

# A Symbolic Analysis of Agent Evidence Packages

Anton Sokolov

Received: date / Accepted: date

**Abstract** This article studies an Agent Evidence Package (AEP) profile from the Agent Trust Framework (EATF) and a Model Context Protocol (MCP) attestation profile using symbolic protocol modeling in Tamarin. The starting point is an ordinary engineering question: if a valid signed evidence package is shown to a verifier twice, is the second acceptance only another inspection, or a new action? The model represents agent authorization, gateway forwarding, signer issuance, timestamping, ledger anchoring, and verifier acceptance under explicit Dolev–Yao and trust-anchor assumptions. Several correspondence-style properties verify, including signer and agent authentication, payload integrity, timestamp-event linkage, forward-integrity-style pre-compromise attribution, and resistance to a malicious gateway acting as a signing oracle. The clearest negative result concerns replay semantics: a stateless verifier can accept the same otherwise valid package hash more than once, so freshness and ledger anchoring alone do not imply unique verifier acceptance. A replay-cache variant verifies uniqueness under added verifier state. The contribution is model-bounded and profile-dependent: replay resistance must be stated as a verifier-state or challenge-bound property when a downstream application treats acceptance as a fresh event.

**Keywords** Tamarin Prover · formal methods · protocol verification · replay resistance · agent attestation · evidence packages

## 1 Introduction

Agent evidence packages are meant to make machine-mediated actions inspectable after the fact. A verifier can open a portable object and see canonical bytes, signatures, timestamp material, ledger anchors, and instructions about what was checked. That promise is useful precisely because the package can travel. It also creates a subtle security question. If the same authentic package is presented twice, is the second verifier acceptance merely another evidence check, or is it a fresh consequential action?

This article analyses that question for an Agent Evidence Package (AEP) profile from the Agent Trust Framework (EATF) and a Model Context Protocol (MCP) attestation profile. EATF is the surrounding Tyche Institute research framework for agent evidence packages, verifier outputs, fixtures, and claim boundaries; in this article it is treated as author-controlled research material, not as a third-party standard or a regulated trust service. AEP is the evidence-container profile studied here. MCP is the agent/tool gateway setting used to model a gateway between agent authorization and downstream signing or verification.

The replay question did not begin as a polished theorem. It surfaced during a local language-model-assisted test pass over the Tyche Institute AEP research package, where the test assistant was asked to behave like a downstream tester probing a source-available evidence package rather than like a standards editor.<sup>1</sup> One generated test prompt treated a signed package as a reusable

---

Anton Sokolov  
Tyche Institute, Tallinn, Estonia  
ORCID: 0000-0003-2452-7096  
E-mail: corresponding-email-to-be-entered

---

<sup>1</sup> Here “source-available” and “open” refer to the reproducibility posture of the research artifact: source snapshots, model files, proof outputs, and manifests are being prepared for public review. The final redistribution license remains an author-side deposit choice.

object and asked what should happen when a verifier sees it twice. That prompt did not prove anything. It identified the engineering question that the Tamarin model tests below: whether verifier acceptance has memory.<sup>2</sup>

The analysis focuses on the core flow from agent to MCP gateway, signer, ledger, and verifier. It asks whether verifier acceptance can be linked to prior honest authorization and signing; whether accepted payloads match their signed canonical form; whether accepted timestamps derive from a modeled timestamping event; whether later agent-key compromise explains earlier verifier acceptance; whether a corrupted gateway can insert claims attributed to an honest agent; and whether freshness plus ledger anchoring is enough to detect replay.

The contribution is deliberately narrow. It is not a new cryptographic scheme and it does not prove deployment-wide security for any implementation. It is a small formal case study showing that “authentic evidence” and “fresh action authorization” are separate verifier-profile questions. A package can be safe to re-read for archival evidence and unsafe to reuse as a second consequential action.

## 2 Related Work and Positioning

This research sits in three overlapping lines of work. First, symbolic protocol analysis asks whether trace properties follow under an ideal-cryptography adversary model. This includes the Dolev–Yao abstraction [1], classic attack-synthesis work such as Lowe’s Needham–Schroeder analysis [2], and modern tools such as Tamarin for correspondence lemmas and adversarial trace search [3, 4].

Second, attestation and transparency infrastructure gives verifiers signed evidence about systems, keys, timestamps, and logs. The standards background includes timestamping work and RFC 3161 timestamp evidence [5, 6], JSON canonicalization [7], remote-attestation architecture and token structure [8, 9], and Certificate Transparency as background for the ledger-anchoring abstraction [10]. PKI and trust-service policy documents provide the surrounding governance vocabulary for keys, certificates, relying-party rules, and trust lists [11, 12, 13, 14]. Long-lived evidence also sits near legal trust-service rules, AI documentation duties, certificate governance, and post-quantum migration work [15, 16, 17, 18, 19, 20, 21, 22].

<sup>2</sup> The language-model-assisted pass is part of the discovery narrative, not part of the formal evidence. The reported claims rest on the Tamarin theories, captured query outputs, and human-reviewed trace interpretations.

Third, infrastructure studies explain why technical checks become operational institutions only when their claim boundaries are visible [23, 24, 25, 26, 27]. Reproducibility practice adds a fourth pressure: artifacts need durable identifiers, metadata, and enough structure for later readers to inspect what was actually run [28, 29, 30]. That is why the article treats replay as a verifier-profile problem, not merely as a cryptographic primitive problem.

## 3 Source Material and Artifact Labels

The primary source material consists of reviewed snapshots of the AEP profile, the MCP Attestation Profile, and the MIRROR evidence-bundle schema. MIRROR is a manifest-and-hash evidence-bundle schema used here to make claims, sources, and artifacts reviewable together. It is treated as a reviewed schema snapshot for the artifact companion, not as an external standard. Public AEP artifact-family records already exist as a Zenodo artifact family [31]; the exact source snapshots reviewed for this article are identified by local source labels until a submission-specific artifact companion is deposited.

The reviewed AEP snapshot also contains an optional OVERT receipt entry and OVERT-aware verifier obligations. This article’s main replay result concerns base AEP/MCP acceptance, but the artifact set includes focused OVERT compatibility checks as well.<sup>3</sup> Those checks ask a related question: if a compatibility format changes what a verifier may claim, which receipt fields, witness references, package identities, and manifest links must be bound to the accepted evidence chain?

The extraction note is a short design record, to be deposited with the artifact companion, that states how those prose sources were translated into model roles, messages, key material, ledger semantics, freshness abstraction, and known simplifications. Each declared model simplification is recorded there; an unrecorded simplification would be a gap in the model’s traceability claim.

Table 1 explains short labels used in the article. They are artifact lookup handles, not new theoretical concepts.

<sup>3</sup> OVERT is treated here as the optional receipt/profile material present in the reviewed AEP snapshot. The article does not claim a complete audit of any independent OVERT implementation or service.

**Table 1** Reader map for artifact labels

Label form	Plain meaning	Where it belongs
S-0001, S-0002, ...	Reviewed source snapshots used by this manuscript.	Evidence-source list and future public artifact record.
01-auth, 07-replay, ...	Tamarin query-output filenames reserved for the artifact companion.	Results table and future query-output artifact directory.
E-0001-E-0004	Short errata proposal IDs for trust-anchor, replay, claim-set, and compatibility-artifact clarifications.	Errata discussion below and future artifact companion.

## 4 Model Overview

The Tamarin theory represents agents, gateways, signers, timestamp authorities, ledgers, verifiers, and a Dolev–Yao adversary. Cryptographic primitives are idealized using Tamarin signing and hashing built-ins. Enrollment is modeled explicitly as key material and public-key publication facts. The verifier acceptance rule checks a signer signature over a canonical term, a timestamp token over the package hash, and a ledger anchor over the same hash. Corruption rules reveal agent, signer, timestamp, and ledger keys.

The first lemma file duplicates the core theory to keep proof runs self-contained. This keeps each proof run easy to inspect, at the cost of duplicated model rules. For readability, the main claims can be sketched as trace formulas. These sketches are not replacement Tamarin syntax.

*Accepted evidence needs signer history.*

$$A(h, i) \Rightarrow (\exists j < i. S(h, j)) \vee K_s(i)$$

If the verifier accepts package hash  $h$ , the model must find an earlier signer issuance for the same package, unless the signer key had already been compromised.

*Accepted content matches canonical bytes.*

$$A(p, m, h, i) \Rightarrow h = H(C(p, m))$$

Here  $p$  is the payload,  $m$  is metadata,  $C(\cdot)$  means canonicalization, and  $H(\cdot)$  means hashing. If the verifier accepts a payload and metadata under hash  $h$ , then  $h$  must be the hash of exactly those canonical bytes.

*The tempting replay claim.*

$$A(h, i) \wedge A(h, j) \Rightarrow i = j$$

A package hash  $h$  should not create two separate accept events. This is the replay-resistance intuition, and it is the lemma that fails for the stateless verifier.

*The same claim with verifier memory.*

$$A_{\text{cache}}(h, i) \wedge A_{\text{cache}}(h, j) \Rightarrow i = j$$

Once the verifier remembers that package  $h$  has already been accepted for this profile, a second fresh acceptance should not appear. Tamarin verifies this cache-backed statement for the replay-cache model.

## 5 Reproducibility and Proof Outputs

The verification evidence is preserved as files rather than copied wholesale into the article. When the artifact companion is deposited, it will include the relevant Tamarin output files so later readers can inspect the emitted theory, derivation-check result, proof status, and toolchain banner. The runnable commands belong in the reproduction notes and query artifacts; the article records the reader-facing contract in Table 2.

The preserved baseline proof outputs were produced under Tamarin 1.12.0 in an environment where the captured banner reported Maude 3.2 as unsupported by Tamarin’s preferred-version list. Maude is the rewriting-logic engine used by Tamarin; version differences matter because proof outputs should be interpreted with the exact toolchain recorded. The installation check, minimal sanity-check proof, and subsequent proof runs still completed, so that warning remains part of the historical baseline record.

A later rerun used a supported Maude 3.5.1 configuration and preserved its outputs with the reproduction materials. That rerun removes the unsupported-Maude warning for the corrected proof commands and reproduces the selected positive lemma outcomes. Because the historical baseline and the supported rerun differ in which lemmas were selected and how they were named, they are comparable as result checks rather than byte-for-byte identical output files.

## 6 Abstractions and Limitations

Time is not arithmetic in the current model. Freshness is represented by successful timestamp verification and

**Table 2** Proof-output artifacts intended for the companion record

Stage	Captured artifact in companion	Reader check
Tool startup check	<b>bootstrap</b> output files: Tamarin startup plus a minimal sanity-check proof.	Tamarin starts and can prove a minimal theory.
Theory parse check	<b>aep-parse</b> output: parse and derivation check for the core AEP theory.	The core AEP theory is syntactically well formed.
Claim proofs	Eight query outputs: <b>01-auth</b> , <b>01-auth-agent-authorization</b> , and <b>02-integrity</b> through <b>07-replay</b> .	Each claim area has a separate captured output.
Repair check	<b>07-replay-cache</b> output: the replay-cache variant of the model.	The cache variant verifies uniqueness and remains executable.
Compatibility checks	<b>16-overt-receipt-binding</b> , <b>20-overt-package-binding</b> , and <b>23-combined-verifier-policy</b> outputs.	Optional OVERT and combined-artifact claims are checked only under explicit verifier obligations.

the existence of a timestamping event, not by proving a numeric  $\pm 2$  second skew bound. The ledger is a symbolic anchor and chain link, not a full Merkle-tree membership proof. The trust-anchor relation between embedded public keys, signer certificates, **agent\_id**, **agent\_uri**, **gateway\_id**, and enrolled identities is an explicit model assumption. Section 10 names the corresponding clarification proposal as trust-anchor binding, with short artifact ID E-0001.

The arithmetic-time limitation matters because the AEP and MCP drafts discuss concrete clock behavior: timestamp evidence, gateway-created times, and local drift bounds. Tamarin’s trace order can show that one event precedes another, but it does not by itself express real-number or integer timestamp comparison. The current freshness lemma therefore supports only the statement that accepted packages are tied to a timestamping event under the model’s **Fresh** precondition. It does not support evidence for a particular skew window, clock-synchronization policy, revocation cutoff, or long-term archival validation interval.

The ledger abstraction is similarly narrow. The model binds verifier acceptance to a symbolic ledger anchor over the same package hash, which is enough to study replay and content-binding questions. It does not model block production races, omitted events, tenant partitioning failures, Merkle inclusion proofs, ledger-signing-key rotation, or equivocation between two ledger views. A stronger future analysis should separate inclusion of a package hash in one authenticated ledger statement, append-only consistency of a ledger sequence, and non-equivocation for verifiers that may see different ledger replicas.

The trust-anchor abstraction is the most important authentication caveat. The lemmas assume an enrolled mapping between identities and keys; the model does not derive that mapping from embedded package material alone. If a verifier accepted arbitrary embedded public keys without a pinned enrollment record, the

Tamarin authentication premise would be stronger than the verifier behavior and the positive lemmas would no longer describe that implementation.

## 7 Results

Table 3 summarizes the model results using the query-output labels reserved for the artifact companion. These labels are file lookup handles; the right-hand column gives the reader-facing interpretation.

The positive lemmas should be interpreted as a chain of model-bounded event correspondences, not as an unrestricted statement that AEP is secure in every deployment. The first authentication lemma starts at verifier acceptance and the verifier’s checked signer signature. In the model, that event must correspond either to a prior signer-issuance event for the same agent, payload, metadata, and package hash, or to the explicit exception that the signer key was revealed.

The second authentication lemma connects signer issuance back to the agent. A signer-issued package must follow an agent-authorization event over the same payload and context, unless the agent key has been revealed. This captures the MCP gateway boundary: the gateway may forward material, including through the corrupt-gateway rule, but it does not acquire a symbolic ability to synthesize an honest agent signature.

The integrity lemma states that the payload and metadata accepted by the verifier equal the payload and metadata in the canonical component checked under the signer signature. It is not about ZIP parser behavior, JSON canonicalization bugs, or alternate encodings; those are outside the symbolic abstraction. Within the model, however, a Dolev–Yao adversary cannot make the verifier accept one claims payload while relying on a signature over a different canonical payload.

The non-repudiation and forward-integrity lemmas are best read as conditional accountability lemmas. Ac-

**Table 3** Model results and reader-facing interpretation

Claim area	Artifact query label	Result and reader meaning
Signer authentication	<b>01-auth</b>	Verified: acceptance links back to signer issuance, except signer-key reveal.
Agent authorization	<b>01-auth-agent-authorization</b>	Verified: issuance needs matching agent authorization, except agent-key reveal.
Integrity	<b>02-integrity</b>	Verified: accepted payload and metadata match signed canonical bytes.
Timestamp linkage	<b>03-freshness</b>	Verified: accepted timestamps link to a timestamping event, except timestamp-authority key reveal.
Accountability	<b>04-nonrepudiation</b>	Verified: acceptance links to prior agent authorization, except compromise cases.
Forward integrity	<b>05-forward-integrity</b>	Verified: later agent-key reveal alone does not explain earlier acceptance.
Gateway compromise	<b>06-gateway</b>	Verified: a corrupt gateway is not an honest-agent signing oracle.
Replay uniqueness	<b>07-replay</b>	Falsified: a stateless verifier can accept the same valid package twice.

ceptance links back to prior agent authorization unless signer-key or agent-key compromise is admitted. For forward integrity, the modeled question is whether a later agent-key reveal alone can explain an earlier verifier acceptance. Tamarin verifies that it cannot.

## 8 Replay Counterexample and Cache Variant

The replay-resistance result is the clearest negative result in the current model. The falsified lemma asked for uniqueness of verifier acceptance per package hash: if two acceptance events occur for the same package hash, then they should be the same event. Tamarin found a 22-step trace that violates this property. An enrolled agent authorizes a payload, the gateway forwards it, the signer issues a package, the timestamp authority signs a timestamp token over the package hash, and the ledger signs an anchor over the same hash. The adversary does not forge any cryptographic value. Instead, the adversary replays the already valid package to the verifier twice. Because the current verifier rule is stateless, the signature, timestamp, and ledger checks succeed on both presentations, producing two distinct accept events for the same package hash.

Put plainly, the failed case is not that the attacker edits the package. The failed case is that the verifier treats the same already-valid package as two fresh accept events. This does not contradict the integrity or authentication lemmas. It shows that content authentication and duplicate-acceptance detection are different properties. A timestamp token says that a package existed at a time under the timestamping abstraction. A ledger anchor says that a package hash was anchored under the ledger abstraction. Neither fact consumes verifier state.

The follow-up replay-cache model encodes one candidate repair for a future verifier profile. It adds a linear unseen-package token emitted when the package is produced and consumed by verifier acceptance. Tamarin linear facts are consumed when a rule uses them; this is useful for modeling stateful protocols such as a replay cache that accepts a package identifier once and then removes the unseen state. The token is not intended to claim that AEP literally creates global single-use state at issuance time. It is a compact Tamarin abstraction of an operational verifier replay cache.

Under this added state, Tamarin verifies the replay-cache uniqueness lemma, so two accept events for the same package hash collapse to the same event in the model. A separate executability lemma also verifies, indicating that the repair does not make honest acceptance impossible.

## 9 OVERT Compatibility and Optional Artifacts

The replay result is about accepting the same signed package twice. The OVERT compatibility checks ask a neighboring question: when an AEP-compatible package carries optional receipt or manifest material, does the verifier bind the optional material to the same package context that it is accepting? The answer matters because optional artifacts can start as helpful metadata and quietly become claim-bearing evidence.

The focused OVERT receipt model verifies that an OVERT-aware acceptance can require the receipt content hash to match the accepted canonical MCP hash and can require witness references to be present. That is the easy case: the verifier mode says it is OVERT-aware and records the receipt checks at acceptance time.

**Table 4** Verifier profiles give replay different meanings

Verifier profile	Replay behavior	Example
Evidence verification	Re-check allowed; repeated presentation should be reported rather than rejected.	An auditor opens the same AEP package twice.
Action authorization	Duplicate presentation should be rejected or treated as an idempotent no-op; a cache or nonce is required.	An agent payment or consequential tool call.
Workflow step	Acceptance should be bound to a workflow identifier, step identifier, and state hash.	A procurement, HR, or legal-review transition.
Ledger publication	The same package can be read many times, but it should not create a second publication event.	An evidence-bundle ledger that records one package once.
Observatory	Repeated verification is expected, but duplicate observations should not advance the time series as new data.	A periodic standards, trust-list, or citation-observation dataset.

The package-binding model shows the sharper risk. If two packages carry the same canonical MCP content hash, an unbound verifier path can accept package A while using an OVERT receipt whose package identity and witness references came from package B. Tamarin verifies that this recombination trace is executable and falsifies the lemma that the receipt package must match the accepted package. The bound repair pattern includes the OVERT receipt digest in the signed core and records same-step checks of package identity and witness references; Tamarin verifies the corresponding repaired lemmas.

The combined verifier-policy model generalizes the point. A package can declare that manifest, disclosure, and OVERT checks are required, while an unbound verifier still emits a combined claim-bearing acceptance after checking only a subset of them. The repaired model binds the manifest, disclosure, and OVERT receipt digests into the accepted core and records all required same-step checks before emitting the combined acceptance event.

These compatibility checks do not change the main replay conclusion. They do, however, make the same design rule harder to miss: AEP-compatible verifier profiles should state which optional formats are merely carried, which are checked, and which fields become claim-bearing. A compatibility profile is itself a small standard and needs the same kind of formal checking as the base package profile.

## 10 Errata Implications

In this article, “errata” means a proposed clarification for a future AEP profile, not a live standards change. The future artifact companion will use four short lookup IDs for these proposals.

The trust-anchor proposal is tied to the model’s explicit enrollment assumptions. The positive authentication lemmas do not derive identity binding from package-contained key material alone. Instead, the verifier rule uses an enrolled signer key, and the signer rule checks an enrolled agent key before issuing a package. The proposal therefore makes the model premise visible at the specification layer: verifier acceptance should require a trust list, registry, pinned mirror, certificate path, or equivalent enrollment lookup that binds key identifiers to permitted subjects and scopes.

The replay-semantics proposal is tied to the failed uniqueness lemma and to the hardened replay-cache variant. The falsified replay-resistance lemma shows that the current verifier rule is content-checking but stateless: a valid package can satisfy the same signature, timestamp, and ledger checks more than once. The replay-cache variant adds a linear single-use abstraction for package hashes and verifies uniqueness under that added state.

The claim-set and OVERT-compatibility proposals are tied to optional artifacts. The positive integrity lemma covers the signed canonical core, not every adjacent file that may travel with a package. The compatibility models therefore treat optional receipts, manifests, disclosure files, and witness references as claim-bearing only when a verifier profile explicitly checks them or binds their digests into the accepted core.

## 11 Artifacts and Reproducibility Package

The related research materials include Tamarin theories, captured query outputs, replay trace notes, the AEP extraction note, errata proposals, SHA-256 indexes, source snapshots, and a reproduction-output comparison helper. The helper is a script for comparing future captured Tamarin outputs with preserved baseline result summaries. These files are research artifacts, not a regulated

**Table 5** Errata proposals mapped to model assumptions

Local ID	Human name	Plain proposal
E-0001	Trust-anchor binding	A verifier should not trust package-contained public keys by themselves; it should bind keys to enrolled identities, allowed agents, signer authority, gateway authority, scope, and expiry or revocation state.
E-0002	Replay semantics	A profile should say whether repeated presentation is allowed, reported, rejected, converted to an idempotent no-op, or bound to a verifier or application nonce.
E-0003	Claim-set boundary	A profile should name the exact signed claim set for each verifier mode and state which optional entries are integrity-critical.
E-0004	OVERT compatibility binding	OVERT-aware and combined optional-artifact profiles should bind claim-bearing receipt, package, witness, manifest, and disclosure fields to the accepted evidence chain.

trust-service output, certificate, timestamp, or external audit record.

A public artifact deposit has not yet been made for this IJIS package. Before external journal submission as reproducible original research, the related models, query outputs, trace notes, and helper scripts need a separate artifact record with metadata, license choice, file manifest, and source-rights review. The preferred artifact route is a Zenodo software/data record because Zenodo records can carry mixed files, metadata, DOI assignment, and DOI versioning [32,33]. The local package prepared with this manuscript follows the same practical logic as FAIR and artifact-availability guidance: name the files, preserve the outputs, expose enough metadata for inspection, and keep public artifact availability distinct from independent reproduction [28,29,30]. If a public source repository is later released, Software Heritage can complement Zenodo with source-code preservation and Software Heritage identifiers; it is not a replacement for depositing the article’s mixed artifact bundle.

## 12 Conclusion

Under the stated symbolic abstraction, AEP’s core signature and verifier-check structure supports evidence for authentication, integrity, accountability, forward-integrity-style pre-compromise attribution, and gateway-compromise resistance. The same model does not support a replay-resistance uniqueness claim unless verifier replay state or challenge-bound uniqueness is added. The optional OVERT and combined-artifact checks point in the same direction: compatibility formats should not be treated as trusted evidence merely because they travel next to a signed package. These results should inform a future AEP errata discussion, not be read as a deployment-wide security guarantee or a legal compliance conclusion.

The architectural lesson is simple: AEP profiles should say, in ordinary verifier language, when re-reading

evidence is allowed, when duplicate presentation is a reportable observation, when a duplicate must be rejected or converted into an idempotent no-op, and which optional compatibility artifacts support which claims. That is the simplest way to keep the formal result useful for engineers rather than leaving it as a theorem about an artificial trace.

## Statements and Declarations

*Funding* No external funding is declared.

*Competing interests* No competing financial interests are declared. The work is presented as research by Anton Sokolov at Tyche Institute and does not claim trust-service, conformity-assessment, supervisory, or legal-advice status. The AEP materials analysed here are part of the author’s Tyche Institute research programme and are treated as author-controlled research artifacts, not as third-party standards, adoption evidence, certification, regulatory approval, or trust-service operation.

*Author contributions* Anton Sokolov is the sole author and is responsible for conceptualization, methodology, formal analysis, software/model preparation, investigation, writing, review, and final submission decisions.

*Generative-AI assistance* The author used generative-AI assistance for final draft polishing. The author remains responsible for all claims, results, design decisions, source choices, evidence boundaries, references, and final submission choices. Reported per COPE and ICMJE recommendations.

*Data availability* The Tamarin theories, captured query outputs, replay trace notes, source snapshots, extraction note, errata proposals, SHA-256 indexes, compatibility checks, and comparison helper have been staged as a local Zenodo-ready artifact package. The DOI placeholder

for the future public record is [Zenodo DOI for IJIS artifact companion]. Until the author completes the external deposit and license selection, this IJIS file remains a format-conversion candidate rather than a complete journal-submission package.

## References

1. Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. <https://doi.org/10.1109/TIT.1983.1056650>.
2. Gavin Lowe. Breaking and fixing the Needham–Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166. Springer, 1996. [https://doi.org/10.1007/3-540-61042-1\\_43](https://doi.org/10.1007/3-540-61042-1_43).
3. Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *Computer Aided Verification*, pages 696–701. Springer, 2013. [https://doi.org/10.1007/978-3-642-39799-8\\_48](https://doi.org/10.1007/978-3-642-39799-8_48).
4. The Tamarin Prover Team. The tamarin prover. <https://tamarin-prover.com/>. Accessed 30 May 2026.
5. Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3:99–111, 1991. <https://doi.org/10.1007/BF00196791>.
6. IETF. RFC 3161: Internet X.509 public key infrastructure time-stamp protocol. <https://www.rfc-editor.org/info/rfc3161>, 2001.
7. IETF. RFC 8785: JSON canonicalization scheme. <https://www.rfc-editor.org/info/rfc8785>, 2020.
8. IETF. RFC 9334: Remote ATtestation procedureS architecture. <https://www.rfc-editor.org/info/rfc9334>, 2023.
9. IETF. RFC 9711: The entity attestation token. <https://www.rfc-editor.org/info/rfc9711>, 2025.
10. IETF. RFC 9162: Certificate transparency version 2.0. <https://www.rfc-editor.org/info/rfc9162>, 2021.
11. IETF. RFC 3647: Internet X.509 public key infrastructure certificate policy and certification practices framework. <https://www.rfc-editor.org/info/rfc3647>, 2003.
12. IETF. RFC 5280: Internet X.509 public key infrastructure certificate and CRL profile. <https://www.rfc-editor.org/info/rfc5280>, 2008.
13. ETSI. EN 319 401: General policy requirements for trust service providers. [https://www.etsi.org/deliver/etsi\\_en/319400\\_319499/319401/03.01.01\\_60/en\\_319401v030101p.pdf](https://www.etsi.org/deliver/etsi_en/319400_319499/319401/03.01.01_60/en_319401v030101p.pdf), 2024.
14. ETSI. TS 119 612: Trusted lists. [https://www.etsi.org/deliver/etsi\\_ts/119600\\_119699/119612/02.03.01\\_60/ts\\_119612v020301p.pdf](https://www.etsi.org/deliver/etsi_ts/119600_119699/119612/02.03.01_60/ts_119612v020301p.pdf), 2025.
15. European Union. Regulation (EU) no 910/2014 on electronic identification and trust services. <https://eur-lex.europa.eu/eli/reg/2014/910/oj>, 2014.
16. European Union. Regulation (EU) 2024/1183 establishing the european digital identity framework. <https://eur-lex.europa.eu/eli/reg/2024/1183/oj>, 2024.
17. European Union. Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence. <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>, 2024.
18. CA/Browser Forum. Baseline requirements for the issuance and management of publicly-trusted TLS server certificates. <https://cabforum.org/working-groups/server/baseline-requirements/requirements/>. Accessed 30 May 2026.
19. PKI Consortium. Post-quantum cryptography working group. <https://pkic.org/wg/pqc/>. Accessed 30 May 2026.
20. National Institute of Standards and Technology. FIPS 203: Module-lattice-based key-encapsulation mechanism standard. <https://csrc.nist.gov/pubs/fips/203/final>, 2024.
21. National Institute of Standards and Technology. FIPS 204: Module-lattice-based digital signature standard. <https://csrc.nist.gov/pubs/fips/204/final>, 2024.
22. National Institute of Standards and Technology. FIPS 205: Stateless hash-based digital signature standard. <https://csrc.nist.gov/pubs/fips/205/final>, 2024.
23. Susan Leigh Star and Karen Ruhleder. Steps toward an ecology of infrastructure: Design and access for large information spaces. *Information Systems Research*, 7(1):111–134, 1996. <https://doi.org/10.1287/isre.7.1.111>.
24. Geoffrey C. Bowker and Susan Leigh Star. *Sorting Things Out: Classification and Its Consequences*. MIT Press, 1999. ISBN 9780262522953.
25. Michael Power. *The Audit Society: Rituals of Verification*. Oxford University Press, 1997. ISBN 9780198296034.
26. Lawrence Lessig. *Code and Other Laws of Cyberspace, Version 2.0*. Basic Books, 2006. ISBN 9780465039142.
27. Carlota Perez. *Technological Revolutions and Financial Capital: The Dynamics of Bubbles and Golden Ages*. Edward Elgar, 2002. ISBN 9781840649222.
28. Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Merce Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J. G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A. C. 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018, 2016. <https://doi.org/10.1038/sdata.2016.18>.
29. Stian Soiland-Reyes, Peter Sefton, Merce Crosas, Leyla Jael Castro, Frederik Coppens, Jose M. Fernandez, Daniel Garjito, Bjoern Gruening, Marco La Rosa, Simone Leo, Eoghan O Carragain, Marc Portier, Ana Trisovic, RO-Crate Community, Paul Groth, and Carole Goble. Packaging research artefacts with RO-Crate. *Data Science*, 5(2):97–138, 2022. <https://doi.org/10.3233/DS-210053>.
30. Association for Computing Machinery. Artifact review and badging, version 1.1. <https://www.acm.org/publications/policies/artifact-review-and-badging-current>, 2020. Accessed 30 May 2026.
31. Anton Sokolov. EATF agent evidence package artifact bundle. Zenodo, 2026. Concept DOI <https://doi.org/10.5281/zenodo.20273730>; version 0.2 DOI <https://doi.org/10.5281/zenodo.20396464>.
32. Zenodo. About records. <https://help.zenodo.org/docs/deposit/about-records/>. Accessed 30 May 2026.
33. Zenodo. What is DOI versioning? <https://support.zenodo.org/help/en-gb/1-upload-deposit/97-what-is-doi-versioning>. Accessed 30 May 2026.