Argentis Labs

Working Paper 02 · 2026

Daubert Meets ChatGPT

The audit substrate beneath every defensible use of AI in regulated work

Argentis Labs Research

2026-05-26

Abstract

The next career-ending question for an expert witness will be five words: *Where did you use AI?*

The expert who has used AI — and at this point that is most experts in most disciplines — has four more questions coming. What inputs did you provide the system. What outputs did it produce. Which outputs did you adopt, and on what basis. How can the trier of fact verify that the workflow you describe is the workflow that actually occurred.

The fifth question is the one nearly no current AI system can answer. The expert's audit trail is held by the expert's own software and attested by the expert. Under Federal Rule of Evidence 702 as amended in December 2023, the proponent must show by a preponderance that the methods were reliably applied to the facts of the case. An audit trail the expert alone can produce, alone can interpret, and alone can attest to is the expert's account of the expert's own work — not a reliable-application showing the rule contemplates. The question is becoming inescapable because the trier of fact no longer accepts the operator's account as the ground of the showing — attestation alone is no longer what the rule contemplates.

Daubert is starting to ask this of every AI-touched expert analysis. Regulators are asking it of legal AI systems; institutional verifiers are asking it of autonomous-agent actions. Across all three, one architectural form answers.

The first wave of AI in regulated professions sold answers. The second wave sells defensibility. The defensibility claim depends on an artifact almost no current system actually produces: an audit substrate the operator cannot rewrite — held in a separate trust boundary, physically immutable, signed by a key any third party can verify with no cooperation from the operator. This is the supervision substrate. It is the layer beneath every defensibility claim a regulated professional now needs to make, and the architectural specification follows.

§1 — Three primitives, and the one this paper specifies

Three architectural primitives together compose the supervision substrate for AI in regulated work. Each addresses a different verification question that the next decade of regulated AI is going to be asked.

Bounded action. What an autonomous system can do is defined in typed structural state and dispatched by deterministic predicate. Agents propose; the architecture decides what executes. The verification question at this layer is whether the action the system took is the action the architecture authorized — a question of relational consistency across institutional state.

Architectural gate. The human-in-the-loop sign-off is the architectural commit point, bound to the professional's licensed identity. The verification question at this layer is whether the human who authorized the action held the authority to do so under the firm's supervision rules — a question of regulatory enforceability at the moment of action.

Verifiable record. Every event — proposal, sign-off, execution — is written to an audit substrate whose integrity does not depend on the operator's attestation. The verification question at this layer is whether the trail any party produces is the trail the workflow generated — a question of evidentiary durability across time.

The first two primitives have been treated in earlier Argentis Labs research. Locally Correct, Globally Wrong (Working Paper 01) specified the bounded-action layer for autonomous systems. The Opinion 512 commentary specified the architectural-gate layer for legal practice. This paper takes up the third: what the audit substrate must be, and why operator-attested substrates fail the verification question they are increasingly going to face.

The three primitives are stacked, not coordinate. The substrate this paper specifies sits beneath the other two. It records what the bounded-action primitive authorized and what the architectural gate finalized — and it is the layer that keeps both records durable across the timelines on which the actions will be challenged.

---

§2 — The common substrate

The same requirement appears in three otherwise unrelated regulated workflows — expert testimony, legal practice, autonomous-agent verification. The architecture that satisfies it is the same in each case. The substrate this paper specifies is that architecture.

Expert testimony. The five voir-dire questions that follow an expert's first use of AI compose into a single architectural demand. *Where did you use AI*, *what inputs did you provide*, *what outputs did it produce*, *which outputs did you adopt and on what basis* — each can be answered in narrative. *How can the trier of fact verify that the workflow you describe is the workflow that occurred* cannot. It requires an artifact.

Under Federal Rule of Evidence 702 as amended in December 2023, the proponent must show by a preponderance that the methods were reliably applied to the facts of the case. The expert producing an operator-held audit trail attestable only by the expert is asking the court to take the expert's word for the reliable-application showing. Under the trend in post-amendment admissibility rulings, this is increasingly insufficient. The expert producing a substrate the opposing examiner can verify independently, signed in a key custody the expert does not control and immutable by physical construction, has a structurally different answer to the same question. It is not a stronger version of the expert's word. It is the evidentiary substrate the word rests on — the artifact opposing counsel inspects to test whether the expert's reliable-application testimony holds.

Legal practice. ABA Formal Opinion 512 reads Rules 5.3, 1.6, and 1.5 as architectural questions about the firm's AI workflow. Rule 5.3(b)'s *reasonable efforts to ensure* obligation can be satisfied two ways: by policy backed by training records and attorney attestations, or by an architecture that cannot be circumvented. Both readings are defensible under the rule. Only the second survives the four-year malpractice timeline, because vendor representations age into the gap between what was attested in procurement and what the system actually does after eighteen months of unattributed code changes. Malpractice carriers and bar grievance committees consume architectural evidence more readily than policy attestations. The architectural answer requires a substrate the firm — and the firm's adversaries — can produce on demand, without trusting any party's reconstruction of the trail.

Autonomous-agent verification. Where autonomous and semi-autonomous systems act on behalf of institutions — agentic compliance review, DeFi authorization, automated transaction approval — the verification question is whether institutional legitimacy held at the moment of action. Working Paper 01 argued that institutional legitimacy is not detectable by the primitives that detect transaction-local validity; an explicit relational layer over typed structural state is required to evaluate it. The audit substrate records the structural decision: that the deterministic authority layer evaluated the action under institutional state at the moment of action and returned the verdict the system acted on. Without that record in a substrate the operator cannot rewrite, the institutional verdict's authority decays into the operator's reconstruction — exactly the wrong place to anchor institutional legitimacy.

Three workflows, one requirement. The architectural form that satisfies the requirement is the same in each case. It is the commitment that runs across all three pieces in this program: the Opinion 512 commentary made it at the regulatory layer, Working Paper 01 reinforced it at the runtime layer, and this paper states it at the substrate layer beneath both — structural state must be explicit and externally verifiable.

---

§3 — Verification without trust

Most current AI systems for regulated work ship an audit substrate that fails the verification question by design. The pattern is not exotic. It is the architecture of most current enterprise AI observability — Datadog, LangSmith, Arize, Braintrust, internal Postgres trace stores. The instrumentation is excellent. The substrate is the operator's.

Events and integrity records are held in the same trust boundary; the same principal that writes the events can rewrite them; the operator attests to its own audit history. This is supervision-by-policy applied at the audit layer — the audit is enforced by the operator's promise to record faithfully, not by an architecture the operator cannot circumvent.

This is more than an engineering oversight; it is the locally-correct-globally-wrong pattern reaching the audit layer directly. Each individual entry in an operator-attested audit trail is locally correct — written at the moment of the event, formatted properly, internally consistent. The relational property that fails is independent verifiability. Local correctness is preserved at every entry; the global integrity claim fails for a structural reason no per-entry inspection can surface.

Verification without trust is what the supervision substrate provides.

The architectural requirement that follows from this critique is precise. The integrity record must live in a trust boundary the operator cannot reach. It must be physically immutable for as long as the work it records remains discoverable. It must be signed by a key any third party can verify without the operator's cooperation. It must be isolated per regulated artifact, so producing it for one matter does not disclose it for any other. These four properties together define the substrate the question demands. The architecture below specifies each in plain terms; Appendix B carries the construction.

---

§4 — What the integrity record must be

The integrity record is the artifact every regulated AI step produces. The AI proposes, the human signs off, the action executes — and every step is recorded into a tracking layer the operator cannot rewrite. The four properties below describe what each record must be.

Separated trust boundaries. The system that produces the work and the system that records the work are run by different identities, with different keys and different permissions. If one is compromised, the other cannot be rewritten from the same compromise. The legal-evidentiary analog is dual-control in chain-of-custody — the practice that no single person can both handle the evidence and certify the handling.

Physical immutability. Once the record is written, no one can delete or alter it for the duration the law of the regulated work requires. Not the operator's most privileged employee, not the operator's CEO, not the operator's lawyer with a subpoena to suppress. The storage layer itself refuses the request — at a level beneath any user account or operator instruction. This is not *we promise not to delete*. It is *we cannot delete*.

Public-key verification. Every record is signed by a key the operator never touches. The signing happens inside hardware the operator cannot access. Anyone — opposing counsel, a regulator, a court — can verify the signature using only the public half of the key, with no cooperation from the operator. The verification needs no Argentis service, no AWS endpoint, no proprietary tool. It is a math operation a college freshman with the right library can run.

Per-matter isolation. Each regulated artifact — each matter, each case, each loan, each claim — has its own track. Producing the track for one does not require revealing any of the others. This is what allows the substrate to operate inside a law firm with attorney-client privilege intact, inside a hospital with HIPAA intact, inside a lender with loan-level confidentiality intact.

These four together are the substrate. None is individually difficult. The architectural requirement is that all four must hold before any application is built on top. The substrate Argentis Labs has built and validated against real infrastructure exhibits all four; the validation methodology and results are in Appendix A, and the technical construction underlying each property is in Appendix B.

Why all four matter together is the heart of the architectural claim. Without separated trust, the tracking system can be rewritten by whoever runs it. Without physical immutability, *not deleted* is a promise rather than a guarantee. Without public-key verification, the only person who can validate the record is the operator. Without per-matter isolation, producing one record means producing all of them. Each property closes one path the verifier of the future will probe.

Together they close the four paths that matter.

---

§5 — Integrity before query

The supervision substrate is the layer beneath every query, dashboard, observability tool, and reporting interface the customer interacts with. The composition rule is exact: the integrity layer is the system of record; everything above it is derived, queryable, and vendor-swappable without affecting the substrate.

Integrity before query in this piece. Authority before alignment in Working Paper 01. Architecture before paperwork in the Opinion 512 commentary. Three composition rules at three layers; one architectural shape. The same architectural claim at each level — the lower layer must hold before the upper one means anything. A regulator, opposing counsel, or court who cannot trust the substrate has no use for the dashboard built on top of it.

The composition shows up empirically. During the validation work for this paper, the verifier's first end-to-end run failed on clean data rather than tampered data: the writer was formatting data in a way the verifier did not expect — a single-character difference the writer's own tests had not caught. The verifier, independently implemented and recomputing every hash from scratch, surfaced the difference as a verification failure on otherwise-clean events. The integrity-before-query composition is therefore not only an architectural principle but the substrate's own continuous-integration witness: any drift in data format, hash formula, or schema between writer and spec surfaces as a verifier failure on otherwise-clean data.

This is the architectural reason the substrate must be designed before the application. Once committed, it allows the application layer to evolve, the observability layer to be swapped, and the query interface to be tuned, all without touching the integrity claim. It is what every claim built above rests on.

---

§6 — The substrate as commercial enabler

The same premise that drives the architectural argument drives a parallel commercial argument. As trust narrows and time for human review compresses, both the architectural form and the commercial form must change in the same direction. The substrate is not a compliance cost or a digression from the specification; it is the operational consequence of the trust-and-time-compression premise expressed as a commercial relationship.

Incumbent legal-AI pricing is seat-based: vendor revenue scales with attorney headcount. As AI compresses headcount and MSO consolidation reduces seats, seat-priced ARR declines with the customer's progress; the vendor is short its own thesis. The architectural shift — verification without trust — produces a commercial shift in the same direction. A substrate that records every matter cleared as a verifiable event neither vendor nor customer alone can rewrite enables matter-velocity pricing: vendor revenue scales with the customer's leverage, not with the customer's headcount. "Pricing per matter cleared" is only possible if an *event-of-truth substrate* exists to count against, and it is only event-of-truth if no party can unilaterally adjust the count. The same architecture enables a structurally different conversation about the cost of expert testimony — where the chain, not the expert, is the artifact under examination.

Thus, pricing per matter cleared is what the substrate makes commercially possible as the workforce compresses. Aligned incentives, verifiable claims on both sides, a commercial model that scales with the customer's success rather than against it — these are the operational consequences of the same architectural specification the piece has been making.

---

§7 — Defensibility as architecture

The architecture that distinguishes systems that survive challenge from systems that do not is not visible at the application layer. It lies in the substrate beneath. Three durability claims have surfaced across this piece — the integrity of supervision across a four-year malpractice timeline, the integrity of authorization across the operational lifetime of an autonomous system, the integrity of process across deposition, voir dire, and discovery. They are not three separate guarantees. They are three facets of one architectural claim, expressed at three layers of the same program.

Opinion 512 specified what supervision must be at the regulatory layer. Working Paper 01 specified what authorization must be at the runtime layer. This paper specifies what the audit substrate must be at the layer beneath both. The commitment is the same in each: make structural state explicit and externally verifiable. The supervision substrate is the architectural completion of the program — the foundational layer the other two architectural claims rest on, not a fourth claim added alongside three.

The architectural specification above is operational; the substrate has been built and validated against real infrastructure (Appendix A). Post-amendment FRE 702, Opinion 512's vendor-vetting checklist, and whatever the next institutional verifier asks of autonomous action converge on the same primitive.

A standalone verification exhibit for this paper will be published, applying the substrate described in §4 and Appendix B to the paper itself: content hash, signed digest envelope, retention metadata, and the verification commands a reader runs locally against the three artifacts.

The next piece in this program takes up the question this one opens: what kind of expertise the substrate makes possible, and what happens to the cost stack of expert testimony, discovery, and the Daubert hearing when the chain itself becomes the artifact under examination.

> *In a world with less trust and less time for human review, our agents must have defensible actions — and defensibility now means independently verifiable.*

Appendix A — Validation findings

The substrate's architectural claims were tested against real AWS and Postgres infrastructure during session `daubert-20260520T224005Z` on 2026-05-20. Five claims were evaluated: three of the four §4 properties — separated trust boundaries (Claim 2), physical immutability (Claim 3), and public-key verification (Claims 4 and 5) — plus the writer-side atomic chain commit precondition (Claim 1, Appendix B.5). The fourth §4 property, per-matter isolation, was not exercised in this session and is treated in Limits below. Each pairs a positive-case test with a deliberate negative-control variant where the construction admits one. Full commands, observed outputs, and decision rationale will be available in the redacted source notes at `argentislabs.io/research/daubert-meets-chatgpt/validation`.

Results

Claim	Apparatus	Positive result	Negative control
1. Atomic chain commit	Five concurrent writers × 20 events each; event row and chain link committed in a single Postgres transaction with `ISOLATION LEVEL SERIALIZABLE`.	Zero orphan observations across 100 events. SSI conflict detection produced 72 serialization failures; all retried successfully and committed.	Split-transaction variant with a 100 ms gap between event-row insert and chain-link insert: 15 unique orphan observations across 15 events (every event caught).
2. Separated trust boundaries	Two IAM principals, each scoped to one S3 bucket via positive-Allow inline policy; cross-principal write attempted to the bucket the principal is not scoped for.	Write fails against IAM default-deny: *is not authorized to perform: s3:PutObject on resource*. No explicit `Deny` exists; none is needed.	Apparatus sensitivity confirmed by the same principal's successful write to its own bucket.
3. Physical immutability	Admin IAM principal with full `s3:*` permissions; `DeleteObject` and `PutObjectRetention` (shortening) attempted against a digest under active Compliance retention.	Both calls fail at the storage layer: *Access Denied because object is protected by object lock*. The retain-until date cannot be reduced.	Root-account variant not exercised. AWS documents identical Compliance enforcement for root; the documented-only framing is stated in Appendix B.2.
4. Public-key signature roundtrip	AWS KMS asymmetric key, `KeySpec=ECC_NIST_P256` and `SigningAlgorithm=ECDSA_SHA_256`; signing under `MessageType=RAW`; offline verifier using the `cryptography` library against the PEM public key only.	Clean digest envelope verifies.	Single-byte tamper of the canonical payload: `InvalidSignature`.

5. Substrate-independent verification	Three-artifact trust path — the substrate's chain-of-custody artifact in machine-checkable form: chain export (JSONL), signed digest envelope (JSON), PEM public key. Verifier passed only these three artifacts; no AWS or Argentis dependency.	Verifier succeeds. Failures localize to specific `event_id` and `chain_sequence`, typed by failure mode (`content_hash` mismatch, `chain_hash` mismatch, signature invalid).	First end-to-end run failed on clean data: the writer computed payload hashes with non-canonical JSON separators. Verifier surfaced the drift as a verification failure on otherwise-clean events (see §5).

Methodological note: paired falsification

Each architectural claim is paired, where the construction admits, with a deliberate negative-control variant of the same workload — a variant that should, if the apparatus is sensitive, produce the failure the positive case denies. The substantive claim is not *we observed no failure*. The substantive claim is *we confirmed the apparatus could detect the failure, then ran it cleanly and observed none*. The pairing is the method's load-bearing element: an unpaired positive result is not falsifiable; a paired result is.

The five-claim suite documented above is composable and re-runnable. Claims 1, 4, and 5 are committed to the substrate's CI pipeline on every commit touching the audit-chain code (they are fast and have no external dependencies once AWS LocalStack or equivalent mocks are wired up); Claims 2 and 3 require real AWS and run on a slower cadence against a dedicated validation account. A failed run is the same kind of signal the verifier provides for chain integrity — substrate integrity through periodic falsification rather than continuous monitoring.

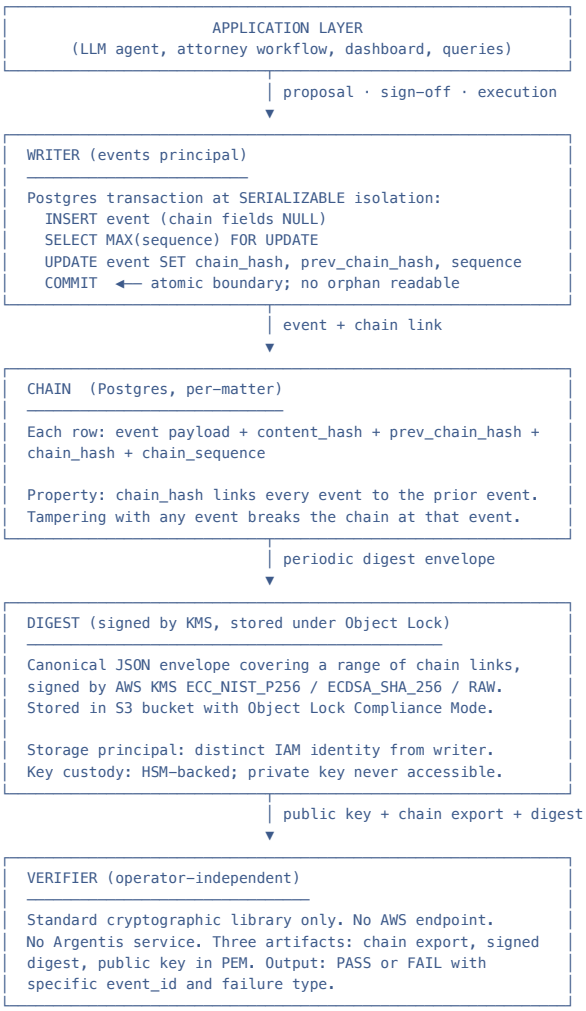
Limits of the validation method

The validation session has three limits we state explicitly. First, per-matter isolation (§4) was not stress-tested independently; it follows from the substrate's `matter_id` partitioning by construction and we intend to exercise it in a subsequent run. Second, Claim 3's root-account variant was not exercised; AWS documents Compliance Mode immunity for root identically to admin-IAM, and the paper states the claim demonstrated and the claim documented as distinct (Appendix B.2). Third, the substrate's evidentiary performance — its behavior in actual chain-of-custody scrutiny in discovery, deposition, and Daubert hearings — has not been exercised in any litigated matter. The architectural properties are validated; the legal-evidentiary properties they imply are documented and intended, but the substrate has not yet been challenged adversarially in a courtroom. The next piece in this program addresses what such challenge would look like and what cost compression would follow.

Appendix B — Technical construction

The four architectural properties specified in §4 are realized by a specific technical construction. This appendix documents each property's implementation in the depth a technical reader requires.

Architecture overview



B.1 — Separated trust boundaries

The writer principal and the storage principal are distinct AWS IAM identities with positive-Allow inline policies, each scoped to one S3 bucket. The cross-principal write attempt fails against IAM's default-deny evaluation, not against an explicit `Deny` statement. The implicit-deny semantics matter architecturally — explicit `Deny` statements can be removed by any actor with `iam:PutUserPolicy`, while the absence of an `Allow` cannot be circumvented without adding a CloudTrail-attributable policy change. The separation property therefore degrades gracefully under partial IAM compromise rather than catastrophically.

B.2 — Physical immutability

The integrity record is held in an S3 bucket with Object Lock Compliance Mode and a default retention period appropriate to the regulated work's discovery window. Compliance Mode produces a storage-layer denial that operates independently of IAM. An admin IAM principal with valid `s3:DeleteObject` cannot delete a digest under active retention; the same principal cannot reduce the retention period via `PutObjectRetention`. Retention is itself immutable, not merely the delete operation. AWS documents that the same enforcement applies to the root account; our validation tested the admin-IAM principal as the closest available demonstration.

B.3 — Public-key signatures

The signing key is an AWS KMS asymmetric key with `KeySpec=ECC_NIST_P256` and `SigningAlgorithm=ECDSA_SHA_256`. KMS hashes canonical-JSON digest bytes internally with SHA-256 under `MessageType=RAW`; the verifier hashes the same canonical bytes locally, and the two hashes must agree by construction. Clean signatures verify; a single-byte tamper of the canonical payload fails with `InvalidSignature`. The alternative — `MessageType=DIGEST` — would require the signer to transmit a hash value alongside the digest, introducing a parameter the verifier must accept rather than recompute. `RAW` is therefore the default because it minimizes the verifier's trust surface, not because it is operationally simpler.

KMS added Ed25519 support in November 2025 as a viable alternative; the substrate uses P-256 for the depth of FIPS 186-5 acceptance in the Daubert-relevant evidentiary context and for compatibility with legacy verification environments. The verifier's trust path is exactly three artifacts: chain export (JSONL), signed digest envelope (JSON), public key in PEM. No AWS endpoint is contacted; no Argentis service is contacted; the KMS `KeyId` in the digest envelope is metadata for key-rotation labelling, not part of the verification path. Every cryptographic primitive on which the verification rests is public and standards-track — RFC 8785 canonical JSON, SHA-256, ECDSA over NIST P-256 (FIPS 186-5), RFC 3279 DER signature encoding, PEM SubjectPublicKeyInfo — and independently implemented in every major language.

B.4 — Per-matter isolation

The substrate implements isolation via `matter_id` partitioning at the row, chain, and digest levels. Each event is associated with one `matter_id`; each chain link is scoped to one `matter_id`; each digest covers events from one `matter_id` only. A request to produce the substrate for one matter does not require touching the substrate for any other.

The validation work documented in Appendix A did not stress-test isolation independently. This is a property the substrate's design carries by construction, and one we intend to exercise in a subsequent validation run.

B.5 — The atomic chain commit (writer-side precondition)

Underlying the four properties is a writer-side construction the validation work also exercised: each event and its chain link commit atomically inside a single Postgres transaction at SERIALIZABLE isolation, so no reader can observe an event without its chain hash and no concurrent writer can produce a chain fork — two valid event sequences each claiming to be the canonical trail —

on the same matter. This is the writer-side precondition that makes the four externally-visible properties achievable.

Two distinct guarantees compose at this layer:

- **Atomicity** — the property that no reader observes the row in an intermediate state where the event row exists but chain fields are still NULL — is given by the transaction boundary itself. Any isolation level prevents this. The substrate uses SERIALIZABLE because of the second guarantee, not the first.
- **No chain forks** — the property that two concurrent writers attempting to append events to the same matter cannot both produce valid sequences — comes from SERIALIZABLE's SSI implementation. Two writers that both read `MAX(sequence) = N` and both attempt to commit `INSERT ... sequence = N+1` produce one successful commit and one serialization failure (`40001`) that retries. The retry rate at high contention on a single matter is roughly 72%, validated empirically. This is the operational cost of the no-fork guarantee; the substrate's deployment plans for it.

References

Argentis Labs Research (May 6, 2026). *Opinion 512 Is an Architectural Specification, Not a Procurement Checklist.* `argentislabs.io/research/opinion-512`

Argentis Labs Research (May 8, 2026). *Locally Correct, Globally Wrong: A state-surface partition for authorization verification in autonomous systems.* Working Paper 01. `argentislabs.io/research`

American Bar Association (July 29, 2024). *Formal Opinion 512: Generative Artificial Intelligence Tools.* `americanbar.org`

Federal Rules of Evidence 702 (as amended December 1, 2023).

Daubert v. Merrell Dow Pharmaceuticals, Inc., 509 U.S. 579 (1993).

Federal Rules of Evidence 902(14) (self-authenticating electronic records).

Haber, S. & Stornetta, W. S. (1991). *How to Time-Stamp a Digital Document.* Journal of Cryptology, 3(2): 99–111.

National Institute of Standards and Technology (February 2023). *FIPS 186-5: Digital Signature Standard (DSS).* `nist.gov`

RFC 3279. *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and CRL Profile.* IETF (April 2002).

RFC 5280. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.* IETF (May 2008).

RFC 8785. *JSON Canonicalization Scheme (JCS).* IETF (June 2020).

Amazon Web Services. *S3 Object Lock — Compliance Mode.* AWS documentation.

Contact

Argentis Labs Research — `solutions@argentislabs.io` — `argentislabs.io/research`