Evidence, Provenance and Legal Reasoning

What integrity mechanisms can establish about a legal statement, and what they cannot

Research paper LL-EVP-01
Version 1.0.0, 26 November 2026

Abstract

A legal statement produced by a machine is worth what its evidence is worth. In legal reasoning that evidence is a chain: an authentic text, retrieved from an official source under a known right to use it; an exact span of that text; a proposition derived from the span; a person who reviewed the derivation; and an artifact that carries all of this to whoever must rely on it. Integrity mechanisms (hashes, hash chains, keyed chains, signatures, Merkle trees, transparency logs) protect some links of that chain and are routinely presented as protecting all of it. This paper separates the links. It defines a provision-level provenance model, states for each link what can establish it and what cannot, and proves what an unkeyed hash chain detects: any edit, deletion, insertion or reordering that leaves an original hash after the point of change, but no truncation of its tail and no rewriting by anyone able to recompute the hashes. Detecting either requires an independent witness of the chain's head [ANALYTICAL]. It shows why a signature on evidence must be judged at the time it was made, not when it is checked, and why that rule is only as strong as the independent evidence of the signing time and of the key's validity.

In the suite's evidence-integrity experiment, Lawie's audit-chain verifier detected every tested edit, deletion, reordering, insertion and deleted prefix, and its portable dossier verifier failed every tampered dossier [MEASURED]. The paper sets a standard for reconstructing an automated decision from its record and states ten requirements for a legal evidence chain. Integrity is not truth. The principle that runs through the paper is that the limitations of every link travel with the artifact.

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:* legal technologists, security engineers, senior lawyers, regulators and researchers who must decide what a machine-produced legal record can be relied on for.

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.

Position

Intelligence is not authority. A model that writes a fluent legal answer has shown competence. It has not shown that the answer rests on the law, and nothing it says about its own sources counts as evidence that it does. Authority in legal work comes from what a hostile reader can check: the authentic text, the exact words relied on, who reviewed the reading, and a record that has not been altered since. This paper is about the last three of those, and about the limits of what checking can establish.

Six positions

1. **A citation is an address, not evidence.** Evidence is an exact span of an authentic text retrieved under an admitted source, which a reader can compare with the statement that relies on it. A reference string names where evidence might be found. *Cost:* a reference that only resolves to an existing, similar-looking source does not qualify, and in Lawie no official source is yet admitted [BLOCKED].
2. **Integrity is not truth.** Hashes and signatures establish that bytes are unchanged since a point in time and which key signed them. They do not establish that a proposition follows from its source, that it is right in law, or that a court would admit it. *Cost:* every integrity mechanism must be paired with a statement of what it does not establish and with a separate check of support.
3. **A chain without a witness proves only its own consistency.** Whoever can recompute the hashes can rewrite the past, and whoever can cut the tail can erase the present. *Cost:* someone the operator cannot instruct must hold the chain's heads, and the interval between two witnessings is an exposure window that must be stated with the evidence.
4. **Evidence outlives keys.** A signature is judged by the facts at the time it was made, not at the time it is checked; a rule that demands a currently valid key makes evidence expire on a rotation schedule. *Cost:* the time of signing must itself be witnessed, or the rule protects backdating.
5. **A decision that cannot be reconstructed cannot be defended.** The record of a consequential step must name its inputs, sources and their versions, the model and instructions that produced it, the checks it passed and the approval it ran under. Re-running a stochastic model is not reconstruction. *Cost:* larger records, and obligations of confidentiality and retention over them.
6. **Limitations travel with the artifact.** What a verification does not establish belongs inside the signed artifact, where a verifier must carry it, not in a disclaimer elsewhere. *Cost:* heavier artifacts, limitation statements that must be maintained, and recipients who may still ignore them.

Where it stands

Lawie implements several links of the chain in source: source observations with an immutable supersession chain [IMPLEMENTED], an authority predicate that accepts only exact byte spans of admitted official text [TESTED], approvals bound to the exact input they authorise [TESTED], and signed dossiers that verify offline and carry their own limits [TESTED]. Only one span derivation, for Czech statutes, is admitted [IMPLEMENTED]. The requirements of section 8 are stated for any system and are not a verdict on Lawie. Nothing described here is deployed in production [BLOCKED], no workflow has a dated production acceptance [ABSENT], and no audit or independent expert review of these components exists [ABSENT].

Contents

Abstract **1**

Position **2**

1 Introduction **4**

2 What evidence is in legal reasoning **4**

2.1 Authentic text, copies and statements 4

2.2 Seven properties that are routinely conflated 5

2.3 Why the separation is operational 5

3 A provenance model for a legal proposition **6**

3.1 The chain 6

3.2 Observation 6

3.3 Admission 7

3.4 Span 7

3.5 Canonicalisation: two purposes that must not be mixed 7

3.6 Custody and supersession 8

3.7 Derivation 8

3.8 Review 9

3.9 Artifact 9

4 Chains and witnesses **9**

4.1 The model 9

4.2 Measurement 10

4.3 Keyed chains and retention 11

4.4 Witnesses: Merkle trees and transparency logs 11

5 Signatures, keys and time **12**

5.1 What a signature establishes 12

5.2 Measurement 12

5.3 Why verification after key expiry is correct 12

5.4 The condition: a witnessed time of signing 13

6 Reconstructing an agent decision **13**

6.1 A standard 13

6.2 What the standard requires of a record 14

7 Limitations travel with the artifact **14**

8 Requirements and open problems **15**

8.1 Requirements for a legal evidence chain 15

8.2 Open problems 16

9 Evidence boundary of this paper **16**

References **17**

1 Introduction

Legal work runs on statements that someone else will rely on: that a provision says something, that it applies to a set of facts, that a step was approved by the person entitled to approve it. When a machine produces such statements, fluency is not the question. The question is what a sceptical reader, with access to the record and to the law, can check, and what checking can and cannot establish. This paper answers that question for the mechanisms that legal software uses to make its output checkable.

Two errors recur. The first treats a citation as evidence. An answer that names a provision is presented as grounded, although the name is an address and nothing has established that the provision says what the answer says. Sanctions for fictitious case citations in court filings [1] and the professional guidance that followed [2] concern the crude form of this error; its refined form survives every retrieval pipeline that checks only that a cited source exists and resembles the answer. The second error treats integrity as truth. A record that is hashed, chained or signed is presented as reliable, although these mechanisms establish only that bytes have not changed since some moment and which key signed them. Both errors convert a narrow, checkable property into a broad, uncheckable one.

This paper takes the opposite route. It separates the links of the chain that connects an authentic legal text to a relied-on statement, states for each link what can establish it, and reports measurements of Lawie's mechanisms where they apply. Lawie's stated mission is to make high-stakes institutional work executable by supervised machines, completed, approved, evidenced and defensible [IMPLEMENTED]; "evidenced" and "defensible" are the subject of this paper, and the standard applied to Lawie is the one it would have to meet in front of a hostile reader.

Method. Statements about Lawie describe its source repository at the basis commit and carry an evidence label from the suite's claim register. Analytical results are stated as propositions with their assumptions and proved. Empirical results come from offline experiments of this suite, which run Lawie's components unchanged on synthetic records and public legal texts; none is a production or customer measurement.

Scope. The paper concerns the evidence for statements about legislation and for the steps of automated legal work. It does not treat the evidential value of documents in litigation, rules of admissibility, or electronic signature law; nothing here asserts that any record would be admitted by any court. Companion papers treat the support relation between a statement and its source (LL-REL-01, LL-BEN-01), the time at which a provision applies (LL-TMP-01) and the authorization of agent actions (LL-BEN-02, LL-SEC-01).

Outline. Section 2 distinguishes the properties that integrity mechanisms are confused with. Section 3 defines a provenance model for a single legal proposition. Section 4 proves what hash chains detect and reports the measurement. Section 5 treats signatures and time. Section 6 sets a standard for reconstructing an automated decision from its record. Section 7 states the principle that limitations travel with the artifact, section 8 the requirements and open problems, and section 9 the evidence boundary of this paper.

2 What evidence is in legal reasoning

2.1 Authentic text, copies and statements

A statement about the law draws its authority from two things: the text that the legal order designates as authentic, and the reasoning that connects that text to the statement. For legislation of the European Union the authentic text is the one published in the Official Journal. Everything else is a copy or a derivative: a database record, a search index entry, a consolidated version, a model's recollection. EUR-Lex says so on the face of its consolidated version of the AI Act, which opens: "This text is meant purely as a documentation tool and has no legal effect." It adds that the authentic versions are those published in the

Official Journal [3][EXTERNAL]. A chain of legal evidence therefore begins at an authentic publication, and every copy in it is evidence only to the extent that its correspondence with that publication can be checked.

A statement that relies on a provision stands in three distinct relations to it. *Identity*: the words the statement attributes to the provision are the provision's words. *Support*: what the statement asserts follows from those words. *Correctness*: the statement is right in law, all things considered, including other provisions, the date, the facts and the case law. Identity can be checked mechanically. Support can be checked mechanically only for narrow classes of statements, of which verbatim quotation is the clearest. Correctness is a legal judgment, and the doctrine of this suite is that a machine supports it and does not make it.

2.2 Seven properties that are routinely conflated

Table 1 separates the properties that a legal evidence chain must keep apart. The point of the table is its last column: each mechanism that establishes one property is routinely offered as evidence of another. A valid signature is presented as proof that a document is correct. A retrieved source is presented as support for whatever was written next to it. A hash chain is presented as a complete record. These are category errors, and an adversary, a careless vendor or an overconfident user exploits each of them.

Table 1: Properties of a legal record and what can establish them.

Property	Question it answers	What can establish it	What it does not establish
Integrity	Are these the bytes that were recorded?	A collision-resistant hash compared with a value held independently of the record	Who wrote the bytes; whether they are true or complete
Attribution	Who made this statement?	A signature under a key whose binding to a person or body is independently trusted	That the statement is accurate or authorised
Time	When did the record exist?	A witnessed timestamp or an anchored commitment	Anything about its content
Provenance	From what, by which steps and agents, was it derived?	Records of observations, transformations and responsible agents	That any step was performed correctly
Source authority	Is the source the authentic text, used under a right?	Retrieval from the official publisher under an approved source contract	That the text was read correctly
Support	Does the statement follow from the cited words?	Exact match for quotations; typed comparison for modelled elements; review otherwise	That the statement is right in law overall
Correctness and reliance	Is it right, and may it be relied on?	Professional judgment by an accountable person	Nothing mechanical establishes it

2.3 Why the separation is operational

The separation decides outcomes in practice. In the pre-registered LLRB-1 benchmark, Lawie's grounding check, as implemented at the code basis, accepted every one of 1,959 statements carrying a single legally decisive edit, and every one of 698 statements cited to the wrong provision [MEASURED]. In each case the citation resolved, the source was retrieved, and the source text was intact; what failed was the support relation. Wrapping such an output in a hash chain, a signature and a transparency log would produce an artifact that verifies perfectly and carries a false statement about the law. The same benchmark shows the reverse limitation: exact-span checking accepted every authentic quotation and rejected every edit [MEASURED], and canonical matching did the same while also accepting typographic variants [MEASURED], but

neither can judge a correct paraphrase. Integrity at the source end cannot repair a weak derivation, and a strong check of identity is not a check of support.

3 A provenance model for a legal proposition

3.1 The chain

We describe provenance in the vocabulary of the W3C provenance data model [4]: *entities* (an observation, a span, a proposition, a review record, an artifact), *activities* that generate and use them (retrieval, admission, extraction, derivation, review, signing) and *agents* responsible for the activities (a connector and its parser, a checking procedure, a model, a reviewer, a signer). The data model also supplies the three derivation subtypes the legal case needs: a *quotation* (a span copied from a source), a *revision* (a later version of the same record) and a *primary source* (the authentic publication from which the rest derives), and the relation of *invalidation*, which marks the end of an entity's usability, for example when an observation is superseded.

Definition 3.1 (Legal evidence chain)**.** Let φ be a proposition about the law. A *legal evidence chain* for φ is a tuple $(O, A, \sigma, \kappa, D, R, X)$ in which

(i) O is a *source observation*: a retrieved copy of an official record with its locator, retrieval time, payload hash and the versions of the software that retrieved and parsed it;

(ii) A is its *admission*: the source contract under which O was retrieved, naming the official source, its jurisdiction, the licence under which it may be used, the approval of the contract and its period of validity;

(iii) $\sigma = (b, e, \mathcal{H}(\text{payload}_{[b,e)}))$ is a *span*: byte offsets into the payload of O and the hash of the bytes between them;

(iv) κ is the declared *canonicalisation* under which the words φ attributes to the source are compared with σ, if any;

(v) D is the *derivation*: the assertion that $\mathsf{Supp}(\sigma, \varphi)$ holds, with the procedure that checked it and that procedure's measured error rates;

(vi) R is the *review*: a named person's decision on D, bound to $\mathcal{H}(\varphi)$ and to the hash of σ, with its time;

(vii) X is the *artifact* that carries $O, \ldots, R$, or commitments to them, to a relying party, together with the limitations of each link.

The chain *verifies* if every hash in it recomputes and every signature verifies under a trusted key. It *supports* φ only if D is true. It *authorises reliance* only if R was made by a person entitled to decide.

The three predicates are independent. A chain can verify while D is false: a perfectly preserved record of a misreading. D can be true while the chain fails to verify: a correct reading whose record was altered. And a verified, supported chain authorises nothing if the reviewer had no standing. Figure 1 shows the chain and marks the three regions in which these predicates live.

3.2 Observation

An observation is the moment a copy of an official record enters the system. Lawie records for each one its locator, retrieval time, payload and content hashes, an object snapshot, the connector and parser versions, the licence revision and an immutable supersession chain [IMPLEMENTED]. These fields answer the questions a dispute about a source will ask: what exactly was retrieved, from where, when, by which software, under which licence, and what replaced it later. The payload hash turns a later disagreement about what the source said at retrieval into a comparison of bytes. An observation remains a copy. Its standing comes from the next two links.

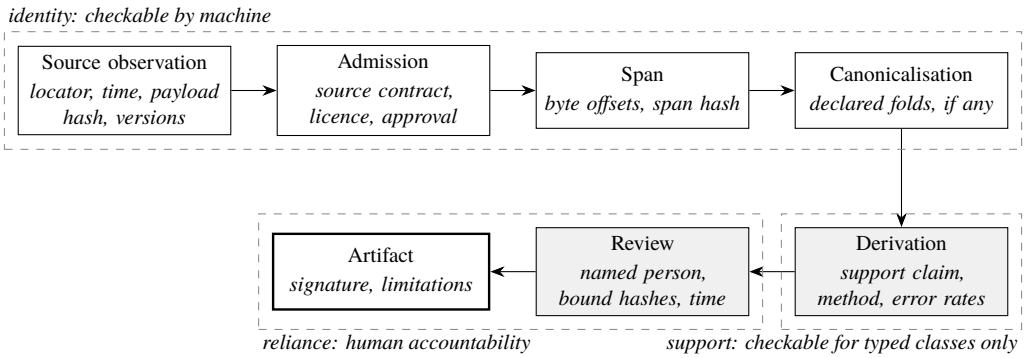

Figure 1: The legal evidence chain of definition 3.1. Hashes and signatures can protect every box; they establish the contents of the top row, not the truth of the derivation or the standing of the reviewer.

3.3 Admission

Admission is the decision that a source may be relied on at all: that it is the official publisher of the text, that the system may use it under its licence, and that someone accountable approved the arrangement. Lawie's authority predicate treats a statute version as authoritative only when its text is an exact byte span of an official-source observation retrieved under an approved, active source contract, the observation is still the current state of the official record, and the version is in force at the citation's date; model output is never an input, and every refusal carries a typed reason [TESTED]. The predicate is strict, and the strictness has a visible cost: no official source is yet admitted [BLOCKED], so no statute text can be authoritative in Lawie today. We regard this as the correct failure mode. A system that cannot admit its sources should say so and refuse, not fall back to an unadmitted copy and label the result verified.

3.4 Span

The span is the quotation in the provenance sense: the bytes of the observation on which the statement relies, located by offsets and fixed by a hash. Computing offsets requires a *span derivation*, a procedure that maps the structure of one official format (article, paragraph, point) to byte positions in its payload. Lawie admits exactly one derivation, for paragraphs of the Czech official collection; no EU, Slovak or case-law text can currently be verified as authoritative [IMPLEMENTED]. The benchmark shows what this means in operation: with the derivation that EU texts would need, exact-span authority rejected every statement, authentic quotations included [MEASURED]. Each official format needs its own reviewed derivation before any quotation from it can be verified; there is no general shortcut, because the mapping from legal structure to bytes is a property of the publisher's format, not of the law.

3.5 Canonicalisation: two purposes that must not be mixed

"Canonicalisation" names two operations with opposite requirements.

Integrity canonicalisation fixes one byte representation of a structured record before it is hashed or signed, so that two parties who hold the same record compute the same hash. The JSON Canonicalization Scheme sorts object keys, removes insignificant white space and fixes the serialisation of numbers and strings [5]. It must be injective on content: it may choose a representation, never alter a value. Lawie hashes structured legal records over a serialisation of this kind, with recursively sorted keys and no insignificant white space; it rejects non-finite numbers and applies no Unicode normalisation to strings [IMPLEMENTED]. Two strings that look identical but differ in code points therefore hash differently, which is right: the hash certifies bytes, and a normalisation applied before hashing would make the hash certify a text that was never published.

Comparison canonicalisation folds typographic variation (no-break spaces against spaces, curly against straight quotation marks, dash variants, letter case, final punctuation) before a quotation is compared with its source. It is deliberately not injective.

Proposition 3.1 (Folding and exactness)**.** *Let κ map every character of a declared set of disjoint classes to a fixed member of its class and leave every other character unchanged. Call q a* variant *of t if the two have the same length and, wherever they differ, both characters belong to the same class. A byte-exact check accepts a quotation q against a source text s only if q is a substring of s, so it accepts no altered quotation and rejects every typographic variant. The check of $\kappa(q)$ against $\kappa(s)$ accepts q exactly when q is a variant of a substring of s; it therefore rejects every alteration that changes, inserts or deletes a character outside the classes, unless the altered quotation happens to be a variant of some other part of s [ANALYTICAL]. The benchmark's canonicaliser also collapses white space, removes final punctuation and applies compatibility normalisation; the proposition idealises it as length-preserving.*

Proof. Byte equality accepts only identical strings, which excludes edits and variants alike. The map κ acts character by character and preserves length, so the substrings of $\kappa(s)$ are exactly the images $\kappa(t)$ of substrings t of s. Because the classes are disjoint, each is mapped to one of its own members, and every other character is fixed, $\kappa(q) = \kappa(t)$ holds exactly when q is a variant of t. □

The benchmark confirms both halves: exact spans rejected every typographic variant [MEASURED], and canonical matching accepted them while rejecting every edit and misattribution [MEASURED]. The proposition also states the price. Any alteration confined to the folded classes passes, and in legal text case and punctuation can carry meaning: a capitalised defined term, a comma that fixes the reach of a qualifier. A comparison fold must therefore be declared, justified for the corpus at hand, and recorded with every comparison that used it. The rule is simple: integrity canonicalisation never folds; comparison canonicalisation folds only declared classes; neither is applied in the other's place.

3.6 Custody and supersession

Official records change: corrigenda replace text, amendments produce new versions, consolidations are reissued, publishers move documents. A span checked against an observation proves what the source said when it was observed, not what it says now. Lawie records supersession as an immutable chain of observations [IMPLEMENTED], and its authority predicate requires that the observation still be the current state of the official record and that the version be in force at the citation's date [TESTED]. "In force" is itself a trap: a provision can be in force for years before it applies, and LL-TMP-01 shows that Lawie's in-force check treats the two as one [MEASURED]. Custody, in the narrower sense of who held the record and could have altered it, is the subject of section 4.

3.7 Derivation

The derivation is the weakest link and the one integrity mechanisms do not touch. Support is a relation between the meaning of a statement and the words of a source. LLRB-1 measured how verification components decide it. Lexical verifiers, even when calibrated to keep 95% of supported statements, still accepted 27.1% to 54.2% of statements carrying one decisive edit [MEASURED]. A typed reference prototype, which compares the decisive elements it models (deontic operator with polarity, quantities, dates, time units, cross-references), rejected every edit of those types at no measured cost to supported statements [MEASURED], and accepted 88.5% of edits of types it does not model [MEASURED]; it is a research instrument of the suite, not a Lawie component [PROPOSED]. The verification confidence that Lawie's pipeline reported was the same for every supported and unsupported statement [MEASURED]. Two consequences follow for the evidence chain.

The derivation must record the procedure that checked it and that procedure's measured error profile, because "verified" without a method is a label, not evidence. And a typed check is exact only for the types it models; its coverage is part of what the record must state.

3.8 Review

Review converts a supported proposition into one that may be relied on, and it is evidence only if it is bound to what was reviewed. Lawie's approval mechanism shows how binding can be done: an approval authorises only a tool identity, its exact version and the SHA-256 of the canonical current input [TESTED], and it is usable only if it was decided by a named reviewer after it was requested, before it is used, and before it expires [TESTED].

In general, a review is evidence of independent judgment only if the record names the reviewer, binds the hashes of what was reviewed, and is dated within the validity of what was reviewed. Lawie's Civic Guard publication path requires a reviewer other than the investigator, a publisher other than the reviewer, and published facts whose hashes match confirmed facts [IMPLEMENTED]; the product is quarantined and its public pages are not served in production [BLOCKED].

3.9 Artifact

The artifact is what leaves the system. Lawie's signed portable dossiers verify offline against pinned key fingerprints, with Ed25519, ECDSA P-256 or RSA-PSS signatures, and cover institutional execution records, not agent runs [TESTED]. Their manifest states that verification does not prove substantive truth, admissibility or legal correctness and creates no authority [IMPLEMENTED]. Section 5 examines how they behave under tampering and over time, and section 7 the way their limitations are bound into them.

4 Chains and witnesses

4.1 The model

Let $x_1, \dots, x_n$ be records, each $x_i = (id_i, t_i, m_i)$ an identifier, a creation time and a payload. An *unkeyed hash chain* stores with each record its predecessor's hash p_i and its own content hash c_i:

$$p_1 = \mathsf{GENESIS}, \qquad c_i = \mathcal{H}(\mathrm{enc}(x_i, p_i)), \qquad p_{i+1} = c_i,$$

where enc is an injective canonical encoding and $\mathcal{H}$ a hash function. A stored sequence $\mathcal{C}' = ((x'_i, p'_i, c'_i))_{i=1}^{m}$ is a *chain* if $p'_1 = \mathsf{GENESIS}$, $p'_i = c'_{i-1}$ for $1 < i \le m$, $c'_i = \mathcal{H}(\mathrm{enc}(x'_i, p'_i))$ for every i, and creation times do not decrease. The verifier V accepts exactly the chains. The model is an idealisation for analysis; real logs differ in encoding, scope and verification rules, and the results below hold for any log to the extent that it matches the model.

Threat model. An adversary $\mathcal{A}$ obtains write access to storage after records are written: an insider, an attacker inside the database, or an operator under pressure to make a record disappear. $\mathcal{A}$ can read, delete, insert, reorder and modify stored records and can evaluate $\mathcal{H}$. $\mathcal{A}$ cannot find collisions of $\mathcal{H}$. A store-enforced rule binds only those who cannot disable it; we treat it as a control within $\mathcal{A}$'s reach.

Proposition 4.1 (What an unkeyed chain detects)**.** *Let $\mathcal{C}$ be a chain with content hashes $c_1, \dots, c_n$, let $\mathcal{C}' \neq \mathcal{C}$ be a stored sequence produced by $\mathcal{A}$, and let d be the first position at which the two differ, a position present in $\mathcal{C}$ and absent from $\mathcal{C}'$ counting as a difference.*

(a) If $\mathcal{C}'$ carries, at some position $k \ge d$, one of the original content hashes c_j, then V rejects $\mathcal{C}'$.

(b) Every prefix of a chain is a chain: V accepts every truncation of $\mathcal{C}$.

(c) For every sequence of records $y_1, \ldots, y_m$, the sequence obtained by computing the hashes over it is a chain: V accepts every rewriting of $\mathscr{C}$ whose hashes are recomputed from the point of change.

(d) Let $w = c_k$ be a head witnessed independently when $\mathscr{C}$ had k records. A verifier that accepts $\mathscr{C}'$ only if it is a chain and $c'_k = w$ rejects every $\mathscr{C}'$ that differs from $\mathscr{C}$ in its first k records, including every truncation below k and every rewriting of the first k records.

Statements (a) and (d) hold except with the probability that $\mathscr{A}$ finds a collision of $\mathscr{H}$ [ANALYTICAL].

Proof. (a) Suppose V accepts $\mathscr{C}'$ and $c'_k = c_j$ with $k \geq d$. Then $\mathscr{H}(\mathrm{enc}(x'_k, p'_k)) = \mathscr{H}(\mathrm{enc}(x_j, p_j))$, and without a collision $x'_k = x_j$ and $p'_k = p_j$; as both sequences are chains, $c'_{k-1} = c_{j-1}$, unless both previous hashes are GENESIS. Repeating the step, $\mathscr{C}'$ and $\mathscr{C}$ agree backwards from positions k and j. The descent ends when one side reaches the genesis value; since GENESIS is not a value of $\mathscr{H}$, both reach it at the same step, so $k = j$ and the two sequences agree on positions $1, \ldots, k$, which contradicts the difference at $d \leq k$. (b) The conditions that define a chain at position i refer only to positions up to i. (c) The verifier uses no secret, and $\mathscr{A}$ evaluates $\mathscr{H}$ as the writer did. (d) If $\mathscr{C}'$ is a chain with $c'_k = c_k$, the argument of (a) with $j = k$ shows that it agrees with $\mathscr{C}$ on its first k records; a sequence of fewer than k records has no position k and is rejected. □

Corollary 4.1 (Mutations without recomputation)**.** *If $\mathscr{A}$ computes no hash, every edit, deletion, insertion or reordering is detected by V, except a truncation of the tail.*

Proof. If $\mathscr{A}$ computes no hash, each record of $\mathscr{C}'$ at or after position d either is an original record with its stored hashes, or carries a content hash that $\mathscr{A}$ did not compute. A record of the second kind fails its own check unless its stored hash is a copied original, which is again a case of (a). A record of the first kind brings (a) into play directly. V therefore rejects unless $\mathscr{C}'$ has no record at or after d, that is, unless $\mathscr{C}'$ is a prefix of $\mathscr{C}$. An edited record keeps its stored hash, which then fails to recompute; a deleted prefix leaves a first record whose previous hash is an original value where GENESIS is required. □

Corollary 4.2 (Exposure window)**.** *If heads are witnessed at times $\tau_1 < \tau_2 < \cdots$, a record sealed after the latest witnessing can be removed or rewritten, permanently undetected, at any time before the next witnessing; once a later head is witnessed it cannot. The interval between witnessings is the exposure window of the chain, and a statement of evidence that rests on the chain must state it.*

4.2 Measurement

The suite's evidence-integrity experiment seals chains with Lawie's own sealing function, applies a mutation at a varying position in 50 trials per mutation type, and runs Lawie's chain verifier. Table 2 reports edits, deletions, reorderings, insertions and deleted prefixes: the verifier detected every one [MEASURED].

Table 2: Detection of mutations of chains sealed and verified by Lawie's audit-chain functions (evidence-integrity experiment).

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

4.3 Keyed chains and retention

A *keyed chain* authenticates each head with a message authentication code. A key changes the adversary of proposition 4.1(c): without it, $\mathscr{A}$ cannot produce valid tags, and rewriting by anyone outside the circle of key holders is detected. Three limits remain. The key holders can rewrite. The verifier must hold the key, so whoever can verify can also forge, and no third party can verify without being trusted with the power to rewrite. And truncation remains undetected, because part (b) holds for keyed chains as well. Schneier and Kelsey showed how to remove part of the key holder's power: evolve the key with each entry and erase the old key, so that an attacker who compromises the logging host at time T cannot alter entries made before T without detection [6]. Forward integrity limits what a later compromise can do. It does not make an operator a witness against itself.

Retention adds a further tension. Where a retention rule calls for old records to be deleted, a verifier that must tolerate a lawfully removed prefix cannot insist on the genesis condition.

Observation 4.1 (Retention without a checkpoint)**.** *A verifier that takes its starting hash from the first record it reads accepts every suffix of a chain. It cannot distinguish the lawful removal of old records under a retention rule from the removal of an inconvenient prefix.*

Proof. The chain conditions at positions after the first are satisfied by any suffix of a chain. The only condition a suffix violates is the genesis condition at its first position, which such a verifier does not check. □

The resolution is to make the deletion itself evidence. Before a prefix is removed, the retention procedure records the hash at the cut and the rule that authorised the removal, and this checkpoint is signed with a key that the database operator cannot use, or witnessed as in the next subsection. The verifier then starts from the checkpoint, not from whatever record it finds first. With the checkpoint, retention duties and tamper evidence are compatible; without it, they are not.

4.4 Witnesses: Merkle trees and transparency logs

Every result above points to one requirement: a value held by someone other than the party who could alter the record. Haber and Stornetta's linking scheme makes each timestamp depend on its predecessors, so that a backdated document would have to be inserted into a sequence that others already hold; their distributed-trust scheme has a document's timestamp signed by witnesses chosen as a pseudorandom function of the document, so that backdating needs the witnesses' collusion [7]. Publishing a log's head where no single party can withdraw it applies the same idea to any record. Merkle's hash trees let a single root commit to many records, with inclusion proofs whose length grows with the logarithm of their number [8]. Crosby and Wallach built tamper-evident logs on history trees with logarithmic proofs both of membership and of consistency between versions [9]. Certificate Transparency assembled these into append-only public logs: signed tree heads, inclusion proofs, and consistency proofs that a later tree extends an earlier one, with auditors and monitors comparing signed tree heads to expose a log that shows different histories to different readers [10]. A consistency proof is proposition 4.1(d) in operational form: it shows that the current log extends a witnessed head, so that neither truncation nor rewriting of the witnessed part can pass.

The security argument lives in construction details. RFC 6962 hashes leaves and interior nodes under different prefixes and splits an unbalanced tree at the largest power of two below its size, instead of duplicating nodes; both choices make the root determine the list. A construction must do the same: an anchor commits to exactly what the anchored value determines.

Independence is the part cryptography cannot supply: a witness the operator can instruct is not a witness. The head must be held by a party with no duty to the operator (a public ledger, a counterparty, a registry, a regulator, an auditor), whose honesty and retention the evidence then assumes and must state.

5 Signatures, keys and time

5.1 What a signature establishes

A valid signature establishes that the holder of a private key signed a given byte string. With a scheme such as Ed25519 [11] the statement is narrow and strong: no one without the key can produce a signature that verifies, and the verifier needs only the public key. Attribution to a person or institution requires a further, independently trusted binding of that public key to that party. Lawie's portable dossiers obtain the binding from a trust policy that pins the fingerprints of the accepted keys, and verify offline with Ed25519, ECDSA P-256 or RSA-PSS [TESTED]. A signature adds attribution and third-party verifiability to integrity. It adds nothing to support or correctness, and, like a keyed chain, it does not reveal the removal of a whole signed record from a collection: what is never presented is never checked.

5.2 Measurement

The evidence-integrity experiment presents Lawie's dossier verifier with an intact sandbox dossier and with altered versions of it (table 3). Every tampered dossier failed, and the intact dossier verified, also after its signing key's validity period had ended [MEASURED].

Table 3: Outcomes of Lawie's portable dossier verifier (evidence-integrity experiment, sandbox dossier, synthetic keys).

Case	Outcome
Intact dossier, verified within the key's validity	verified
A disclosed record edited	failed
A disclosed record removed	failed
A signature replaced by one made with a different key	failed
Trust policy pins a different key fingerprint	failed
Intact dossier, verified after the signing key's validity ended	verified

5.3 Why verification after key expiry is correct

The last row looks like a defect and is the opposite. Consider what a signature on evidence asserts.

Definition 5.1 (Validity of an evidential signature)**.** Let ς be a signature on a message m under key k, made at time t_s. Let k be valid on $[v_0, v_1]$ and possibly revoked at time r, the revocation covering either all signatures (compromise) or only signatures made at or after r (retirement). The signature is *valid as evidence*, as known at time t_v, if it verifies under the public key, $v_0 \le t_s \le v_1$, and no revocation known at t_v covers t_s.

Proposition 5.1 (Evidence outlives keys)**.** *Validity in the sense of definition 5.1 depends on the verification time t_v only through the revocations known at t_v. In particular it does not depend on whether $t_v \le v_1$. A rule that additionally required $t_v \le v_1$ would invalidate every signature on evidence once its key's validity ended, whatever the facts at the time of signing.*

Proof. The definition refers to t_v only in the clause on known revocations; the other two clauses concern the message, the public key and t_s. The second statement is immediate: for $t_v > v_1$ the additional clause is false for every signature made with k. □

The evidential content of a signature is "the holder of k signed m at t_s". That is a statement about the past, and its truth is fixed by the facts at t_s: whether the key was valid and uncompromised then. Keys

are retired to limit the damage of a future compromise, not to withdraw what was signed before. A rule that demanded a currently valid key would make every legal record expire on the operator's key-rotation schedule, which is destruction of evidence by timetable. A verification whose outcome does not depend on when it is run is also what an opposing party needs, because it can reproduce the result years later.

5.4 The condition: a witnessed time of signing

For any signature scheme, proposition 5.1 moves the weight onto t_s and onto the facts about the key. If the time of signing is declared by the signer, the rule protects backdating: whoever holds the key after its validity has ended, or after a compromise nobody has noticed, can sign today and declare a time inside the validity period. Against a compromise discovered later, the only revocation that helps is one covering all signatures of the key, because a backdated signature evades a revocation of later signatures; and revocation of all signatures destroys the good evidence with the bad.

The remedy in principle is a witness of the signing time: a timestamp from an authority independent of the signer, or the inclusion of the signature in an anchored log before the key's validity ends [7, 12, 13]; and key statements published by the key's owner outside the artifact, so that the producer of an artifact cannot restate them. With a witnessed t_s, definition 5.1 is decided by facts the signer cannot move, backdating requires corrupting the witness, and a later compromise can be handled by revoking only signatures not witnessed before it. The same witness that closes the truncation gap of proposition 4.1 closes the backdating gap here; one independent anchoring service serves both.

6 Reconstructing an agent decision

6.1 A standard

When an automated step in legal work is challenged, the question is what happened and on whose authority. Re-running the step does not answer it: a language model run again on the same input may answer differently, its provider may have changed it, and the sources may have moved. The answer has to come from what was recorded at the time.

Definition 6.1 (Reconstructible decision)**.** A decision taken in an automated run is *reconstructible* from a record R if a third party who holds only R, the artifacts that R references by hash, and the public keys or witnessed heads that R relies on, can determine

(i) the request: what was asked, by which principal, in which matter;
(ii) the inputs, including every legal source used, with its observation, span and version;
(iii) the procedure: the identity and version of the model, of its instructions, and of every tool and checker;
(iv) the actions: every tool call with its input and result;
(v) the checks: every verification applied, with its procedure and result;
(vi) the authority: for every consequential action, the approval it ran under, by whom, bound to which exact input;
(vii) the output;

and can establish that R is unaltered (integrity, section 4) and complete for the run.

Completeness is the hard clause. Proposition 4.1 concerns records that were written; a step that was never recorded leaves no gap in any chain. Integrity mechanisms establish that nothing recorded was changed. Only the architecture of recording can establish that everything consequential was recorded, for example by making the record a precondition of the action rather than a report written after it.

6.2 What the standard requires of a record

Each clause of definition 6.1 translates into a property of the record, and each property has a cost.

Commitments and retention. Inputs, tool results and outputs may be committed by hash rather than copied, which keeps confidential legal material out of a log that many people need to read, and still lets a party who holds a candidate document prove or refute that it was the one used. Hash commitments reconstruct nothing, however, unless the committed content is retained where a third party can obtain it; the record's value as evidence depends on that retention, and the record should say where the content is kept and for how long.

Named procedure. A record that does not name the model, the version of its instructions and the version of every tool and checker fails the procedure clause for every model-driven step, whatever else it contains. Model and prompt identity are as much part of the evidence as the inputs.

Bound authority. An approval value such as "approved" is not evidence of authority. The record must reference the approval itself, bound to the tool, its version and the hash of the exact input, as approval mechanisms of the kind Lawie implements produce [TESTED].

Witnessed integrity and time. The chain over the records needs the properties of section 4: a witnessed head at a stated interval, and creation times that are witnessed rather than supplied by the party whose conduct the record documents.

Completeness by construction. A record written after the run, on a best-effort basis, can be missing without trace. Completeness requires that a consequential step cannot run unless its record has been durably accepted, and that a refusal to accept the record stops the step.

The standard is demanding. It is nevertheless the standard a hostile reader will apply when a run is challenged.

7 Limitations travel with the artifact

A verified artifact invites over-reading. The recipient sees "verified", and the narrow statement that the bytes are unchanged and were signed by a pinned key becomes, in the recipient's mind, a statement that the content is correct. Disclaimers published elsewhere do not prevent this, because they are not where the reader is looking and they do not travel when the artifact is forwarded.

> **Principle.** What a verification does not establish is part of the artifact. The limitations of every link of the chain are stated inside the signed scope, so that an artifact stripped of them fails verification, and every verification report repeats them.

Lawie's dossier format enforces the principle mechanically. A dossier manifest is valid only if it declares itself integrity evidence only, states that verification proves neither substantive truth, admissibility, legal correctness nor authority, and carries three fixed limitation statements; these sit inside the hashed and signed manifest, and the verification report repeats them [IMPLEMENTED]. The three statements read: "Integrity verification does not prove substantive truth or factual accuracy." "Integrity verification does not prove legal admissibility or evidentiary weight." "Integrity verification does not prove legal correctness, authority, or completeness outside the declared export scope." A dossier from which they were removed, or in which they were altered, fails validation, and the manifest hash binds them to the signature.

Consequences. No verification comes without the limitations; a forwarded copy carries them; what the producer represented about the artifact is fixed in the signed text.

Assumptions. The recipient uses a verifier that reports the limitations, and reads the report. The limitations are true and sufficient for the artifact at hand.

From fixed statements to link-specific ones. Fixed statements of this kind are a floor. The principle asks for more: limitations generated from the chain itself, so that each link records what it established, by which procedure, and what it did not establish (for instance that a quotation was compared under a declared fold, that support was assessed by a method with a measured false-accept rate, that the source was a research copy rather than the authentic publication, or that no qualified lawyer reviewed the derivation), and the artifact carries the conjunction.

Cost. Artifacts become heavier and less flattering, limitation statements must be maintained as procedures change, and recipients may still ignore them. An artifact that verifies and says nothing about what verification means invites the reading that it proves more than it does.

8 Requirements and open problems

8.1 Requirements for a legal evidence chain

The requirements below follow from the preceding sections. They are stated for any system that produces legal records for others to rely on, as properties a hostile reader can test, not as a programme of work and not as a verdict on any particular system.

- **R1 Admission before reliance.** A legal statement may rest only on observations retrieved from the official publisher under an approved source contract with a reviewed licence; without admission the system refuses (section 3).
- **R2 Exact spans over the authentic payload.** Every quotation is located by byte offsets and a span hash through a reviewed derivation for its official format.
- **R3 Two canonicalisations, kept apart.** Integrity canonicalisation never alters a value; comparison canonicalisation folds only declared classes and records the fold with each comparison (proposition 3.1).
- **R4 Currentness for the date that matters.** Each span carries its observation's place in the supersession chain and the date for which its authority was assessed, and authority is assessed for application, not only for force (LL-TMP-01).
- **R5 Derivations with their method.** A support claim records the procedure that checked it and that procedure's measured error rates; a confidence that carries no information is not recorded as evidence.
- **R6 Independent, bound review.** Review is by a named person distinct from the requester, bound to the hashes of what was reviewed and dated within its validity.
- **R7 Named procedure.** Records of automated steps name the model, the instructions and every tool and checker version (definition 6.1).
- **R8 Record before acting.** Consequential steps are recorded as a precondition of their execution, so that completeness is a property of the architecture.
- **R9 Independent witnessing.** Chain heads, signing times and key status are witnessed or published by a party the operator cannot instruct, at a stated interval; deletion under retention leaves an authenticated checkpoint; signatures are judged at the witnessed time of signing and outlive their keys (propositions 4.1 and 5.1).
- **R10 Limitations inside the signed scope, link by link.** Every artifact states, inside what is signed, what each link established and what it did not (section 7).

8.2 Open problems

Support beyond quotation. Exact and canonical matching settle quotations [MEASURED]. LLRB-1 does not measure whether a verifier accepts a correct paraphrase; the lexical measures that would accept one also accept decisive edits [MEASURED], and typed comparison is exact only for the element types it models [MEASURED]. How to produce evidence of support for paraphrase, short of recorded human review, is open.

Completeness. Integrity mechanisms protect what was recorded. Establishing that every consequential step was recorded is a property of system architecture and of independent observation, and has no general cryptographic solution.

Long-term validity. Legal records must outlive hash functions, signature schemes and the organisations that hold their keys. A record whose hash function weakens must be committed again under a stronger one while the old commitment is still trusted, as renewal of timestamps was proposed to do [12]; who renews legal records over decades, and at whose cost, is not settled.

Who witnesses, and what it reveals. The cryptography of witnessing is mature; the institutions are not. Whether courts, registries, notaries, regulators or public ledgers should hold the heads of legal records is open, and a published head, though it reveals no content, reveals volume and timing.

Weight in law. What weight a court gives to integrity evidence depends on the procedural law of the forum. Nothing in this paper establishes admissibility, and the dossier format says so itself [IMPLEMENTED].

9 Evidence boundary of this paper

Evidence boundary. What was measured. The chain and dossier results come from the suite's evidence-integrity experiment, which runs Lawie's sealing, chain-verification and dossier-verification functions unchanged and offline on synthetic records, a sandbox dossier and synthetic keys [MEASURED]; the benchmark results come from the pre-registered LLRB-1 confirmatory run on English EU legislation [MEASURED]. Nothing concerns a deployed system, a customer or production data.
What was not done. The propositions hold under their stated assumptions, chiefly collision resistance and an independent witness. The experiment uses one chain length and one sandbox dossier, and does not exercise key management, concurrent writers or failed writes. No production deployment exists [BLOCKED], and no independent audit or human expert review of these components or of this paper has taken place [ABSENT]; AI-assisted drafting and internal review are not peer review. Nothing here concerns the admissibility of any record in any court.
What this paper is not. This paper 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.

References

[1] United States District Court for the Southern District of New York. Mata v. Avianca, Inc., No. 22-cv-1461 (PKC), Opinion and Order on Sanctions, 2023. URL https://www.courtlistener.com/docket/63107798/mata-v-avianca-inc/. 22 June 2023.

[2] 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.

[3] Publications Office of the European Union. Regulation (EU) 2024/1689, consolidated text of 27 July 2026. EUR-Lex, document 02024R1689-20260727, 2026. URL https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:02024R1689-20260727.

[4] Luc Moreau and Paolo Missier. PROV-DM: The PROV Data Model. W3C Recommendation, 2013. URL https://www.w3.org/TR/prov-dm/.

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

[6] Bruce Schneier and John Kelsey. Secure Audit Logs to Support Computer Forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, 1999. doi: 10.1145/317087.317089.

[7] 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.

[8] Ralph C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378, 1988. doi: 10.1007/3-540-48184-2_32.

[9] 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.

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

[11] 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.

[12] Dave Bayer, Stuart Haber, and W. Scott Stornetta. Improving the Efficiency and Reliability of Digital Time-Stamping. In *Sequences II: Methods in Communication, Security, and Computer Science*, pages 329–334. Springer, 1993. doi: 10.1007/978-1-4613-9323-8_24.

[13] 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.