

A Symbolic Analysis of Agent Evidence Packages

Preprint note: this is an open research preprint and reproducibility note for a reusable formal model of Agent Evidence Package (AEP) replay semantics. It studies an AEP profile from the Agent Trust Framework (EATF) and a Model Context Protocol (MCP) attestation profile with open formal-methods tooling. The manuscript is intended for deposit alongside a public artifact companion: a separate record containing the Tamarin model files, proof outputs, readable replay-trace notes, reviewed source snapshots, compatibility checks, and a manifest.

Anton Sokolov

Tyche Institute, Tallinn, Estonia

ORCID: 0000-0003-2452-7096

Abstract

This preprint studies an Agent Evidence Package (AEP)¹ profile from the Agent Trust Framework (EATF)² and an MCP Attestation Profile³ using symbolic protocol modeling in Tamarin.⁴ 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-Yao⁵ 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 result 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; verifier state; symbolic model.

¹Agent Evidence Package. In this paper, an AEP is the evidence-container profile introduced at the beginning of the manuscript: it binds canonical bytes, hashes, signatures, timestamp material, ledger references, and verifier instructions. Public AEP artifact-family records include Zenodo concept DOI 10.5281/zenodo.20273730 and v0.2 DOI 10.5281/zenodo.20396464; exact source snapshots reviewed here remain identified in the manuscript’s artifact source snapshot list. It is evidence infrastructure, not by itself a legal compliance certificate.

²EATF, the Agent Trust Framework, is the surrounding Tyche Institute research framework for agent evidence packages, verifier outputs, fixtures, and claim boundaries. It is a research framework, not a trust service, conformity-assessment body, or certification scheme.

³Model Context Protocol. MCP is the agent/tool integration setting used here to model a gateway between an agent authorization and downstream signing or verification. The paper treats it as an architecture role, not as an endorsement by any platform provider.

⁴Tamarin Prover is a formal-methods tool for symbolic analysis of security protocols. It asks whether a property follows in all traces of an idealized protocol model, or whether an adversarial trace violates it.

⁵The Dolev-Yao model is the standard “strong network attacker” abstraction used in symbolic protocol analysis: the attacker can intercept, replay, and construct messages from known material, but cannot break ideal cryptographic primitives.

Terminology note

Non-obvious abbreviations are expanded at first mention and the hardest terms are also footnoted. The short version is that EATF is the surrounding research framework, AEP is its portable evidence-container profile, MCP is the agent/tool gateway setting studied here, Tamarin is the formal-analysis tool, a verifier profile is the rule set that says what verifier acceptance means,⁶ and replay resistance is the property that a valid old package cannot be silently treated as a new fresh action.

1. Scope and research questions

The analysis focuses on the core flow from agent to MCP gateway, signer, ledger, and verifier. It asks four questions:

- Can verifier acceptance be linked to prior honest authorization and signing?
- Do accepted payloads match their signed canonical form?
- Do accepted timestamps derive from a modeled timestamping event?
- Does later agent-key compromise explain earlier verifier acceptance?

It also tests whether a corrupted MCP gateway can insert claims attributed to an honest agent and whether freshness plus ledger anchoring is enough to detect replay.

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.⁷ One generated test prompt treated a signed package as a reusable 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.⁸

Out of scope: implementation bugs in ZIP handling, JSON canonicalization, ASN.1 parsing, RSA or ML-DSA cryptanalysis, operational key management, and production deployment controls. These are important, but this report only states what follows inside the symbolic model.

2. Source material and extraction method

Primary sources are the AEP profile snapshot, the MCP Attestation Profile snapshot, and the MIRROR bundle schema.⁹ The extraction note is a short design record, to be published 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.

⁶A verifier profile is the application-facing contract for acceptance: content-authentic archival evidence, online one-time authorization, workflow-state transition, ledger publication, and observatory deduplication can all inspect the same package while requiring different replay behavior.

⁷Here “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.

⁸The 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.

⁹MIRROR is a manifest-and-hash evidence-bundle schema used here to make claims, sources, and artifacts reviewable together. In this manuscript it is treated as a reviewed schema snapshot for the artifact companion, not as an external standard.

The reviewed AEP snapshot also contains an optional OVERT receipt entry and OVERT-aware verifier obligations.¹⁰ This preprint’s main replay result concerns base AEP/MCP acceptance, but the artifact set includes focused OVERT compatibility checks as well. 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?

Some short labels in this paper are artifact labels, not new concepts. They are included so a reader can later find the exact source or proof file in the companion artifact:

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, ...	Artifact-companion Tamarin query-output filenames.	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.

2.1 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, classic attack-synthesis work such as Lowe’s Needham-Schroeder analysis, and modern tools such as Tamarin for correspondence lemmas and adversarial trace search.¹¹ Second, attestation and transparency infrastructure gives verifiers signed evidence about systems, keys, timestamps, and logs. The relevant standards background spans timestamping work and RFC 3161 for timestamp tokens, RFC 8785 for JSON canonicalization, RFC 9334 for remote-attestation architecture, and RFC 9711 for attestation-token structure; RFC 9162 Certificate Transparency is background for the ledger-anchoring abstraction. Long-lived evidence also sits near legal trust-service rules, AI documentation duties, certificate governance, and post-quantum migration work. Third, infrastructure studies explain why technical checks become operational institutions only when their claim boundaries are visible. Reproducibility practice adds a practical pressure: artifacts need durable identifiers, metadata, and enough structure for later readers to inspect what was actually run. That is why this paper treats replay as a verifier-profile problem, not merely as a cryptographic primitive problem.

The practical question is simple: when a verifier sees the same valid-looking package twice, is the second acceptance another evidence check or a new action? Signatures and timestamps alone do not answer that question. The answer lives in the verifier profile.

The contribution is modest by design. It does not propose a new cryptographic scheme or prove deployment-wide security for any AEP implementation. It is a small case study showing that “authentic evidence” and “fresh action authorization” are separate acceptance questions. A package can be safe to re-read for archival evidence and unsafe to reuse as a second consequential action.

¹⁰OVERT is treated here as the optional receipt/profile material present in the reviewed AEP snapshot. The manuscript does not claim a complete audit of any independent OVERT implementation or service.

¹¹In this paper, “correspondence” means a trace relation such as “if a verifier accepts this package, then a matching signing or authorization event occurred earlier, unless a modeled compromise exception applies.” It is not a statistical correlation claim.

3. Reproducibility and proof-output handling

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 below.

The labels in the middle column are lookup handles for the future artifact deposit, not new theoretical concepts:

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

The preserved baseline proof outputs were produced under Tamarin 1.12.0 in an environment where the captured banner reported Maude 3.2¹² as unsupported by Tamarin’s preferred-version list. 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.

Proof outputs are treated as artifacts, not as transient console logs. The artifact companion, once deposited, separates three layers: proof-output files with Secure Hash Algorithm 256-bit (SHA-256) digests, a claim register with cautious result summaries, and prose trace notes for falsified lemmas. Verified universal lemmas are recorded as model-bounded support for the corresponding property. Falsified lemmas are therefore not left as a bare tool status; their traces are translated into a human interpretation, while specification ambiguities and repair candidates are kept as research errata. This separation prevents the report from overstating a Tamarin result: the query output is the machine artifact, the trace note is the human interpretation, and the errata entry is only a research proposal for a possible future AEP change.

¹²Maude is the rewriting-logic engine used by Tamarin. Version differences matter because proof outputs should be interpreted with the exact toolchain recorded.

4. Model overview

The Tamarin theory represents agents, gateways, signers, timestamp authorities (TSAs),¹³ ledgers, verifiers, and a Dolev-Yao adversary. Cryptographic primitives are idealized using Tamarin signing and hashing builtins. 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 written as small trace formulas. These are reader-facing sketches, not replacement Tamarin syntax. The notation is deliberately plain:

Symbol	Plain meaning
h	the package hash, the short fingerprint of the evidence package
i, j	moments in the modeled trace; $j < i$ means “j happened before i”
$A(h, i)$	the verifier accepts package hash h at moment i
$S(h, j)$	the signer issued package hash h at moment j
$K_s(i)$	the signer key was already compromised before moment i

Lemma 1: accepted evidence needs a signer history.

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

Plain meaning: if the verifier accepts package h , then the model must find an earlier signer issuance for the same package, unless the signer key had already been compromised. In human terms, acceptance should have a history; it should not appear from nowhere.

Lemma 2: accepted content must match the 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. Plain meaning: if the verifier accepts a payload and metadata under hash h , then h must be the hash of exactly those canonical bytes. Changing the bytes should change the claim.

Lemma 3: the tempting replay claim.

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

Plain meaning: 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.

Lemma 4: the same claim with verifier memory.

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

¹³Timestamp authority. A TSA provides a cryptographic timestamp token, commonly associated with RFC 3161 Time-Stamp Protocol evidence. In this model it proves a package existed in the modeled timestamping event, not that every real-world time-policy rule has been checked.

Plain meaning: 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. The mathematical difference is tiny; the architectural difference is the whole case study.

5. Abstractions and limitations

Time is not arithmetic in the current model. Freshness is represented by successful timestamp verification and the existence of a timestamping event, not by proving a numeric ± 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 9 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. Future work should either keep this as an explicit verifier-side assumption or add a small dedicated model for time-window checks outside Tamarin, for example with a bounded arithmetic test harness over verifier policy inputs.

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 at least three ledger properties: 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 present results should therefore be read as evidence about package-to-anchor binding, not as a complete ledger-consensus or transparency-log proof.

The trust-anchor abstraction is the most important authentication caveat. The lemmas assume an enrolled mapping between identities and keys; the report does not derive that mapping from the embedded package material. That is why the trust-anchor errata proposal is authentication-critical rather than editorial. If the specification permits a verifier to trust any embedded key without a pinned enrollment record, then the Tamarin authentication premise would be stronger than the verifier behavior and the positive lemmas would no longer describe that implementation. Future versions should make the enrollment lookup explicit enough to be modeled as data: accepted key identifiers, certificate subjects, allowed agent identifiers, gateway authority, delegation scope, and revocation or expiry state.

6. Current results table

Table 1 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 result.

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

7. Interpretation of the positive lemmas

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 either correspond 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. This supports evidence for the signer side of the chain only because the verifier rule uses an enrolled signer key and checks the canonical term under that key. It does not remove the need for the future trust-anchor clarification: if a real verifier accepted arbitrary embedded public keys, the formal enrollment premise would no longer describe the implementation.

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 is the part of the model that 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 result therefore supports evidence for attributed agent authorization only under the explicit assumptions that the agent signing key remains uncompromised and that the signer verifies the agent authorization term before issuing the package.

The integrity lemma is narrower but important. It states that the payload and metadata accepted by the verifier equal the payload and metadata in the canonical component checked under the signer signature. The lemma 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. This separates content-integrity support from the replay result: a replayed package can still be content-authentic.

The non-repudiation and forward-integrity lemmas are best read as conditional accountability lemmas. Acceptance links back to prior agent authorization unless signer-key or agent-key compromise is admitted. For forward integrity, the specific modeled question is whether a later agent-key reveal alone can explain an earlier verifier acceptance. Tamarin verifies that it cannot: an accepted pre-compromise package still requires earlier authorization, unless the signer key was already compromised before acceptance. This is weaker than operational non-repudiation, because it does not model key custody, human intent, certificate revocation, or legal evidentiary standards.

Finally, the two gateway-compromise lemmas show that a malicious MCP gateway is a transport and selection adversary in the model, not a signing oracle for an honest agent. If the corrupt gateway forwards a claim that the signer accepts as an agent authorization, Tamarin requires either a matching earlier agent-authorization event or agent-key compromise. At the acceptance level, the same dependency reappears with the additional signer-key compromise exception. This supports the cautious claim that AEP/MCP can preserve agent-attribution evidence across an untrusted gateway when the signer verifies agent authorization and verifier trust anchors are explicit.

8. Replay counterexample and hardened replay-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. The trace is intentionally mundane. 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 “the attacker edits the package.” The failed case is “the verifier treats the same already-valid package as two fresh accept events.”

What still holds	What failed	Why it matters
Signatures, timestamps, and ledger anchors still bind the package content.	Duplicate acceptance is not detected by a stateless verifier.	A receipt may be safe to re-read, but unsafe to reuse as a new command.
The Dolev-Yao adversary does not break ideal cryptography.	The adversary can replay known valid material.	Replay is a protocol-semantics issue, not a hash-function issue.
Archival verification can intentionally reopen the same package.	Online action authorization needs memory, nonce, or idempotency.	AEP profiles should say which kind of verifier they are.

This result should be read narrowly. It does not indicate that package contents can be modified undetected, and it 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. If a downstream application treats every verifier acceptance as a fresh authorization trigger, counter, payment event, workflow transition, or audit observation, then accepting the same hash twice may be security-relevant even when the package is otherwise valid.

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.¹⁴ This 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: a verifier that needs replay resistance tracks accepted package identifiers and permits first acceptance only. 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.

One artifact caveat is important for the supported-toolchain rerun. When deposited, the artifact companion will include the baseline replay output as `07-replay.out`; that output records the weaker stateless-verifier counterexample described above. A later supported rerun also produced an output with the same filename, but that rerun selected a replay-cache uniqueness lemma from the stateful repair model and verified it under Maude 3.5.1. The two outputs therefore need to be compared by selected lemma and model semantics, not by filename alone. The companion trace note, `07-replay.md`, records this distinction as a prose interpretation.

The model-bounded interpretation of the replay-semantics proposal is therefore conditional. The current AEP profile draft supports evidence for package integrity and timing under the stated assumptions, but the current abstraction does not derive replay detection from freshness and ledger anchoring alone. A future AEP profile can choose between at least two precise semantics: local duplicate detection using a cache keyed by package hash, timestamp serial, ledger anchor, or equivalent package identifier; or stronger challenge-bound verification where a verifier nonce or application nonce is included in the signed canonical payload and checked for single use. The cache option is simpler and supports retrospective evidence verification. The challenge option is stronger for online authorization flows, but may not fit asynchronous evidence publication. The specification should state which behavior is required, optional, or application-profile dependent.

8.1 Case-study interpretation: authentic evidence is not always a fresh action

The replay result is also useful as a case study for evidence-package design. A signed and timestamped package is similar to a receipt: it can be inspected many times without becoming false. That behavior is desirable for archival verification, reproducibility review, and longitudinal observatories. The same behavior is unsafe if every verifier acceptance is treated as a fresh authorization event, such as a payment, tool execution, account change, workflow transition, or ledger publication. The core distinction is therefore not between “secure” and “insecure” packages, but between verifier profiles.

¹⁴Tamarin linear facts are consumed when a rule uses them. They are useful for modeling stateful protocols, such as a replay cache that allows a package identifier to be accepted once and then removes the “unseen” state.

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 id, step id, 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.

This profile view connects the formal result to evidence-infrastructure design more broadly. Public-key infrastructure (PKI),¹⁵ trust lists, timestamping, and certificate policy provide governance infrastructure: they say which keys and trust services a verifier may rely on. AEP-style packages add an evidence layer: they bind concrete bytes, policy decisions, timestamps, ledger anchors, and verifier outputs into portable artifacts. This case study shows that the evidence layer also needs verifier semantics. A package can be content-authentic while still being the wrong input for a second fresh action. Future AEP profiles should therefore expose replay status explicitly: package hash, verifier profile, validity, previous-observation status, policy identifier, timestamp result, ledger result, and the action to take on duplicates.

8.2 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.

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.

¹⁵Public-key infrastructure. PKI is the governance and technical system of keys, certificates, certificate policies, trust lists, and relying-party rules that lets a verifier decide which signatures can matter.

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.

8.3 Why this matters beyond this model

The broader point generalizes beyond any single research programme. Governance infrastructure says who may be trusted and under what policy. Evidence infrastructure records what was checked, when it was checked, under which verifier profile, and with what result. This research sits at the connection between those two layers.

$$\text{PKI} = \{\text{identity, authority, policy, status, time}\}$$

$$\text{AEP} = \text{PKI} + \{\text{canonical bytes, receipt, verifier profile, claim boundary}\}$$

This is deliberately not a shortcut around PKI. It is a layer above it. PKI answers “which key, certificate, timestamp, or trust service may matter to a verifier?” AEP asks a different question: “what exact evidence package was checked, and what is the verifier allowed to do after checking it?” That difference is why replay is not just an attacker trick. It is a meaning problem. The same receipt can be harmless when re-read and dangerous when treated as a second authorization.

9. Errata implications

In this paper, “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:

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/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.

All four are research proposals only. The current model identifies where a future AEP profile could clarify obligations; it does not establish any absolute deployment-wide safety conclusion.

9.1 Mapping errata proposals to model assumptions and lemmas

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. This affects the signer-authentication, agent-authorization, non-repudiation, and acceptance-level gateway-compromise lemmas. In those proofs, the relevant assumption is not merely that a signature verifies, but that the key used for verification is already bound to the intended signer or agent identity. The proposed errata 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 practical reading is conservative. If an implementation accepts any embedded `public_key.pem` or certificate without a pinned identity decision, then the model's authentication proof would be stronger than the implementation. The trust-anchor proposal therefore affects the preconditions for the positive authentication, non-repudiation, forward-integrity, and gateway-compromise interpretations. It does not refute those lemmas; it records the condition under which they are meaningful. A future AEP profile can reduce ambiguity by naming the verifier input that supplies enrolled identities, allowed agent identifiers, signer authority, gateway authority, delegation scope, and revocation or expiry state.

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, while a separate executability lemma confirms that the repair does not block an honest acceptance trace. The proposal therefore maps directly to a modeling delta: duplicate detection appears only when verifier state or challenge-bound uniqueness is added.

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.

This mapping also narrows what the replay-semantics proposal should and should not claim. It is not a claim that every AEP use case must reject every repeated package globally. Retrospective evidence verification may intentionally re-open and re-check the same package. The model result instead indicates that if a profile claims replay resistance for verifier acceptance events, the profile needs an explicit state or nonce semantics. A future profile text could make this profile-dependent: archival verification may report repeated observation of the same package identifier, while online authorization profiles should reject duplicate package identifiers or require a verifier/application nonce inside the signed canonical payload.

10. Artifacts and reproducibility package

The related research materials include the material needed to prepare a public artifact companion: 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 trust-service output, certificate, timestamp, or external audit record.

A public artifact deposit has not yet been made. Before external circulation as a reproducible preprint, the related models, query outputs, trace notes, and helper scripts need a separate artifact record with its own 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. 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. If a public source repository is later released, Software Heritage can complement Zenodo with source-code preservation and SWHIDs; it is not a replacement for depositing the paper’s mixed artifact bundle.

The reproducibility claim remains narrow until that deposit exists. Until a public artifact record exists, artifact file references and hashes identify author-reviewed materials, not public availability. Once a public artifact record exists, this section can name the exact DOI or SWHID-bearing record and replace deposit-pending references where appropriate.

11. Conclusion

The present analysis supports a cautious conclusion: under the stated symbolic abstraction, AEP’s core signature and verifier-check structure supports evidence for authentication, integrity, non-repudiation-style 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 are intentionally model-bounded. They should inform a future AEP errata discussion, not be read as a deployment-wide security guarantee or a legal compliance conclusion.

12. Future work

The next step is to separate three levels of evidence more cleanly. First, the paper needs a public artifact companion that contains the reviewed theories, outputs, traces, extraction note, errata proposals, and reproduction helper. Second, a clean-machine reproduction would rerun the selected lemmas under the supported Tamarin/Maude toolchain and record exactly which outcomes are reproduced. Third, future modeling can factor shared Tamarin rules into a stable module structure, add a small arithmetic time-window harness, and use a stronger ledger abstraction that separates inclusion, append-only consistency, and non-equivocation.

The research direction is also architectural. 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.

Declarations

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.

Funding. No external funding is declared.

Competing interests. No competing financial interests are declared here. 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 analyzed 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.

Data and materials. The related models, query outputs, trace notes, source snapshots, extraction notes, errata file, compatibility checks, and helper scripts are prepared as a local Zenodo-ready artifact bundle. Public release still requires source-rights, redaction, license, and platform-terms review by the author. A public artifact deposit has not yet been made by this manuscript or by the agent session that prepared it.

Artifact Source Snapshots

The S-* labels below identify the reviewed source snapshots used for this manuscript. They are source-evidence labels, not public bibliographic references. A public artifact DOI can be added here after the related Zenodo companion deposit is made; until then, the labels identify author-reviewed materials only.

- S-0001: AEP (Agent Evidence Package) Profile, v1 source snapshot.
- S-0002: MCP Attestation Profile, v1 source snapshot.
- S-0003: MIRROR Evidence Bundle Schema v0.1.0.
- S-0004: Tamarin Prover installation and bootstrap output.
- S-0005: Supported-toolchain Tamarin rerun.

References

Protocol analysis and cryptographic evidence:

- David Basin, Cas Cremers, Jannik Dreier, Simon Meier, Ralf Sasse, and Benedikt Schmidt, [The Tamarin Prover](#), official project documentation.
- Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin, “[The TAMARIN Prover for the Symbolic Analysis of Security Protocols](#)”, CAV 2013.
- Danny Dolev and Andrew C. Yao, “[On the Security of Public Key Protocols](#)”, IEEE Transactions on Information Theory, 1983.
- Gavin Lowe, “[Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR](#)”, TACAS 1996.
- Stuart Haber and W. Scott Stornetta, “[How to Time-Stamp a Digital Document](#)”, Journal of Cryptology, 1991.

PKI, timestamping, canonicalization, and attestation standards:

- IETF RFC 3161, [Internet X.509 Public Key Infrastructure Time-Stamp Protocol](#).
- IETF RFC 3647, [Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework](#).
- IETF RFC 5280, [Internet X.509 Public Key Infrastructure Certificate and CRL Profile](#).
- IETF RFC 8785, [JSON Canonicalization Scheme](#).
- IETF RFC 9162, [Certificate Transparency Version 2.0](#).
- IETF RFC 9334, [Remote ATtestation procedureS Architecture](#).
- IETF RFC 9711, [The Entity Attestation Token](#).

- ETSI EN 319 401, [General Policy Requirements for Trust Service Providers](#).
- ETSI TS 119 612, [Trusted Lists](#), used here as standards background for trusted-list observation.

Legal and institutional trust infrastructure:

- Regulation (EU) No 910/2014, [eIDAS](#), legal baseline for electronic identification and trust services.
- Regulation (EU) 2024/1183, [European Digital Identity Framework](#), amending eIDAS.
- Regulation (EU) 2024/1689, [Artificial Intelligence Act](#), context for AI documentation, logging, transparency, and post-market evidence demands.
- CA/Browser Forum, [Baseline Requirements for TLS Server Certificates](#), as an example of non-state PKI governance.
- PKI Consortium (PKIC), [Post-Quantum Cryptography Working Group](#), as practitioner context for certificate and trust-service migration.
- NIST FIPS 203, [ML-KEM](#); NIST FIPS 204, [ML-DSA](#); and NIST FIPS 205, [SLH-DSA](#), as current post-quantum cryptographic standards relevant to long-lived evidence.

Evidence and infrastructure theory:

- Mark D. Wilkinson et al., “[The FAIR Guiding Principles for Scientific Data Management and Stewardship](#)”, *Scientific Data*, 2016.
- Stian Soiland-Reyes et al., “[Packaging Research Artefacts with RO-Crate](#)”, *Data Science*, 2022.
- Association for Computing Machinery, [Artifact Review and Badging, Version 1.1](#), 2020.
- Zenodo, [About records](#) and [DOI versioning](#), accessed 30 May 2026.
- Carlota Perez, *Technological Revolutions and Financial Capital: The Dynamics of Bubbles and Golden Ages*, Edward Elgar, 2002. ISBN 9781840649222.
- Susan Leigh Star and Karen Ruhleder, “[Steps Toward an Ecology of Infrastructure](#)”, Information Systems Research, 1996.
- Geoffrey C. Bowker and Susan Leigh Star, *Sorting Things Out: Classification and Its Consequences*, MIT Press, 1999. ISBN 9780262522953.
- Michael Power, *The Audit Society: Rituals of Verification*, Oxford University Press, 1997. ISBN 9780198296034.
- Lawrence Lessig, *Code and Other Laws of Cyberspace, Version 2.0*, Basic Books, 2006. ISBN 9780465039142.

AEP artifact reference:

- Anton Sokolov, *Agent Evidence Package artifact bundle*, Zenodo concept DOI 10.5281/zenodo.20273730 and v0.2 DOI 10.5281/zenodo.20396464; public artifact record for AEP/verifier materials.