evaluators **11**
References **12**

1 Purpose

The companion papers analyse mechanisms: LL-GMA-01 proves what an authorization layer can and cannot enforce, LL-BEN-02 measures one, and LL-EVP-01 treats evidence chains. This paper inverts the view. It lists every party and component a user of Lawie must trust, states precisely what each is trusted *for*, and records what breaks, how the failure would be noticed, what limits the damage and what remains. It is written for people whose job is to disagree with it.

Lawie's doctrine is that intelligence is not authority: a model's competence never confers the right to act or the warrant to be believed. The security question follows directly. If authority does not come from the model, it comes from people and institutions, and the system is secure only to the extent that it represents their authority faithfully, refuses what they did not authorize, and leaves a record that someone other than its operator can check. Lawie's own authority model states that design: every consequential act bound to a principal, a named human authority, a scoped and expiring approval receipt, the act and its artefact [PROPOSED]. This paper examines the implementation against it.

> **Not an audit.** Nothing in this paper is the result of an independent audit, a certification or a review by an independent expert; none is recorded [ABSENT]. The analysis is Lawie Labs' own, drawn from source at the basis commit, from repository tests and from the suite's offline experiments. Nothing described is deployed in production [BLOCKED], so no statement concerns a live service, a customer or real legal matters. This edition describes controls as they exist and states requirements for any platform; it contains no field names, thresholds or identifiers. It reports research measurements at a stated code basis. It is not a security assurance, a compliance statement or legal advice, and no independent security, privacy or legal review of it has taken place.

2 The shape of trust

Five properties are often bundled under "security", and each rests on different parties (figure 1). *Confidentiality*: tenant data reaches only those entitled to it. *Authorization*: an act executes only if someone entitled to authorize it did. *Attribution*: the record says who did authorize it. *Record integrity*: the record has not been changed. *Correctness*: what the system asserts about the law is right. Availability is a sixth, and its failures should degrade into refusal.

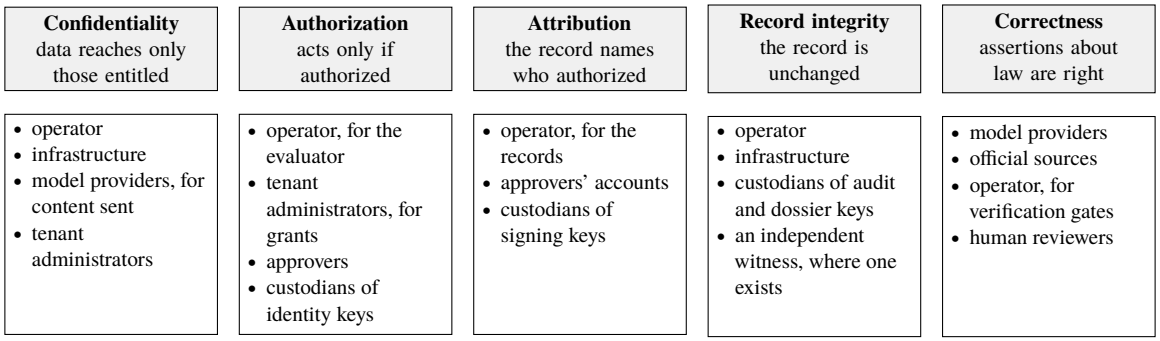

Figure 1: Properties and the parties each depends on at the basis commit. The operator appears in every column.

One feature of the figure governs the rest of this paper: the operator appears in every column. A user of a platform operated this way trusts its operator for confidentiality, for the integrity of the evaluator and of the records, and for the gates that check the model's assertions. What a third party can verify without that trust is the subject of section 6.

The design goal is that failures degrade into refusal, never into an unauthorized act or a false record. Several Lawie components implement it at the basis commit: the agent orchestrator refuses requests when it

has no verification key [TESTED], and the model gateway dispatches nothing when it cannot read the processor-agreement registry [TESTED]. The verification pipeline's production configuration defaults to fail-closed; its non-production default lets an unavailable model service count as a stage not run rather than a failure [IMPLEMENTED], under which an unsupported claim carrying a citation string was released with high confidence [MEASURED].

3 Adversaries

The trust register in section 4 is organised by party. Table 1 organises the same analysis by adversary: who might act against a user of the platform, with what capability, which properties are at stake, and where the register treats the case. The list is a threat model for any platform that executes legal work through agents; it asserts nothing about the frequency of any attack.

Table 1: Adversaries considered, their capabilities and the properties at stake.

Adversary	Capability	Properties at stake	Entries
External attacker	Reaches public endpoints without valid credentials	Confidentiality, availability	T-03, T-04
Careless or malicious tenant user	Holds a valid session; starts runs and approves within the permissions granted	Authorization, attribution	T-05, T-06, T-07
Author of untrusted content	Writes documents, e-mails or pages that an agent reads	Authorization	T-07
Misbehaving model	Proposes any act and asserts anything, fluently	Correctness, authorization	T-01, T-07
Insider at the operator	Holds operator access to systems and records	Confidentiality, record integrity	T-03
Holder of a stolen key	Signs or asserts as the key's owner until the key is revoked	Attribution, record integrity	T-08
Over-reading recipient	Treats verified integrity as proof of truth or admissibility	Correctness of what is inferred	T-09

Three assumptions underlie every entry. The cryptographic primitives hold: SHA-256 is collision resistant, and Ed25519, ECDSA P-256 and RSA-PSS signatures cannot be forged without the private key [1]. Keys are bound to the parties they are said to belong to, which is a matter of custody and issuance, not of cryptography. And the platform's own software does what its source says, which is the operator's responsibility (T-03). An adversary who breaks any of the three defeats the controls that rest on it; the register states which controls those are.

Two features distinguish agent platforms from conventional software in this model. The author of untrusted content is an adversary who needs no account: the agent carries the attack inside by reading it [2]. And the model is an adversary by construction, not because it is malicious, but because nothing it proposes may be trusted for authority (section 1); every consequential act must therefore be authorized by something other than the model's output.

4 Trust assumption register

Each entry states what the party is trusted for, how trust fails, how the failure would be noticed, what limits it and what remains. Evidence labels follow the claim register.

4.1 Sources of content

T-01 Model providers. *Trusted for* the confidentiality of content sent to them, under their contracts, and for availability. *Not trusted for* correctness or authority: model output is never an input to the authority of a statute version [TESTED]. *Failure*: a wrong assertion (a fabricated citation, a misquotation, a provision cited for the wrong date); disclosure or retention of content sent; unavailability. *Detection*: wrong assertions meet the output gates. An agent output whose factuality or citation-accuracy metric is missing or failing is withheld [TESTED], and a research answer with a citation not backed by a retrieved source is withheld [TESTED]. Disclosure by a provider is not detectable by Lawie. *Limits*: the model gateway refuses to send content classified privileged or secret to a hosted model unless the tenant has an active processor agreement or the call is routed to an in-boundary model, and sends nothing when the agreement registry cannot be read [TESTED]. *Residual*: confidential content that is not classified privileged or secret goes to hosted models unchanged, so its confidentiality rests on the providers' contracts; ABA Formal Opinion 512 discusses lawyers' duty of confidentiality for client information entered into such tools [3]. The automatic checks have measured blind spots: Lawie's lexical grounding accepted every decisive edit tested [MEASURED], its reported confidence carried no information [MEASURED], and its citation-required rule passed fabricated citations [MEASURED] (LL-REL-01, LL-BEN-01). Correctness rests on human review.

T-02 Official sources. *Trusted for* the authenticity and currency of the legal texts Lawie cites. *Failure*: an erroneous text published by the source, an outage, a change of format. *Detection*: each source observation records its locator, retrieval time, payload and content hashes, a snapshot, connector and parser versions, licence revision and an immutable supersession chain [IMPLEMENTED], so a change at the source appears as a new observation. An error in the official text itself is not detectable: by construction it is the reference. *Limits*: a statute version is authoritative only when its text is an exact byte span of an official observation retrieved under an approved, active source contract, still current and in force at the citation's date. *Residual*: one span derivation is admitted, for Czech statute paragraphs [IMPLEMENTED], and the production connector profile has no admitted source because no source contract is approved [BLOCKED]; until then nothing can be verified as authoritative in production.

4.2 The operator and its infrastructure

T-03 The operator. *Trusted for* the confidentiality of tenant data, the integrity of the software and configuration that decide authorization, and the integrity of the records. *Failure*: an operator, an insider or an attacker with operator access can read tenant data, change what the evaluator decides and rewrite records. *Detection*: the agent audit chain's verifier detected every tested edit, deletion, reordering and insertion of records and every deleted prefix [MEASURED]; evidence that holds against the operator itself needs a party the operator does not control (section 6). *Limits*: in the application, tenant scoping compares the requested tenant with the authenticated tenant, for administrators too [TESTED]; it is a control on application requests, not on the operator. *Residual*: users must trust Lawie Labs, as operator, for confidentiality and record integrity. This paper does not claim that Lawie cannot access tenant data.

T-04 Infrastructure and key-management providers. *Trusted for* availability, the confidentiality of stored data and the custody of the dossier signing key, which is held in a managed key-management service [IMPLEMENTED]. *Failure*: a compromise of hosting gives the attacker the operator's position (T-03); a compromise of the key service lets the attacker sign dossiers that verify (T-08). *Residual*: the providers are trusted as the operator is; no separation between their control domains and the operator's is claimed.

4.3 People inside the tenant

T-05 Tenant administrators. *Trusted for* the authority that every other control enforces: which tools each tenant's agents may use and with what data-class and risk ceilings, memberships and roles, and which agents are admitted. *Failure*: grants broader than intended; an administrator who acts in bad faith. *Limits*: grants deny by default and are checked with their ceilings, the run's actor and the agent's admission on every call [TESTED], against current state [IMPLEMENTED]; the requested tenant is compared with the authenticated tenant, for administrators too [TESTED]. *Residual*: as in any tenant model, the controls described do not limit an administrator acting in bad faith inside their own tenant.

T-06 Approvers. *Trusted for* judgment over the exact act they approve. *Failure*: approving without reading; colluding; approving an act induced by content the reviewer cannot see. *Limits*: an approval binds the tool, its exact version and the digest of the current input [TESTED]; it is usable only if approved by a named reviewer after it was requested, before use and before expiry [TESTED]; the reviewer must be an authenticated human holding the approval permission, and service calls and API keys are refused [IMPLEMENTED]; the reviewer's permission is re-read when the approval is used [IMPLEMENTED]; on provider dispatch each approval is consumed once [IMPLEMENTED]. *Residual*: an approval is as good as the reviewer's attention and independence; colluding reviewers defeat any separation (LL-GMA-01).

T-07 Requesting users and their agents. *Users are trusted for* their intent; *agents for nothing*. Every call an agent proposes passes the effective mode, which is the most restrictive of all applicable rules [TESTED], and the runtime requires an exact-input approval before any privileged action in every run [TESTED]. *Failure*: a stolen session. *Limits*: requests to the agent orchestrator carry a signed identity assertion bound to method, path and body, single-use when they change state, with tenant or user substitution rejected [TESTED]. *Residual*: a signed identity establishes which session made a request, not why it was made.

4.4 Keys and those who verify

T-08 Key custodians. Two kinds of key carry authority. *Dossier signing keys* are asymmetric and held in a key-management service [IMPLEMENTED]; whoever controls a trusted signing key can sign dossiers that verify. *Request-identity keys*: requests to the orchestrator carry signed identity assertions, so whoever holds a private key that the orchestrator trusts can issue assertions it accepts, within what that key may assert. *Residual*: no hardware custody, key ceremony or rotation evidence is claimed in this paper.

T-09 Recipients and verifiers of exported evidence. *Trust* their own trust policy, the dossier format and the operator's signing key. *Obtain*: offline verification against pinned key fingerprints for Ed25519, ECDSA P-256 and RSA-PSS signatures [TESTED]; in the suite's trials the verifier failed every tampered dossier and verified the intact one, also after its disclosure authorization and its key's validity period had ended [MEASURED]. *Residual*: verification establishes that the bytes are those the key holder signed; the dossier itself states that verification does not prove substantive truth, admissibility or legal correctness and creates no authority [IMPLEMENTED]. Dossiers cover institutional execution, not agent runs.

5 Controls

Table 2: Controls, their implemented behaviour at the basis commit (not an assurance of a deployed service), and what no control of their kind establishes. Evidence labels are those of the statements in the row.

Control	Implemented behaviour at the basis commit	Not established by a control of this kind	Evidence
Signed request identity	Which tenant and user a request to the agent orchestrator carries, bound to method, path and body; single use when state changes; refusal without keys	That the principal intended the act	[TESTED] [TESTED]
Tenant scoping	The requested tenant is compared with the authenticated tenant, for administrators too	Confidentiality against the operator or its infrastructure (T-03, T-04)	[TESTED]
Default-deny grants	No tool without an active grant; ceilings, actor and admission checked against current state	That the authority granted is wise	[TESTED] [IMPLEMENTED]
Approval binding and validity	What executes is the tool, version and input a named human approved, in time; on provider dispatch, each approval is consumed once	That the reviewer attended to the act	[TESTED] [TESTED] [IMPLEMENTED] [IMPLEMENTED]
Audit chain	Detection, by its verifier, of every tested edit, deletion, reordering and insertion of records and of a deleted prefix	Evidence against the chain's keeper, which no self-kept record provides	[MEASURED]
Signed dossiers	Offline integrity and origin under pinned keys; every tested tampered dossier failed	Truth, admissibility or authority	[TESTED] [IMPLEMENTED] [MEASURED]
Model data boundary	Privileged or secret content reaches a hosted model only under an active processor agreement or an in-boundary route; nothing is sent when the registry cannot be read	Confidentiality of other content, which rests on providers' contracts	[TESTED]

Table 2 lists the controls that carry the trust analysis. The subsections give their measured outcomes.

5.1 Approvals and authorization

An approval binds the tool, its exact version and the digest of the current input [TESTED]; it is usable only if approved by a named reviewer after it was requested, before use and before expiry, and a malformed composite policy is invalid [TESTED]. Approval decisions come from an authenticated human holding the approval permission; service calls and API keys are refused [IMPLEMENTED], and the grant decision re-reads current state, including the reviewer's permission, on every call [IMPLEMENTED].

5.2 Records and dossiers

In the suite's evidence-integrity study the audit chain's verifier detected every tested edit, deletion, reordering and insertion of records and every deleted prefix, in 50 trials each, and the portable dossier verifier failed every tampered dossier [MEASURED] (table 3). Signed portable dossiers verify offline against pinned key fingerprints [TESTED]; the verifier also verified the intact dossier after its disclosure authorization and its signing key's validity had ended [MEASURED], which is what an archive needs. A dossier's manifest states that verification does not prove substantive truth, admissibility or legal correctness and creates no authority [IMPLEMENTED].

Table 3: Agent audit chain: tested mutations detected by the chain verifier, 50 trials each [MEASURED].

Mutation	Trials	Detected by the chain verifier
Edit a record	50	50
Delete a record	50	50
Swap two records	50	50
Insert a forged record	50	50
Delete a prefix	50	50

Two general principles bound what any such record can show. A hash chain proves internal consistency to whoever holds it; evidence that holds against its keeper needs a party the keeper does not control, such as a witness of the chain's head [4, 5]. A signature proves origin under a key. The suite's evidence paper, LL-EVP-01, treats chains, anchoring and dossiers in depth.

5.3 Fail-closed behaviour

The agent orchestrator refuses requests when it has no verification key, and state-changing requests when it has no replay store [TESTED]. The model gateway dispatches nothing when it cannot read the processor-agreement registry [TESTED]. The verification pipeline's production configuration defaults to fail-closed; outside production its default treats an unavailable model service as a stage not run, which does not fail the output [IMPLEMENTED]. Under that default the suite observed the release, with high confidence, of a claim that had no retrieved source but carried a citation string; under fail-closed all four tested configurations were blocked [MEASURED]. Behaviour seen in development and testing should not be taken as evidence of production behaviour.

6 Accountability

Lawie's stated vision ends with a test: "here is who authorized it, on what evidence, with what result — verify it yourself" [IMPLEMENTED]. This section states the test and what Lawie records against it.

Definition 6.1 (Third-party accountability)**.** An act is *accountable to a third party* if a party that does not trust the operator can determine, from an exported record and trust anchors of its own choosing, (i) which principals requested the act and which named persons authorized it, (ii) on what inputs and evidence it was authorized, (iii) what result it produced, and (iv) whether any of this has been altered since it was recorded.

The definition excludes trust in the operator on purpose. Accountability to the operator's own staff is a matter of internal control; accountability to a court, a regulator, an auditor or a client requires that the record be convincing to someone the operator cannot influence. Frameworks for internal algorithmic audit close part of the accountability gap from inside an organisation and are meant to complement external scrutiny [6]. Article 12(1) of the AI Act, as published, requires that high-risk AI systems "shall technically allow for the automatic recording of events (logs) over the lifetime of the system" [7]. Recording is the keeper's act; whether a record proves anything to a third party depends on whether the keeper can rewrite it undetected. This paper makes no statement that any Lawie system is high-risk or that it meets that or any other provision; the company's claims policy prohibits unqualified claims of compliance with the AI Act [IMPLEMENTED].

Table 4: What Lawie records or checks for each question of definition 6.1.

Question	What Lawie records or checks	Evidence
Who requested	The acting user of the run, checked against current state on every call	[IMPLEMENTED]
Who authorized	A named reviewer, who must be an authenticated human holding the approval permission	[IMPLEMENTED] [TESTED]
On what input	The SHA-256 digest of the approved input, recomputed from the input proposed	[IMPLEMENTED]
On what evidence	Source observations with locator, retrieval time, payload and content hashes, connector and parser versions and a supersession chain	[IMPLEMENTED]
With what result	Signed portable dossiers for institutional execution, verifiable offline	[TESTED]
Unaltered	A hash-chained audit record whose verifier detected every tested in-place edit, deletion, reordering, insertion and deleted prefix	[MEASURED]

6.1 What Lawie records

Table 4 lists what Lawie records or checks for each question of the definition at the basis commit.

A record convinces a third party to the extent that it does not depend on trusting whoever keeps it. A dossier shows that a trusted key signed it [TESTED]; an audit chain shows its internal consistency to whoever holds the whole of it [MEASURED]. Whether either shows anything against its keeper depends on the properties below.

6.2 What third-party verification requires

Five properties make the answer independent of the operator. Each is a principle for any platform, stated with its cost.

1. **Approvals signed by the approver.** A key held by the reviewer signs the digest of the act. A third party then verifies who authorized the act without trusting the operator, provided it trusts whoever bound the key to the person. *Cost:* key issuance and recovery for every reviewer.
2. **A standard encoding of the act.** The approved digest is computed over a canonical serialization that any party can reproduce, such as RFC 8785 [8]. *Cost:* every party must implement the canonicalization identically.
3. **Complete records.** Records name the requester lineage, the reviewers relied on, the model, prompt and policy versions and the provenance of the request, so that each question of the definition is answered from the record itself. *Cost:* more sensitive metadata to protect.
4. **Independent witnessing.** The head of each chain is periodically committed to a party the operator does not control, as in tamper-evident logs with signed tree heads and consistency proofs [4, 5]; changes made before the last witnessed head then become detectable by a third party [ANALYTICAL]. *Cost:* a witness and a disclosure policy for what it sees.
5. **Trusted time.** Signatures are bound to a time attested by an independent time-stamping service [9, 10], and key validity and revocation are taken from the verifier's trust policy or a status source the producer does not control. *Cost:* a further external dependency.

Even with all five, accountability would rest on assumptions: that keys are bound to the right persons, that the witness is independent, and that the record says what happened rather than what was recorded. A verified record establishes integrity and origin, not truth; Lawie's dossiers say so themselves [IMPLEMENTED].

7 What the evidence establishes, and what it does not

Table 5: Kinds of evidence behind this paper and their limits.

Evidence	What it establishes	What it does not
Source review at the basis commit	What a check does when it runs	That every path calls it; how it behaves when deployed
Repository tests	Specified behaviour on the tested inputs	Behaviour outside the tested inputs; a property nobody wrote down
BEN-02 (component level, synthetic, offline, run once)	What three authorization functions decide on 552 stated scenarios [MEASURED]	End-to-end behaviour; the grant decision and the policy engine; any real tenant
EVD trials (offline, synthetic)	What the chain verifier and the dossier verifier detect for stated mutations [MEASURED]	Behaviour against an adversary who controls the records' keeper
Analytical results	What follows from the stated assumptions [ANALYTICAL]	That the assumptions hold in a deployment
Independent audit or certification	None performed [ABSENT]	—

In the BEN-02 benchmark three of Lawie's authorization functions denied none of the 96 legitimate requests, refused every scenario of some attack classes and allowed every scenario of others [MEASURED]. Which classes these are is recorded in restricted engineering records pending remediation.

Every measurement in this paper was made by Lawie Labs on its own code, offline, on synthetic or public data. None was made on a deployed service or on real legal matters, since nothing described here is in production [ABSENT]. The benchmark designers knew the code they tested, which favours finding the gaps they suspected and says nothing about gaps they did not; LL-BEN-02 discusses this.

8 What this paper does not claim

The company's claims policy prohibits several unqualified claims, among them "tamper-proof", "zero hallucinations" and "AI Act compliant" [IMPLEMENTED]. For security and accountability the following non-claims apply.

Not audited, not certified. No independent audit, certification or review by an independent expert of the components described is recorded [ABSENT]. In particular, no certification under a management-system standard such as ISO/IEC 42001 [11] is claimed.

Not tamper-proof. We do not call any record tamper-proof. The evidence is of tamper detection for the mutations tested [MEASURED], under the conditions of section 5.

Not "Lawie cannot access data". The operator can access tenant data (T-03).

Not "zero retention". Content that is not classified privileged or secret is sent to hosted models [TESTED]; its retention is governed by the providers' contracts, which this paper does not assess.

Not compliance. Nothing here states that Lawie, or any use of it, complies with the AI Act or any other law.

Not admissibility. A verified dossier establishes integrity and origin under the signing key; its own manifest says it does not prove substantive truth, admissibility or legal correctness and creates no authority [IMPLEMENTED].

Not production. No product or workflow described here has a dated production acceptance [ABSENT], and production is not deployed [BLOCKED].

9 Assumptions and the properties that retire them

A user of any platform that executes acts for others makes the assumptions below unless the stated property holds. These are principles for any such platform, not statements about a particular one.

1. *The operator does not rewrite records.* Retired by independent witnessing of chain heads and by approvals signed with the approvers' own keys (section 6).
2. *Agents act on their principals' requests, not on content they read.* Retired only by carrying provenance from content to proposals and refusing or escalating content-influenced consequential acts.
3. *A sub-agent does no more than its parent may.* Retired by intersecting the child's authority with the parent's at every delegation.
4. *A tool input designates no tenant other than the requester's.* Retired by tools that take the tenant only from the authenticated principal.
5. *Model providers keep confidential content confidential.* Rests on contracts and on the classification of what is sent.
6. *Official sources publish the right text.* Inherent: the official source is the reference by construction.

10 Guidance for evaluators

Treat an approval as evidence that a named person approved the exact act. Treat audit records as the keeper's statements until their head is witnessed by someone else. Verify dossiers with a trust policy you control, read their limitations. Treat a refusal as an answer, and treat a development environment's behaviour as no evidence of production behaviour. Ask, for every act, the question of definition 6.1, and accept as an answer only what you can verify without the operator.

References

[1] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed High-security Signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, 2012. doi: 10.1007/s13389-012-0027-1.

[2] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90, 2023. doi: 10.1145/3605764.3623985.

[3] American Bar Association Standing Committee on Ethics and Professional Responsibility. Formal Opinion 512: Generative Artificial Intelligence Tools, 2024. URL https://www.americanbar.org/content/dam/aba/administrative/professional_responsibility/ethics-opinions/aba-formal-opinion-512.pdf. 29 July 2024.

[4] Scott A. Crosby and Dan S. Wallach. Efficient Data Structures for Tamper-Evident Logging. In *Proceedings of the 18th USENIX Security Symposium*, pages 317–334, 2009. URL https://www.usenix.org/conference/usenixsecurity09/technical-sessions/presentation/efficient-data-structures-tamper-evident.

[5] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, Internet Engineering Task Force, 2013.

[6] Inioluwa Deborah Raji, Andrew Smart, Rebecca N. White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 33–44, 2020. doi: 10.1145/3351095.3372873.

[7] European Parliament and Council of the European Union. Regulation (EU) 2024/1689 of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union, L, 2024/1689, 12.7.2024, 2024. URL https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32024R1689.

[8] Anders Rundgren, Bret Jordan, and Samuel Erdtman. JSON Canonicalization Scheme (JCS). RFC 8785, Internet Engineering Task Force, 2020.

[9] Stuart Haber and W. Scott Stornetta. How to Time-Stamp a Digital Document. *Journal of Cryptology*, 3(2): 99–111, 1991. doi: 10.1007/BF00196791.

[10] Carlisle Adams, Pat Cain, Denis Pinkas, and Robert Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161, Internet Engineering Task Force, 2001.

[11] ISO/IEC. ISO/IEC 42001:2023 Information technology — Artificial intelligence — Management system. International Organization for Standardization, 2023. URL https://www.iso.org/standard/42001.