AUTHORIZE THE ACT, NOT THE AGENT

Carried Authority and the Computable Authority Equation

Allan Sendagi
SafeHaven AI / NOMOS Protocol
October 22, 2026

Abstract

Every time institutions scaled to coordination among strangers, the new scale became stable only after infrastructure existed that let strangers rely on the same rules without trusting one another: written law, double-entry bookkeeping, standard measures. Autonomous agents are the next such scale. Part I of this series argued that it needs institutional authority represented as a computational object [R-07]. This paper asks what it takes for that object to govern an agent that carries it.

Computable Authority defines an authorized action by one equation, V = Eval_A(a, F, t): an institution's sealed authority, applied to proven facts about an exact action at the moment of action [R-17]. The equation has no term for the agent. In deployment, however, the agent is the *carrier* of A, a and F to the system that will make the change. A carrier can choose versions, facts, omissions, certificate orders and audiences without forging anything.

We formalize "the agent is not an input" as **carrier indifference** and characterize it. Four conditions are jointly sufficient for it, and the six obligations into which we refine them are individually necessary:

- the authority is genuine and current;
- the facts are proven;
- the action is bound;
- the verdict is computed where the change lands.

Each necessity result names a simplification found in current practice. We executed every construction against the reference verifier with that one obligation removed. All twelve attacks commit under the weakened verifier and are refused by the full one.

The sharpest result concerns silence. Institutional rules are defeasible: their exceptions override their permissions. Withholding information to obtain a better decision is a known attack on attribute-based access control, studied as attribute hiding [Tschantz & Krishnamurthi 2006; Crampton & Morisset 2012]. Here every fact is carried by the adversary, so the attack is always available, and two-valued evaluation lets the carrier choose which exceptions exist. Kleene's third value with a single gate makes every prioritized rule set safe against it by construction, without restricting the rule language, at a cost confined to the exceptions that could actually stop the act. Safety must also preserve which people approve, not only whether the act is permitted. We prove this in Lean 4. A small-scope check over 20,000 random prioritized rule sets finds no counterexample for the corrected evaluator; the two-valued evaluator fails on 46% of the same sets.

Proving the theorems against the reference verifier also exposed a soundness gap in the specification as then published (SPEC-008 0.4.0), which 29 vectors, 729 mutants and two agreeing implementations had all missed. When two escalations share a priority, a carrier could withhold one fact and so choose which people approve the act. The same check shows the 0.4.0 gate failing on 1.3% of rule sets, and the corrected gate on none. The correction, with a regression vector, shipped as SPEC-008 0.5.0 and in the platform verifier.

We also prove compromise containment. It shows that a relying party's choice of currency source decides whether compromise of an issuer's key is contained. We prove that verdicts are reproducible by third parties. We map recent agent-authorization proposals against the conditions, and state the open problems.

The construction is specified as NOMOS-SPEC-007 and NOMOS-SPEC-008, published with two implementations and 30 conformance vectors on which both agree; the implementations also agree on 729 mutants derived from the first 29 [R-16].

1. Introduction

1.1 The coordination principle

Institutions have scaled to larger groups of strangers only after acquiring infrastructure that lets strangers rely on the same rules without trusting one another. Written law codes, Ur-Nammu's around 2100 BCE and Hammurabi's around 1754 BCE, fixed rules that a subject could consult without knowing the ruler [Roth 1997]. Double-entry bookkeeping, complete in Florentine and Genoese ledgers between 1299 and 1340 and codified by Pacioli in 1494, let a merchant's accounts be checked by someone who was not the merchant [Lee 1977; Peragallo 1938; Pacioli 1494]. Standard weights and measures, and in 1875 a shared international basis for length and mass, let a contract mean the same thing in cities that had never met [Kula 1986; Metre Convention 1875]. Economic historians describe the same transition from the side of exchange: trade among strangers grew once institutions made rules enforceable by third parties rather than by personal relationship [North 1990; Greif 2006; North et al. 2009], and trust itself moved from persons to institutions [Zucker 1986; Seabright 2004]. In each case the leap in scale followed the leap in infrastructure. It did not precede it.

Hart gave the principle its jurisprudential form [Hart 1961]. A small community that knows itself can live by primary rules of conduct. As it grows, those rules suffer three defects: uncertainty about which rules are valid, a static character, and no authoritative way to determine whether a rule was broken. A legal system answers each defect with a secondary rule, of recognition, of change and of adjudication. §2 shows that the obligations of this paper are those three rules, stated for a machine that must apply them to an adversary.

Autonomous agents are the next change of scale: institutional decisions at the speed and volume of software, on behalf of parties who have never met. The historical requirement reappears in exact form. When the party that proposes an action also supplies the evidence and chooses which version of the rules to present, institutions, relying parties and regulators can coordinate on the same rule only if **the agent is not an input to the verdict**. This paper makes that condition precise, as carrier indifference, and proves what it takes.

1.2 The confused deputy, deployed by the million

In 1988 Norm Hardy described a compiler that was tricked into overwriting a file it had been granted the right to write [Hardy 1988]. The compiler was not malicious. It held authority for its own purposes and used it on behalf of whoever named the output file. Hardy called it a *confused deputy*, and the observation became the founding argument for capabilities: authority should travel with the designation of what it is for, not sit in the program that happens to hold it.

Large language model agents are that compiler, deployed at scale. They hold credentials to payment systems, ledgers, registries and APIs. They act on text, and anyone can write text: a document, a web page, an email, a tool result. When an injected instruction redirects an agent [Greshake et al. 2023; Debenedetti et al. 2024], nothing about the agent changes:

- its identity is the same;
- its credentials are the same;
- its operator is the same.

An honest agent misled by injected text and a compromised agent are indistinguishable at the moment they act.

1.3 The field's answer: authorize the agent

The response emerging in 2025 and 2026 starts from the agent:

- **Know who it is.** Standards for agents to identify themselves to the services they call [RFC 9421; Visa TAP; KYA 2026], identity frameworks for agents, and credentials that "travel with" an agent, several of them called agent passports [Uchibeke 2026].
- **Know whom it acts for.** OAuth extensions that carry the agent as an *actor* through the user's consent and into the token [draft-oauth-ai-agents-on-behalf-of-user].
- **Check it on the way out.** Pre-action hooks in agent frameworks [Uchibeke 2026], governance proxies [Racioppi 2026] and authorization gateways [Kodathala 2026] that evaluate each tool call against a policy before forwarding it.
- **Sign what it may do.** Mandates signed by the user that state what an agent may buy [AP2].

Each of these is useful, and some are necessary. Traffic management needs to know what is calling. A user's delegation has to be recorded somewhere.

None of them, alone, answers the question a consequential change actually raises: **does an institution's authority permit this exact change, on these facts, now?** Identity cannot answer it, because identity cannot tell the honest agent from the misled one. A check in the agent's own path cannot enforce it, because the agent controls its own path. A signed mandate cannot settle it, because the facts that decide whether the mandate applies are still the agent's to supply. An independent security analysis of the payment mandate protocol reached the same conclusion from the other side: the inputs that shape a transaction before signing sit outside the mandate's protection [Aviv et al. 2026].

Clark and Wilson saw the institutional form of this problem in 1987 [Clark & Wilson 1987]. Commercial integrity, they argued, rests on *well-formed transactions* and *separation of duty*. Data changes only through certified procedures, and no single person controls every step. Their model certified the *procedure*: a transformation procedure is checked once, then trusted to run. An agent cannot be certified that way. Its behavior is not a fixed program but a response to whatever it reads.

1.4 Certify the act, not the program

If the procedure cannot be certified, each transaction must be. That is the move this paper formalizes. Computable Authority defines authorization by an equation whose inputs are the institution's sealed authority, the exact action, the proven facts and the moment, with no term for whoever asks [R-17]:

```
V = Eval_A(a, F, t)
```

Removing the agent from the equation does not remove it from the system. Something must carry A, a and F to the place where the change would happen, and in an agent deployment that carrier is the agent. It chooses:

- the action;
- which version of the rules to present;
- which facts to bring and which to leave out;
- which certificates to include, and in what order;
- which system to address.

None of these choices requires forging a signature. A verdict computed from carried inputs equals the equation's verdict only under conditions, and stating those conditions exactly is the purpose of this paper.

The resulting architecture has the agent hold no authority at all. Authority travels inside the act, sealed by those who hold it, and the system of record checks it before anything changes. An agent that holds no authority cannot be a confused deputy. At worst it is a confused courier, and a courier's confusion has no effect on what the recipient verifies.

1.5 Contributions

1. **A model of carried authority** (§3). It treats the agent as an adversarial carrier of the equation's inputs and defines *carrier indifference*: whatever the carrier does, a commit implies that legitimate, current authority applied to the true facts permits the action, or that the required people approved exactly it.
2. **Necessity** (§4). Six obligations refine the equation's four conditions, and each is necessary. Each theorem names the simplification it forbids and constructs an attack from genuine statements alone. Every construction was executed against the reference verifier with that one obligation removed, and all twelve succeed (§7.1). Since each ablation keeps the other obligations intact, the obligations are independent.
3. **Defeasibility and silence** (§4.5, §5.2). Attribute hiding is known in attribute-based access control, where it is countered by checking each policy for resistance or by restricting the policy language to monotone forms (§2). Carried facts make the attack universal: every fact passes through the adversary. We give an evaluator-level remedy instead. Kleene's third value with one gate makes *every* prioritized defeasible rule set safe, with no restriction on the language law is written in, at a cost confined to the exceptions that outrank the would-be outcome. We also show that safety over grant and deny is not enough when a permission can route to people: the evaluator must also preserve *which* escalation applies, which is the case SPEC-008 0.4.0 missed.
4. **Sufficiency and its corollaries** (§5):
   - the four conditions imply carrier indifference;
   - honest carriers reach truthful verdicts by disclosure, and, with the corrected gate, silence is never rewarded;
   - `exists` over witnessed facts is a liveness hazard authors must avoid;
   - compromise of any one key is contained, except that issuer compromise is contained only when the relying party establishes currency from its own record;
   - verdicts are reproducible by third parties.

Proposition 1, the omission-safety property, is mechanized in Lean 4, with the 0.4.0 counterexample.

5. **Evidence** (§7):
   - twelve executed necessity witnesses;
   - a small-scope model check of omission safety over 20,000 random rule sets, with the two-valued evaluator as a control;
   - the published conformance cases, re-run under 0.4.0 (31/31) and 0.5.0 (32/32);
   - the seven defects of R-16, each classified by the obligation it violated;
   - a soundness gap in SPEC-008 0.4.0 found by proving Proposition 1, which agreement between the two implementations could not reveal; its fix and regression vector shipped as SPEC-008 0.5.0;
   - two further refinements to the specifications.
6. **A landscape** (§6) of current agent-authorization proposals against the conditions, with each judgment tied to the published specification it rests on.

1.6 Relation to the rest of the series

- **Part I** [R-07] established authority as an object: what it is, how it is compiled, why compiling differs from executing. Part II establishes how that object travels with an action and is recognized and enforced by a system that has never met its issuer.
- **R-16** [Passports for Agents] is the engineering report on NOMOS-SPEC-007 and NOMOS-SPEC-008: threat model, the original omission proof, conformance vectors, mutants, defects and cost. This paper relies on it for the implementation evidence and does not repeat it.
- **R-17** defines the equation.
- **R-18** names the human roles the equation leaves to people.

We do not use the word *passport* for the object an agent carries. The name is in use for identity credentials bound to agents [Uchibeke 2026], which is the model this paper argues against. A passport also identifies its holder, and what an Act carries is not the holder's identity but an institution's authority

over one change. What we keep from electronic passports is their verification architecture [ICAO 9303]:

- the document is checked locally, without calling its issuer;
- against trust the verifier chose itself;
- by a border that, not the issuer, decides admission.

The relying party is that border.

2. Lineage

Carried authority is not a new idea about cryptography. It applies old ideas about authority to a carrier that cannot be trusted. Eight traditions meet in it. In each case we state what we take from it, and where we depart.

Secondary rules. Hart distinguished the primary rules that govern conduct from the secondary rules that make a body of rules a legal system, each answering a defect that appears when a community outgrows personal knowledge of its members [Hart 1961]. The obligations of §3.7 are those secondary rules, made checkable by a verifier from what an Act carries:

Defect of primary rules alone	Secondary rule	Obligation
Uncertainty: which rules are valid, and who may establish a fact	Rule of recognition	C1.1 recognized authority; C2.1 provenance
Static character: rules cannot be deliberately changed	Rule of change	C1.2 the version in force, supersession and revocation
Inefficiency: no authoritative determination of what happened	Rule of adjudication	C2.2, C3 and C4: a verdict computed at the point of effect, INCOMPLETE while undecided, approvals bound to the exact act

Hart's secondary rules are addressed to officials who know which rules they apply. Here the official is a relying party that has never met the issuer, and the requester is an adversarial carrier, so each secondary rule must be verifiable from signed statements alone.

Complete mediation. Saltzer and Schroeder required that every access to every object be checked for authority, that decisions default to denial, and that privilege be separated so no single key unlocks everything [Saltzer & Schroeder 1975]. C4, C2.2 and the purpose separation of §5.5 are those principles. What this paper adds is *where* the check must sit when the requester is the adversary and may hold credentials of its own. It must sit at the holder of the only credential that can make the change (Theorem 1).

Well-formed transactions. Clark and Wilson grounded commercial integrity in two ideas [Clark & Wilson 1987]:

- data changes only through certified transformation procedures;
- duties are separated so that no one person controls a whole transaction.

The relying party's state plays the role of their constrained data. Dual consent from distinct keys is separation of duty. The receipt is their log. The departure is the one §1.3 describes: Clark–Wilson certifies procedures in advance, which presumes a procedure that can be certified. An agent is not such a procedure, so we certify each transaction at the moment of commit instead.

Capabilities and the confused deputy. Dennis and Van Horn introduced capabilities as unforgeable tokens that combine designation with authority [Dennis & Van Horn 1966]. Hardy showed why ambient authority fails: a deputy using its own authority on another's behalf can be steered [Hardy 1988]. Modern capability tokens make authority attenuable and verifiable offline [Birgisson et al. 2014; UCAN 1.0; Biscuit].

We take from this tradition the refusal to let authority reside in the actor. We depart in what the carried object says:

- a capability says what its bearer may do;
- an Act says why one specific change is permitted: an institution's rules, evaluated on facts attested by others, sometimes requiring named people to consent.

The two compose. A capability can authorize a carrier to submit Acts at all. §4.5 shows why the departure is forced. Capability conditions are positive conjunctions, so silence cannot help the bearer. Institutional rules have exceptions, and silence can.

Speaks for. Lampson, Abadi, Burrows and Wobber reduced authorization in distributed systems to a calculus of principals that *speak for* one another [Lampson et al. 1992]. In that vocabulary, current agent proposals let the agent speak for its user. Carried authority makes the agent speak for no one: each statement in an Act speaks for its signer, within the scope its certificate grants. The carrier is not a principal in the authorization logic. It is a channel for statements, with no standing of its own.

Trust management and proof-carrying authorization. PolicyMaker, KeyNote and SPKI separated the credentials a request presents from the local policy that judges them, and made delegation monotone [Blaze et al. 1996; RFC 2704; RFC 2693]. SPEC-007's scoped, narrowing certificates are in this lineage (Proposition 2). Proof-carrying code and proof-carrying authentication had the requester carry a proof the verifier can check cheaply [Necula 1997; Appel & Felten 1999; Bauer et al. 2005].

An Act is proof-carrying in a deliberately restricted sense:

- it carries the evidence, not a proof in a general logic;
- the verifier performs one fixed, decidable evaluation.

That keeps verification cheap, near a millisecond [R-16 §9.6], and keeps verdicts auditable by parties with no theorem prover.

Defeasible rules. Hart observed that legal concepts are *defeasible*. A rule's application can be defeated by exceptions not listed in the rule [Hart 1949]. Logics of prioritized

defeasible rules formalize how exceptions override general provisions [Prakken & Sartor 1997], and legal rule interchange formats represent defeasibility directly [LegalRuleML 2021]. NOMOS authorities are prioritized defeasible rule sets. §4.5 shows that defeasibility, which is natural for law, is exactly what makes carried facts dangerous. §5.2 shows that Kleene's strong three-valued logic [Kleene 1952], with one gate, contains the danger. XACML's Indeterminate result is a precedent for refusing to permit on incomplete information [XACML 3.0]. We use a third value for the same reason, but in a setting where the evidence is carried rather than fetched.

Missing attributes and attribute hiding. The danger of withheld information is established in attribute-based access control.

- Tschantz and Krishnamurthi defined a policy as *safe* when an incomplete request is granted only if the complete one would have been [Tschantz & Krishnamurthi 2006].
- Crampton and Morisset named the *attribute-hiding attack*: a requester withholds an attribute to obtain access it would otherwise be denied. They proposed weakly monotonic targets as a language-level defence [Crampton & Morisset 2012].
- Griesmayer and Morisset defined *policy resistance* to such attacks and certified it per policy, finding counterexamples with Maude and proofs with Isabelle [Griesmayer & Morisset 2013].
- Crampton and Morisset showed that XACML is neither monotonic nor complete, and defined XACML-style models that are each [Crampton & Morisset 2014].
- Crampton, Morisset and Zannone studied evaluation when attributes may be missing or must be retrieved [Crampton et al. 2015].
- Related work makes evaluation resilient to evaluation failures [Crampton & Huth 2010] and composes policies in Belnap's four-valued logic [Bruns & Huth 2011].

Proposition 1 is this safety property, for the setting of this paper. The departures are three:

- **Setting.** In attribute-based access control, a decision point receives or retrieves attributes, and hiding needs a requester who controls an attribute source. Here every fact passes through the adversary, as signed testimony it may include or leave out, so the attack is always available. The authority's data contract decides which absences are authoritative (only those of the action and of the relying party's own state).
- **Remedy.** The established remedies certify safety one policy at a time, or restrict the policy language to monotone forms. The gate of §3.3 instead makes every prioritized defeasible rule set safe at the evaluator, by returning INCOMPLETE. The language keeps its exceptions, and the cost is bounded (Corollary 3).
- **What is preserved.** Safety over grant and deny is not enough when a permission can route to people. The evaluator must also preserve *which* escalation applies, or the carrier chooses its approvers. This is the case SPEC-008 0.4.0 missed (§7.5). We know of no earlier statement of safety over escalation identity.

Machine-readable travel documents. An electronic passport is verified at a border, locally, against country signing certificates the border chose to trust. The issuing state vouches for the document, and the border decides admission [ICAO 9303]. We adopt this verification architecture, not the object. The relying party is the border, and the certificates it pins are its own choice (§4.2).

3. The carried-authority model

This section fixes the objects, the adversary and the property we want. Notation is kept to what the theorems need. Truth values are written Kleene's way, as numbers: 1 is true, 0 is false and ½ is undecided.

3.1 The equation, and the problem it hides

Computable Authority defines an authorized action by one equation [R-17]:

```
V = Eval_A(a, F, t)

A   the authority   an institution's sealed, versioned rules
a   the action      the exact change being proposed
F   the facts       the evidence the rules read
t   the time        the moment of action
V   the verdict     AUTHORIZED | DENIED | ESCALATED | INCOMPLETE
```

Computing a decision from a policy and several inputs is not new. Attribute-based access control [NIST SP 800-162] and trust management [Blaze et al. 1996] do it. What sets this equation apart is a term it leaves out. It has no subject: the party asking is not an input.

On paper that omission is free. In deployment it is not, because something has to bring A, a and F to the system that will make the change. Here that something is the agent. It chooses the action. It assembles the facts it can find. It picks which version of the rules to present. It decides which statements to include and which to leave out. It chooses the certificates and the order they come in. So the agent never appears in the equation, yet it carries every input.

The question for this paper is therefore exact. **Under what conditions does a verdict computed from carried inputs equal the verdict the equation defines, when the carrier is adversarial?** When those conditions hold, the agent is not an

input in fact as well as on paper. When they fail, the agent is an input through whichever of A, a or F it was able to choose.

3.2 Parties

- **Issuer.** An institution that authors and seals authorities. It holds a *sealing key*.
- **Witness.** A party certified to state particular facts. It holds a *claim key* whose certificate names those facts.
- **Approver.** A person certified to consent in a particular role. They hold a *consent key*.
- **Root.** A key that certifies other keys within a scope, until an expiry.
- **Relying party ρ.** The system of record for the state an action would change: a ledger, a registry, an API in front of its own database. It pins its own root key, keeps a nonce ledger, holds its own state σ, and may keep its own record of which version of each authority is in force.
- **Carrier.** Whoever assembles an Act and submits it, usually an AI agent. It has no key that appears anywhere in verification.

These roles follow the three human positions developed in [R-18]:

- the Issuer authors A;
- the Escalation Target answers an escalation with a signed consent;
- the Sovereign is accountable for what the relying party commits.

3.3 Authorities and their semantics

An authority `A` is a sealed artifact. It contains:

- a finite set of **decisions**. Each decision `d` has a condition `cond_d`, an outcome `out_d` ∈ {allow, block, escalate(e)} and a priority `prio_d` ∈ ℕ;
- a **tie order** `≻_A` on outcome types, strict (it ranks allow, block and escalate differently), used when decisions of equal priority conflict, and a **default outcome** `δ_A`, used when no decision applies;
- **escalations**. Each escalation `e` names a role, `role_e`, and the number of distinct approvers it needs, `n_e`;
- a **data contract**. It gives each input `x` a declared *channel* `κ(x)` and a declared *type* `τ(x)`. The four channels are:
  - `act`: the parameters of the action itself;
  - `wit`: signed testimony;
  - `rp`: the relying party's own state;
  - `pre`: the carrier's own word.

Every authority also has an identifier, a version, a seal hash `h(A)` and a sealing key.

Facts. A fact assignment `F` is a partial map from inputs to values. `F ⊑ F′` means `F′` extends `F`: it is defined wherever `F` is, and agrees with it there.

Conditions. A condition is built from atoms with `and`, `or` and `not`.

- An atom reads a set of inputs. If it reads an absent input, its value is ½ (undecided). The one exception is `exists(x)` where `κ(x) ∈ {act, rp}`, which is decided by absence.
- An atom whose inputs are present but outside the atom's domain also gets ½. An ordering comparison on a string is an example.
- Otherwise an atom has its classical value, 0 or 1.
- The connectives follow Kleene's strong logic [Kleene 1952]: `and` is min, `or` is max, `not` is 1 − x.

The resulting value of a condition is written `⟦c⟧_F` ∈ {0, ½, 1}.

Resolution.

- Let `Win(F)` be the decisions whose condition is 1 under F.
- If `Win(F)` is empty, the outcome is `δ_A` and there is no deciding decision.
- Otherwise each decision has a **rank**, compared lexicographically:

`` `rank(d) = ( prio_d, tie rank of out_d's type under ≻_A, −position of d in the sealed list )` ``

The deciding decision `D` is the member of `Win(F)` with the highest rank, and the outcome is `out_D`. Because rank is a total order, the decider is unique.

- A decision `X` **outranks** `D` when there is no deciding decision, when `prio_X > prio_D`, or when the priorities are equal and `X`'s outcome type ranks above `D`'s under `≻_A`. Outranking ignores position. It is the relation SPEC-008 §6.2 states, and it implies higher rank.

Ranking by position states what both SPEC-008 implementations always did, and what 0.5.0 makes explicit: when two decisions of the same priority and the same outcome type both hold, the one listed first decides.

The gate. Suppose the outcome is allow or escalate, and some block or escalate decision `X` has `⟦cond_X⟧_F = ½` and satisfies one of:

- (i) `X` outranks `D`;
- (ii) `X` and `D` have equal priority, both escalate, and they name different escalations.

Then `Eval_A` returns INCOMPLETE.

Clause (ii) was not in SPEC-008 0.4.0; 0.5.0 adds it. We found its absence while proving Proposition 1, and §7.5 shows it was exploitable. Without (ii), a carrier can withhold the fact that triggers a stricter escalation and obtain a weaker one at the same priority. The gate above is the corrected one, and the theorems use it.

If the gate does not fire:

- allow gives AUTHORIZED;
- block gives DENIED;
- escalate(e) gives ESCALATED, which becomes AUTHORIZED once `n_e` distinct approvers certified for `role_e` have consented to this exact act (§3.4, C3).

3.4 Acts and the relying party's verdict

An **Act** α carries:

- an action a, made of a type, a subject and parameters;
- an authority A;
- a set of certificates;
- testimony: signed statements of the form (claim, value, subject, window, key);
- consents: signed statements of the form (digest, role, decision, key);
- an audience, which names the relying party;
- a nonce;
- a validity window;
- optionally, an in-force statement and carrier facts.

An Act has no signature from the carrier.

The relying party assembles the facts F_α from four sources:

- the action's parameters;
- the values of valid testimony;
- its own state σ_ρ(t);
- any carrier facts.

It then computes a verdict V_ρ(α, t) and commits a only if that verdict is AUTHORIZED.

3.5 The adversary

The carrier is the adversary. It may build any Act from:

1. any action it likes;
2. every authority any issuer has ever sealed, every version of each, superseded ones included;
3. every certificate, testimony statement, consent and in-force statement anyone has ever signed. The adversary sees everything ever signed and may replay it;
4. any statement signed by a key it controls. Its own key, or a root it generated, is certified by nobody the relying party pins.

The adversary may also delay messages, reorder them and submit Acts concurrently.

It cannot:

- forge an Ed25519 signature under a key it does not control (assumption A1, existential unforgeability [RFC 8032]);
- find SHA-256 collisions (A1).

A2 says that no key outside the adversary's control is compromised. §5.5 relaxes this. A3 says the relying party's clock is within a small bound of real time. A4 says witnesses state true values for the claims they are certified for, and that a statement remains true throughout the validity window it states. A witness that cannot vouch for a fact for that long must state a shorter window.

A4 is a governance assumption, not a cryptographic one: certification limits *which* facts a key may state, not whether it tells the truth [R-16 §10.2]. The theorems are relative to it, and §5.5 shows what breaks when it fails.

3.6 What "the agent is not an input" means, formally

Let Cur_ρ(t) be the set of authorities *legitimately in force* for ρ at time t. An authority belongs to it when all of the following hold:

- its sealing key chains, within scope, to the root ρ pinned;
- nothing on that chain is revoked at t;
- the authority itself is not revoked;
- it is the version its issuer designates as in force at t.

Let F★_t be the true values at t of every wit and rp input.

> **DEFINITION 1** **Carrier indifference.** A verifier is *carrier-indifferent* if, for every Act α the adversary can construct and every t, V_ρ(α, t) = AUTHORIZED implies all of the following:

- **(S1) Legitimate authority.** α.authority ∈ Cur_ρ(t).
- **(S2) Truthful outcome.** Evaluating the authority on the true facts does not refuse the action. That is, Eval_A(a, F★_t ⊕ a, t) is AUTHORIZED, or it is ESCALATED by an escalation the Act satisfies. Here F★_t ⊕ a is the true facts extended with the action's own parameters.
- **(S3) Approval of exactly this act.** If the commit rests on escalation e, then n_e distinct keys certified for role_e signed a digest that determines all of: a, h(A), the testimony carried, the audience ρ, the nonce and the window.
- **(S4) Once, here.** α is committed at most once, and only by the relying party it names.
- **(S5) Exactly this change.** What is applied to σ is a and nothing else.

Inputs declared pre are taken as presented. Declaring them is the institution's explicit acceptance of the carrier's word, and anyone can see that acceptance by reading the authority's data contract. S2 is relative to that declaration.

Carrier indifference is the formal content of the claim that the agent is not an input. The carrier can choose which action to propose, and nothing else that matters. Whatever certificates it includes, in whatever order, and whatever version it presents, whichever statements it adds or leaves out, and whatever it says itself: if the action commits, then legitimate, current authority applied to the truth permits it, or the right people approved exactly it.

3.7 Four conditions

R-17 states four conditions on the inputs. For the theorems we refine them into six obligations. The refinement is the paper's working vocabulary.

```
C1  A is genuine and current
    C1.1  Recognized   the sealing key chains, within scope, to a root the relying party
                       pinned; resolution does not depend on presentation order; nothing on
                       the chain is revoked at t
    C1.2  Current      A is the version in force at t, established from the relying party's
                       own record or from an issuer-signed in-force statement, as a condition
                       separate from authenticity

C2  F is proven
    C2.1  Provenance   every fact arrives through the channel A declares for its input, with
                       its declared type; witnessed facts only from keys certified for that
                       claim, about the act's subject, inside their window
    C2.2  Omission     an absent or out-of-domain fact is undecided; an allow or escalate
          safety      outcome commits only if the gate of §3.3 is closed: no outranking block
                       or escalate condition, and no equal-priority escalation naming a
                       different escalation, is undecided

C3  a is bound    every consent signs a digest of the action, h(A), the testimony, the
                  audience, the nonce and the window; the commit applies exactly a

C4  V is computed where the change lands
                  the component holding the only credential able to apply a to σ computes
                  V, refuses any change that does not arrive as a verified Act, records the
                  nonce atomically with the change, and re-reads its own facts at commit
```

Sections 4 and 5 show two things:

- each obligation is necessary: drop any one and an adversary using only genuine statements breaks carrier indifference;
- together the obligations are sufficient.

So the conditions characterise carrier indifference rather than merely achieving it.

4. Necessity: each obligation rules out a simplification found in practice

Each obligation in §3.7 forbids something a reasonable engineer would otherwise do. Here is what each one forbids:

Obligation	The simplification it forbids
C1.1	Verify the issuer against whatever key the request names.
C1.2	A genuine, unrevoked authority is a usable one.
C2.1	Accept the facts the caller sends.
C2.2	A fact that is not there is false.
C3	An approval of the request is an approval of the change.
C4	Check the agent's request at a gateway, then let it through.

Each theorem in this section shows that the simplification breaks carrier indifference. In every case the adversary uses only genuine statements: it forges no signature and needs no key compromise.

Every construction is also executed. For each obligation, we removed that obligation alone from the SPEC-008 reference verifier and left all the others in place. We then submitted the attack Act to both the weakened verifier and the unmodified one, and evaluated the institution's rules on the true facts. All twelve constructions were witnessed (§7.1, Table 3):

- the weakened verifier committed;
- the unmodified verifier refused;
- the rules on the true facts did not permit the commit.

Because each ablation removes one obligation while keeping the rest, the twelve witnesses also establish *independence*: no obligation is implied by the others.

4.1 Where the verdict is computed (C4)

> **THEOREM 1 Point of effect.** Let Q be any component, other than the verifier, that holds a credential able to apply some action a to σ, and suppose the carrier can reach Q with that credential's authority. Then the carrier can apply a whatever V_ρ returns.

Proof. Q applies a without consulting V_ρ. ∎

The theorem is elementary. Saltzer and Schroeder's complete mediation already demands that every access be checked [Saltzer & Schroeder 1975]. What makes it worth stating is where agent architectures place the check.

A pre-action hook in the agent's own framework [Uchibeke 2026] sits in front of a tool. So does a governance proxy [Racioppi 2026] or an authorization gateway [Kodathala 2026]. Each of these satisfies C4 only if it holds the *only* credential able to make the change. Mandato states this requirement explicitly: upstream servers accept sessions only from the proxy's workload identity [Racioppi 2026]. When a component holds the only credential, it is the relying party in our sense, and it inherits every duty in C4, including atomic nonce recording and re-reading its own state at commit.

An agent that holds an API key to the downstream system, or that can run code in the process hosting the hook, has a path that does not pass through the check. In our adversary model

the carrier controls its own process, so a check located inside that process is advice, not enforcement.

C4 relocates the check to the one place the carrier cannot route around: the system whose state the action would change. SPEC-008 states the consequence plainly. The carrier may bypass every NOMOS component, and its act still cannot take effect, because the only system able to make the change demands the proof [SPEC-008 abstract].

4.2 Whose root (C1.1)

THEOREM 2 Pinned recognition.

(a) A verifier that resolves the sealing key against a trust anchor named or supplied in the Act is not carrier-indifferent.

(b) A verifier whose resolution depends on the order of the presented certificates lets the carrier choose among verdicts.

Proof of (a). The adversary generates a root, certifies its own issuer key under it, and seals an authority whose only decision is `amount > 0 → allow`. Because the relying party keeps no record for this new authority identifier, the adversary adds an in-force statement signed by its own issuer key. Every signature is genuine under keys the adversary controls, and the verifier accepts the adversary's root because the Act supplies it. The Act commits, but the authority is not in $\mathrm{Cur}_\rho(t)$, so S1 fails. Witnessed: T1. ∎

Clause (a) is the authorization analogue of a well-documented token flaw: verifiers that fetch the signing key from a URL or key embedded in the token's own header. Current best practice for JSON Web Tokens forbids this [RFC 8725]. The relying party must pin its root itself. SPEC-007 never supplies a default [SPEC-007 §4.1].

Proof of (b). Take one certificate set in which the witness key is reachable by two paths: one through a valid certificate and one through an expired one. Resolve it first-match in presentation order. Presenting the expired path first yields TESTIMONY_INVALID, while presenting the valid path first yields AUTHORIZED. Witnessed: P6. ∎

Under assumption A2, every path that resolves is a legitimately certified path. So in our adversary model the carrier's choice is between a correct verdict and a wrongful refusal, not an illegitimate commit. Order independence is therefore necessary for something other than S1–S5: that two parties checking the same Act reach the same verdict, which is the property audit needs (Theorem 10). It is also necessary for the principle that the carrier chooses only the proposal.

SPEC-007 rev. 1.7.0 had this defect. A second implementation found it, and rev. 1.8.0 fixed it with target-first resolution in fingerprint order [R-16 §5.3].

4.3 Which version (C1.2)

Issuers replace authorities: limits change and rules are added. A replaced version is not wrong, so issuers do not revoke it. Revoking it would also destroy something institutions need: the ability to show, later, that a decision taken under version 1.2 while 1.2 was in force was valid when it was taken. Call this requirement **H**, historical verifiability.

THEOREM 3 Currency is not authenticity. Let a verifier satisfy H, and check only that an authority is authentic and unrevoked. Then whenever some superseded version authorizes an action the current version does not, the carrier obtains that action. S1 fails, and in general S2 fails too.

Proof. Under H, a superseded, unrevoked version must still verify as authentic and unrevoked; otherwise decisions committed under it could no longer be shown valid. A verifier that checks only those two properties therefore accepts the superseded version for a new action. In the witnessed case, version 0.9.0 lacks the review threshold that the current version applies to loans above $10,000. A $20,000 loan presented under 0.9.0 is AUTHORIZED, while under the version in force it is ESCALATED to a reviewer. Witnessed: T5. ∎

The theorem says that two questions must be answered separately:

- *Was this version valid for a decision made at t_0?*
- *May this version authorize a new action at t?*

A single revoked/unrevoked bit cannot answer both. Revocation answers only "is this authority withdrawn for every purpose?" [SPEC-006]. SPEC-008 §5.4 answers "is it the one in force now?" from the relying party's own record, or from an in-force statement the issuer signs, valid for at most a day.

An outside reader's challenge to a published scenario exposed this gap [R-16 §9.4]. The test vectors of the version before it did not cover the case.

4.4 Who may say what (C2.1)

THEOREM 4 Rule-declared provenance. Suppose an authority A depends on an input x, meaning some two fact assignments that differ only at x produce different outcomes. If a verifier accepts a value for x from the carrier, the carrier decides the outcome on x. In particular, whenever the true value of x produces a refusal and another value produces a commit, S2 fails.

Proof. The carrier supplies the value of x that produces the commit. In the witnessed case the bureau holds a credit score of 600. Under the authority no rule then matches, and the default outcome refuses the loan. The carrier asserts 768 itself, and the permission rule R0 applies. Witnessed: T2. ∎

The theorem is immediate. The design consequence is not. **The channel of each fact must be part of the authority**, for two reasons:

- If the relying party decided acceptable sources by local configuration, two relying parties evaluating the same sealed authority could reach different verdicts.

- The institution would have delegated its most consequential choice — whose word counts for each fact — to whoever configures the verifier.

Deciding who may establish a fact is itself an exercise of authority, so it belongs inside A. That is why SPEC-008 reads each input's channel from the sealed data contract [SPEC-008 §7], and why a `pre` input is visible to any reviewer of the authority as a deliberate acceptance of the carrier's word.

4.5 What silence means (C2.2)

This is the obligation the authors violated most often: three of the seven defects in R-16 are instances of it. The attack is the attribute-hiding attack of attribute-based access control (§2) [Crampton & Morisset 2012]. What carried facts change is its reach: every fact passes through the adversary, so for any authority with exceptions on witnessed facts the attack is available by construction.

DEFINITION 2 Defeasible authority. An authority is *defeasible in x*, with $\kappa(x) = \text{wit}$, if there is a world $F\star$ such that:

- on $F\star$ the decider is a block or escalate decision `X` whose condition reads x;
- on $F\star$ with x withheld, and conditions evaluated two-valued (an atom reading x is false), `cond_X` is false and the decider is a decision `D` whose outcome would commit.

Informally: some permission has an exception that outranks it, and the exception depends on a witnessed fact. The simplest instance is an exception that is a single comparison on x, such as "an active bankruptcy blocks the loan", outranking a permission that does not read x.

The definition asks that withholding x actually *falsifies* the exception. Not every condition that reads x has this property: under two-valued reading, `not (x < 3)` becomes true when x is withheld.

THEOREM 5 Silence is not falsity. Let a verifier evaluate conditions two-valued over carried facts, reading an absent fact, or a present fact outside an atom's domain, as false. Then it is not carrier-indifferent for any authority that is defeasible in some x. Equivalently: no such authority is safe in the sense of Tschantz and Krishnamurthi [2006] under a two-valued evaluator over carried facts.

Proof. Take the world $F\star$ of Definition 2. The carrier includes every true statement except the testimony for x. By the definition, the two-valued verifier's decider is then D, and the Act commits, while on the true facts X decides and refuses or escalates. ∎

The same construction succeeds without omitting anything, by sending x with a value of the wrong kind. In the witnessed case, the string "480" inside an object-typed parameter makes `terms.months > 360` false. It also succeeds through `exists(x)`: if absence decides `exists(x)` for a witnessed input, the carrier withholds the statement and the exception that reads `exists` falls silent.

Consent does not rescue a two-valued verifier. Human approvers who consent to an escalation computed on the incomplete facts approve an act that, on the true facts, the rules deny outright.

Witnessed: T4, four variants (absence; absence with two genuine consents; `exists`; wrong kind).

Why capability systems never needed a third value. In macaroons [Birgisson et al. 2014], UCAN [UCAN 1.0] and Biscuit [Biscuit], authority is a conjunction of restrictions, each of which must be positively discharged. A missing discharge leaves a restriction unsatisfied, so silence can only remove authority. Those systems are omission-safe because their conditions are positive and conjunctive, a property Hardy's argument for capabilities did not need to name [Hardy 1988].

Law and institutional policy are not written that way. They state general permissions and then exceptions that override them, and the exceptions are conditions on facts. Hart named this structure *defeasibility* [Hart 1949], and the logic of prioritized defeasible rules has been studied for decades in AI and law [Prakken & Sartor 1997].

Theorem 5 explains why carrying such rules is dangerous: under two-valued evaluation, the carrier decides which exceptions exist by deciding which facts to bring. XACML meets the same problem through its Indeterminate result under deny-overrides [XACML 3.0], but its decision point fetches attributes itself. That architecture keeps the carrier out of the evidence path at the price of an online dependency on every attribute source, and XACML is still not monotonic in general [Crampton & Morisset 2014]. The attribute-hiding literature answers by certifying each policy's resistance or restricting targets to monotone forms [Griesmayer & Morisset 2013; Crampton & Morisset 2012]. SPEC-008 keeps offline verification and unrestricted defeasible rules. §5.2 shows that Kleene's third value, with one gate, is enough to make the combination sound.

4.6 What an approval approves (C3)

THEOREM 6 Binding. Let consents be signatures over a digest that omits any one of these components:

- the action;
- the authority's seal hash;
- the testimony;
- the audience;
- the nonce.

Then there is an Act, built from genuine statements alone, that commits a change no approver approved.

Proof, one construction per component:

- **Action.** Two senior approvers consent to a $75,000 loan. The carrier changes the amount to $750,000 and resubmits with the same consents.
- **Testimony.** Two genuine bureau statements of the applicant's debt-to-income ratio are both inside their windows: 31% and, issued later, 48%. The approvers see the Act carrying 31% and consent. The carrier substitutes the 48% statement. The rules still escalate rather than deny (the limit is 50%), and the consents satisfy the escalation.
- **Audience.** Consents obtained for an Act addressed to the core ledger are presented, re-addressed, to the card ledger.
- **Nonce.** After the Act commits, the carrier resubmits it with a fresh nonce and the same consents. Approved once, it commits twice.
- **Seal hash.** Consents given to escalation R6 under the institution's base policy are presented under a second in-force policy of the same issuer. That policy escalates the same action to the same role for a different reason, a cosigner waiver.

In each case the weakened verifier commits, the full verifier returns CONSENT_INVALID, and the approvers approved something else. Witnessed: T3, five variants. ∎

The validity window belongs in the digest for a related reason. A relying party keeps nonces only while the Acts they belong to could still be valid [SPEC-008 §5.2]. An approval whose window the carrier can extend therefore outlives the nonce ledger that makes it single-use.

Abadi and Needham's principle that a message should say explicitly what it means [Abadi & Needham 1996] is the general form of Theorem 6. The theorem gives the particular list of what an approval of an agent's act has to say.

5. Sufficiency: the four conditions make the agent irrelevant to the verdict

NOMOS-SPEC-007 and NOMOS-SPEC-008 implement C1–C4. SPEC-008 0.4.0's gate lacked clause (ii) of §3.3 (§7.5); SPEC-008 0.5.0 adds it. The theorems below are stated for the corrected gate. This section summarises the construction only as far as the proofs need. The specifications are normative and contain the details [SPEC-007; SPEC-008]. We then prove that any verifier meeting the four conditions is carrier-indifferent. We also prove what else it guarantees: honest carriers reach truthful verdicts, compromise of any one key is contained, and verdicts are reproducible by third parties.

5.1 The construction in brief

Recognition (SPEC-007). A key certificate is a signed statement:

```
(parent_kid, child_kid, child_public_key, scope, issued_at, expires_at)
```

It says the child key may sign within the scope until the expiry. Scope has two kinds of dimensions:

- *artifact dimensions* (artifact identifier, industry, jurisdiction) constrain which authorities a key may seal;
- *statement dimensions* (`claim`: a set of fact names; `consent`: a set of roles) constrain which statements a key may make.

A key whose scope carries a statement dimension cannot seal rules. A sealing key carries no claim or consent scope, so it cannot testify or consent (rev. 1.8.0 §3.5).

To resolve a key, the verifier:

1. enumerates candidate paths from that key toward the pinned root, without revisiting a key and up to 20 certificates;
2. tries the candidates in ascending order of certificate fingerprint;
3. validates each candidate root-first: signature, validity window, revocation, and that scope only ever narrows;
4. takes the first path that validates.

Revocation names the revoked key's identifier, not a certificate.

Binding (SPEC-008). Consents sign:

```
binding_digest = SHA-256( JCS( { act_version, act_id, nonce, issued_at, expires_at,
                                 relying_party, action, artifact_hash: h(A),
                                 testimony_digest: SHA-256(JCS(sorted testimony)) } ) )
```

Here JCS is RFC 8785 canonical JSON [RFC 8785].

Verification (SPEC-008 §6). The relying party runs nine steps in order and stops at the first failure:

1. envelope;
2. audience;
3. window;
4. nonce;
5. issuer chain, seal and scope, with 5a: revocation and currency;
6. testimony (chain, claim scope, signature, window, subject);
7. channels and types;
8. three-valued evaluation with the gate;
9. consent counting.

On AUTHORIZED it applies exactly a. In the same atomic operation it records the nonce and re-reads its own facts, aborting if they changed [SPEC-008 §8].

5.2 Omission safety

We write $F \sqsubseteq F'$ for extension, as in §3.3. The extensions that matter are completions on witnessed and carrier inputs. Action inputs are fixed by a, and relying-party inputs are fixed by σ, which the commit re-reads.

LEMMA 1 Atoms are monotone. Let F′ extend F on witnessed and carrier inputs only. For every atom p, if $[\![p]\!]_F \neq \frac{1}{2}$ then $[\![p]\!]_{F'} = [\![p]\!]_F$.

Proof. An atom that is decided under F reads one of three things:

- inputs that are all present and inside its domain, on which F′ agrees with F;
- exists(x) with x present, which is 1 under both;
- exists(x) with κ(x) ∈ {act, rp}, which F′ does not change.

Every other atom is ½ under F. ∎

LEMMA 2 Kleene connectives are monotone. min, max and 1 − x map refinements of their arguments (½ becoming 0 or 1) to refinements of their result. Hence for every condition c: if $[\![c]\!]_F \in \{0, 1\}$ then $[\![c]\!]_{F'} = [\![c]\!]_F$ [Kleene 1952; Fitting 1994]. ∎

LEMMA 3 Order facts. Rank is lexicographic on triples, so it is a strict total order. Outranking is lexicographic on (priority, tie rank), so it is transitive, and it implies higher rank. ∎

PROPOSITION 1 Omission safety. Let F′ extend F on witnessed and carrier inputs. Suppose Eval_A(F) is AUTHORIZED, or is ESCALATED by escalation e, with deciding decision D (or none, if the outcome came from the default). Then:

- Eval_A(F′) is neither DENIED nor INCOMPLETE;
- if the outcome under F was allow, it is allow under F′;
- if it was escalate(e), then under F′ it is escalate with the *same* escalation e, or it is allow.

Proof.

Step 1: decisions that outrank D. Let X be a block or escalate decision that outranks D.

- X is not 1 under F: it would be a member of Win(F) with higher rank than D (Lemma 3).
- X is not ½ under F, by gate clause (i).
- So X is 0 under F, and by Lemmas 1–2 it is 0 under F′.

Step 2: no decider under F. If the outcome came from the default, every block and escalate decision outranks "none", so by Step 1 each is 0 under F and under F′. The decider under F′, if any, is an allow; otherwise the default stands.

Step 3: the decider D′ under F′, when D exists. D is 1 under F, so 1 under F′, and D′ is D or a decision of higher rank. Suppose D′ ≠ D is a block or escalate decision.

- D′ is not 1 under F, or it would have outranked D in rank under F; it is not 0, since it is 1 under F′. So it is ½ under F.
- By gate clause (i) it does not outrank D. Its rank is higher, so it has D's priority and D's tie rank, hence (the tie order being strict) D's outcome type.
- That type is not block, since D could commit. So both escalate, and by gate clause (ii) they name the same escalation.

So D′ is an allow, or carries D's outcome at D's priority. This last case, an equal-priority escalation listed earlier that takes over with the same escalation, is easy to miss; the mechanized proof below would not close without it.

Step 4: the gate under F′. By Lemmas 1–2, a decision that is ½ under F′ was ½ under F. Let X be such a block or escalate decision.

- If X outranks D′, then X outranks D: D′ is D; or D′ outranks D, and Lemma 3 applies; or D′ has D's priority and tie rank, so outranking D′ is outranking D. Gate clause (i) would have fired under F.
- If gate clause (ii) holds for X against D′, then D′ escalates, so by Step 3 it carries D's escalation at D's priority, and clause (ii) holds for X against D under F.

Either way the gate under F would have fired, contrary to assumption. ∎

Mechanized. Proposition 1, and the corollary that completing the facts never yields DENIED, are proved in Lean 4 for exactly the semantics of §3.3: atoms over partial facts, exists, Kleene's connectives, rank, outranking and both gate clauses (papers/carried-authority/lean/CarriedAuthority.lean in the specification repository, core library only, standard axioms only, checked in seconds). The same file mechanizes the counterexample of §7.5 for the 0.4.0 gate: withholding one fact moves the escalation from two senior approvers to one reviewer, and the 0.5.0 gate fires on it. Writing the mechanized proof corrected two errors in an earlier draft of this paper. That draft defined outranking with position, under which clause (ii) is redundant and does not match the specification. And its proof omitted the case at the end of Step 3.

Proposition 1 is the safety property of Tschantz and Krishnamurthi [2006], refined from grant/deny to the outcome's kind and escalation, and established for every authority by the evaluator rather than per policy [cf. Griesmayer & Morisset 2013]. R-16 proved the case in which D remains the decider [R-16 §7.2]. Three cases are new here:

- a higher permission takes over under the completed facts (Steps 3–4);
- an earlier-listed escalation of equal priority takes over with the same escalation (Step 3);

- equal-priority escalations naming different escalations (gate clause (ii), Step 3).

The third is where the 0.4.0 gate failed. In SPEC-008 0.4.0 the tie order compares only outcome types, so two equal-priority escalations naming different escalations do not gate each other. A carrier that withholds the fact triggering the stricter one obtains the weaker one, while the true facts would have required the stricter (§7.5).

The small-scope check of §7.2 confirms both halves on 20,000 random rule sets whose escalations carry distinct identities:

- the 0.4.0 gate violates the proposition on 265 rule sets;
- the corrected gate violates it on none;
- the two-valued control violates it on 9,271.

5.3 Carrier indifference

THEOREM 7 Sufficiency. Under assumptions A1–A4, a verifier satisfying C1–C4 is carrier-indifferent (Definition 1).

The bound Δ on in-force statements matters here. When an Act establishes currency by an issuer statement, S1 holds up to staleness $\Delta \leq 24$ hours: a version superseded less than Δ ago may still authorize under a statement issued before the supersession.

Proof. Suppose `V_ρ(α, t) = AUTHORIZED`.

S1 (legitimate authority).

- By C1.1, the sealing key resolves to the root ρ pinned. Its effective scope admits the authority, and every key on the path is unrevoked at `t` (Propositions 2–3, §5.6).
- Under A1–A2 the seal was produced by that key.
- By C1.2, the authority is the version ρ's record designates, or one its issuer stated was in force within the last Δ.
- Hence `α.authority ∈ Cur_ρ(t)`, within the stated bound.

S2 (truthful outcome).

- By C2.1, every fact in `F_α` arrived through its declared channel with its declared type. Witnessed facts come from keys certified for that claim, about this subject and inside their windows.
- Action facts are `a` itself. Relying-party facts are σ at verification, and by C4 the commit re-reads them and aborts if they differ.
- Under A1–A2 and A4, witnessed facts are true.
- So `F_α` agrees with `F⋆_t ⊕ a` wherever `F_α` is defined, with carrier inputs taken as presented by definition. Hence `F_α ⊑ F⋆_t ⊕ a`.
- By C2.2 and Proposition 1, the outcome on the true facts is the committed outcome or an allow, and never a refusal. If the commit rests on escalation e, the outcome on the true facts is the same escalation e, which the Act's consents satisfy, or an allow, which needs none.

S3 (approval of exactly this act). Consents are counted only for keys that:

- resolve to ρ's root, with consent scope for `role_e` (C1.1);
- signed the digest recomputed from α (C3).

Under A1 each signature was produced by its key. By collision resistance, the digest an approver signed determines the core recomputed from α.

S4 (once, here).

- The audience check admits α only at the relying party it names.
- The nonce check and atomic recording (C4) admit at most one commit, concurrency included.
- The window bounds how long the ledger must keep the nonce.

S5 (exactly this change). The commit rule (C4) applies `a` and nothing else. ∎

COROLLARY 1 Silence is never rewarded, under the corrected gate. If an Act commits, adding any further true testimony would not have produced a refusal. If an Act does not commit, it is not because a true statement was left out to the carrier's advantage.

For an honest carrier, disclosure weakly dominates: Proposition 1 shows added true facts never turn a commit into a refusal, and Theorem 8 shows full disclosure always reaches the truthful verdict. For a dishonest carrier, omission yields at most INCOMPLETE, which names the facts it left out.

This reverses an incentive. R-16 opened with a familiar argument: absent an identification standard, agents will hide what they are in order to stay useful [R-16 §1]. Under carried authority, hiding buys nothing, because nothing about the verdict depended on what the agent is or on what it chose not to say.

5.4 Completeness, and the price of soundness

THEOREM 8 Completeness under disclosure. If every input read by any decision is present in `F_α` and of its declared kind, then no condition is ½, the gate cannot fire, and `Eval_A` returns the classical outcome. In particular, a carrier that presents truthful testimony for every witnessed input obtains exactly the verdict the rules give on the truth.

Proof. Every atom reads present, in-domain inputs and so is 0 or 1. Kleene's connectives on {0, 1} are the Boolean ones. ∎

The small-scope check confirms Theorem 8 on every fully disclosed, well-kinded assignment it generated (§7.2).

INCOMPLETE is therefore reachable only by leaving something out, or by sending a value of the wrong kind. That is the intended cost: soundness charges the carrier for silence, and nothing else.

There are two consequences for authors of authorities.

COROLLARY 2 Negative facts are values. If κ(x) = wit, then `exists(x)` is 1 or ½, never 0. A block or escalate decision of the form `exists(x)` on a witnessed input can never be cleared. Every permission it outranks is therefore unreachable, and the authority fails safe but never commits. We checked this against the reference verifier with the rule "an open fraud alert blocks the loan" written as `exists(fraud_alert)`:

- With the witness silent, the Act is INCOMPLETE.
- With the witness honestly stating `fraud_alert: false`, the Act is DENIED, because a present value makes `exists` true.
- A null statement is absence, so the Act is again INCOMPLETE.

No testimony clears the rule. The institution must write the condition on the value (`fraud_alert_open == true`), and its witness must state `false`.

This is the formal reason SPEC-008's test-vector witness states "no 90-day delinquency" explicitly [SPEC-008 §6.2]. It is also a rule a linter can check before an authority is sealed: flag any `exists(x)` on a non-authoritative input in a block or escalate decision.

COROLLARY 3 The cost is local. The extra testimony soundness requires is bounded. It covers exactly the inputs read by block and escalate decisions that outrank the would-be outcome, and only when the carrier is trying to commit. Exceptions ranked below the deciding permission cost nothing.

Capability systems pay this cost globally, by requiring a positive discharge for every restriction. The third value lets an institution keep its rules in their natural defeasible form and pay only for the exceptions that could actually stop the act.

5.5 What one compromised key can do

A2 assumed no key outside the carrier's control is compromised. Relaxing it shows that the separation of the three kinds of authority — rules, facts and consents — contains the damage. It also shows one choice that matters more than the specifications previously said.

THEOREM 9 Compromise containment. Let K be a set of compromised keys, and suppose verification otherwise satisfies C1–C4.

1. **The carrier's own keys.** They have no effect. No carrier key is read anywhere in verification.
2. **A witness key** certified for claims C. It can falsify only facts in C, though about any subject. It cannot change rules, other facts or consents. Every committed Act still satisfies S2 on the facts outside C.
3. **Approver keys** for role r, $|K_r|$ of them. They can satisfy escalations requiring role r with at most $|K_r|$ consents. They cannot turn a DENIED or INCOMPLETE result into a commit, because consents are counted only after the rules have escalated with no undecided exception outranking the escalation. Dual control ($n_e = 2$) therefore requires two compromised keys.
4. **A sealing key.** Two cases:
   - If ρ accepts authorities **only from its own record of the version in force** (it never accepts in-force statements), a newly sealed authority from the compromised key is not in force. It authorizes nothing until ρ's record is updated through the issuer's publication channel, which the key alone does not control.
   - If ρ **accepts in-force statements**, the compromised key can sign both an authority and the statement declaring it in force. It then authorizes any action within its certificate's artifact scope, until revocation of the key reaches ρ.
5. **A root key.** It can do anything within its scope. The pinned root is the whole trust boundary [SPEC-007 §10.3].
6. **No single key** can both seal rules and supply the facts or consents those rules read (rev. 1.8.0 §3.5).

Proof. Items 1–3 and 6 follow from the channel, scope and purpose checks in steps 5–9, and from the order of those steps. Rules are evaluated before consents are counted, and the gate precedes counting.

For item 4, the relying party's record pins a seal hash per authority identifier, so a different authority under a recorded identifier is SUPERSEDED. A *new* identifier has no record and can establish currency only through a statement, which a record-only relying party does not accept. Item 5 is immediate. ∎

Item 4 is the theorem's practical content. **A relying party that accepts in-force statements gives up containment of issuer compromise, in exchange for the ability to accept authorities it has never been told about.**

SPEC-008 permits both sources, and it should, because the second one is what lets a system act on authority from an institution it has no prior relationship with. But the choice should be made knowingly. For high-consequence state, a relying party should keep its own record.

Today the specification does not name a record-only configuration. In the reference verifier, an authority identifier with no record and no statement is VERSION_UNVERIFIED, and with a statement it is accepted. We propose that SPEC-008 name record-only mode explicitly.

5.6 Recognition, reproducibility and the record

PROPOSITION 2 No amplification. A key's effective scope is the narrowing of every scope along its path. Any certificate that widens a dimension its parent constrained fails validation. A delegate never exceeds its delegator.

Proof. The validation in §5.1 rejects widening at each hop. ∎

This is the monotone delegation of SPKI [RFC 2693], extended to statement dimensions.

PROPOSITION 3 Revocation is complete. Because revocation names the revoked key's identifier, a revoked key fails validation on every path that reaches it, not only on the path that certified it.

Proof. Validation checks every key on a path against the revoked set (§5.1). ∎

PROPOSITION 4 Order independence. The resolved path, and therefore the verdict, depends on the set of certificates and not on their order. Candidates are enumerated, and tried, in fingerprint order.

A refinement, found while writing this paper. The reference resolver takes the first path that validates *cryptographically*, and only then evaluates that path's effective scope. We certified an issuer key twice from the same root:

- directly, with scope `industry:insurance`;
- through an intermediate, with no scope constraint.

The direct certificate has the lower fingerprint. An Act carrying both paths was refused as OUT_OF_SCOPE. The same Act carrying only the intermediate path was AUTHORIZED. Adding a valid certificate turned an authorization into a refusal.

This is deterministic and fails closed, so it is not exploitable. It is nevertheless a wrongful refusal. Including scope admission in path validation removes it, by trying the next candidate. We record it as a proposed change to SPEC-007 (rev. 1.9), not as a defect of the security argument.

THEOREM 10 Reproducibility. `V_ρ(α, t)` is a deterministic function of:

- the Act;
- ρ's pinned root;
- t;
- the relying-party facts it read;
- the in-force source;
- the nonce ledger;
- the revocation statements dated no later than t.

A receipt that records t, the relying-party facts and the in-force source therefore lets any third party who holds the Act and the public revocation statements recompute the verdict. That party needs no access to the agent, the relying party's systems or any NOMOS service [SPEC-008 §9].

Proof. Each step of SPEC-008 §6 reads only those inputs. Determinism of the digests requires a canonical form that every implementation computes identically [RFC 8785]. Two independent implementations disagreeing on that form is a real failure mode, not a theoretical one: it was the first of R-16's seven defects. ∎

Theorem 10 connects Part II to Part I. Part I asked for an institutional record that makes machine action attributable and inspectable [R-07 §7, Layer 6]. Theorem 10 is what makes that record checkable by someone who does not trust whoever keeps it.

6. The landscape

Table 4 places current proposals against the questions the conditions raise. Each entry describes the approach as its published specification or paper states it, as of October 2026. What a document does not specify is recorded as *not specified*.

TABLE 4 · COMPARISON OF AGENT-AUTHORIZATION APPROACHES

Approach	Authorization and check	Carrier, evidence, and approval
OAuth 2.0 on-behalf-of for agents [draft-oauth-ai-agents-on-behalf-of-user; RFC 8693]	**AUTHORIZED:** The agent, acting for a user, within granted scopes **CHECKED:** Decided at the authorization server; token checked at the resource server	**AGENT INPUT:** Yes: the agent is the *actor* **FACTS:** The authorization server, at issuance **MISSING:** Not specified **APPROVAL:** Scopes at grant time; per-transaction details possible with RFC 9396
AP2 mandates [AP2]	**AUTHORIZED:** A purchase the user signed for, or conditions the user signed in advance **CHECKED:** Merchant, credentials provider, network and issuer	**AGENT INPUT:** Yes: the agent presents the mandates **FACTS:** The agent and the merchant (the merchant signs the cart) **MISSING:** Not specified; pre-authorization inputs are outside the mandate's protection [Aviv et al. 2026] **APPROVAL:** Human present: the user signs a mandate naming the exact goods. Human not present: the user signs the conditions for fulfilment
Visa Trusted Agent Protocol; Know Your Agent [Visa TAP; KYA 2026]	**AUTHORIZED:** Recognition of the agent and of the consumer it acts for **CHECKED:** Merchant	**AGENT INPUT:** Yes: identity is the object **FACTS:** n/a **MISSING:** n/a **APPROVAL:** n/a
Open Agent Passport [Uchibeke 2026]	**AUTHORIZED:** Each tool call, against a policy pack **CHECKED:** Pre-call hook in the agent's framework	**AGENT INPUT:** Yes: the passport's capabilities and limits **FACTS:** Tool arguments and the passport **MISSING:** Missing required fields deny **APPROVAL:** Escalation specified, not yet implemented
aiAuthZ [Kodathala 2026]	**AUTHORIZED:** Each tool call, against role and argument policies **CHECKED:** Gateway, off the agent's host	**AGENT INPUT:** Yes: the role **FACTS:** The call's arguments **MISSING:** Not specified **APPROVAL:** Per-message signature (HMAC) with nonce
Mandato [Racioppi 2026]	**AUTHORIZED:** Each tool call, against a signed mandate chain (tools, argument constraints, oversight class, quota, validity, principal) **CHECKED:** Governance proxy; upstream servers accept sessions only from the proxy's identity (mTLS)	**AGENT INPUT:** Yes: the agent acting under the mandate **FACTS:** The call's arguments **MISSING:** Deny by default; constraints over arguments are conjunctive **APPROVAL:** Synchronous human confirmation per call, for the `confirm` oversight class
Macaroons [Birgisson et al. 2014]	**AUTHORIZED:** The bearer, within caveats **CHECKED:** The target service	**AGENT INPUT:** No: bearer token **FACTS:** The target service (first-party caveats); named third parties (discharges) **MISSING:** Moot: caveats are conjunctive, so silence only removes authority **APPROVAL:** n/a
UCAN 1.0 [UCAN 1.0]	**AUTHORIZED:** The invoker, through a delegation chain, with policies over arguments **CHECKED:** The executor, on receipt of the invocation	**AGENT INPUT:** Yes: the chain ends at the invoker **FACTS:** The invocation's arguments **MISSING:** Moot: policies constrain arguments conjunctively **APPROVAL:** Each delegation, per capability
XACML 3.0 [XACML 3.0]	**AUTHORIZED:** A subject's request **CHECKED:** Enforcement point, via a decision point	**AGENT INPUT:** Yes: subject attributes **FACTS:** Fetched by the decision point from information points **MISSING:** Indeterminate; under deny-overrides it prevents Permit **APPROVAL:** n/a
Notarized Agents [Figuera 2026]	**AUTHORIZED:** Nothing: the receiver records what it observed **CHECKED:** The receiver, after the action	**AGENT INPUT:** n/a **FACTS:** The receiver **MISSING:** n/a **APPROVAL:** n/a
SPEC-007 / SPEC-008	**AUTHORIZED: One exact change** **CHECKED: The system of record**	**AGENT INPUT: No: no carrier signature exists** **FACTS: The channel the authority declares, per input** **MISSING: INCOMPLETE until decided** **APPROVAL: A digest of action, authority, facts, audience, nonce and window**

Four observations follow.

1. The agent is an input almost everywhere. It appears as identity, role, actor, delegate, invoker, or as the holder of a passport or mandate. Macaroons are the exception, being bearer tokens, but there possession *is* authority. That is the failure Hardy described: a deputy holding a token can be steered into using it.

Carrier indifference requires something stronger than bearerness. The carrier may hold the Act's statements and still be unable to use them for anything but the one change they bind.

2. Most checks sit where the carrier can route around them, or before the facts are known. A framework hook runs in the agent's own process. A gateway or proxy enforces only if it holds the only credential. Mandato is the one that requires this explicitly: its upstream servers accept sessions only from the proxy's workload identity, which is precisely C4. An authorization server decides at issuance, before the facts of a particular change exist.

Three designs place the check where it cannot be bypassed: macaroons and UCAN at the target, Mandato at a proxy the target exclusively trusts. All three evaluate positive constraints over the call's own arguments. None carries institutional rules over facts attested by third parties, so the question of what a withheld fact means does not arise for them.

A recent survey of authorization architectures for tool-using agents identifies runtime enforcement as an unresolved gap [Surapani et al. 2026]. The security analysis of AP2 finds that the inputs shaping a transaction before signing are unprotected [Aviv et al. 2026]. Both independently corroborate C4 and C2.

3. Silence is either moot or unspecified. Capability systems, and Mandato's argument constraints, make it moot by expressiveness: positive, conjunctive conditions. XACML makes it safe by fetching the facts itself, at the price of an online dependency on every information source. The attribute-based access control literature has studied the problem as attribute hiding (§2), but no agent-authorization proposal we reviewed specifies what a missing fact means for rules that have exceptions. That is the setting of §4.5, and the one institutional rules occupy.

4. Approval rarely binds the whole act. The strongest bindings in the table:

- the Cart Mandate binds the user's signature to the cart;
- RAR carries per-transaction details;
- aiAuthZ signs each message with a nonce.

None binds an approval jointly to the action, the governing rules, the facts the approver saw, the audience and a single-use nonce. Theorem 6 shows that each omission is exploitable.

These approaches compose with carried authority more than they compete with it.

- An OAuth token or an AP2 mandate can be carried as a consent or as testimony, under a key certified for that role.
- A capability can authorize a carrier to submit Acts at all.
- Agent identification remains the right tool for traffic management and abuse response, which the verdict deliberately ignores.
- Receiver-attested receipts [Figuera 2026] are close kin to SPEC-008's receipt (§5.6). Notarized Agents' receiver records *after* the fact what it observed; the relying party here decides *before* the change whether it is authorized, and records why.

Two further lines of work address problems this paper leaves open:

- zero-knowledge authorization, which proves a policy was satisfied without revealing private attributes [Llambi-Morillas & Fernández-Fernández 2026];
- delegation-provenance chains, which record each hop by which human authorization reached an agent [Dalugoda 2026].

The first speaks to privacy (§9.3). The second speaks to how a principal's instruction reaches the carrier in the first place, which the verdict does not need but an audit may.

7. Evidence

The theorems of §4–§5 are proved on paper. This section reports what we executed to check them, and what we take from the implementation record in R-16.

All scripts are published with the specification, in `papers/carried-authority` of github.com/nomos-spec/spec:

- `ablate.py` runs the necessity witnesses;
- `modelcheck2.py` runs the small-scope check;
- `lean/CarriedAuthority.lean` is the mechanized proof of §5.2.

They run against the SPEC-008 0.4.0 reference verifier and generator, frozen byte-for-byte alongside them, using the published deterministic seeds [SPEC-008 §10]. Anyone can re-run them (Appendix A).

7.1 Necessity witnesses (Table 3)

For each obligation we built a weakened verifier: the reference verifier with that one obligation removed by a single textual patch, all others intact. We then presented an attack Act whose every signature is genuine and was produced by its key's holder.

TABLE 3 · EXECUTED NECESSITY WITNESSES

#	Obligation removed	Attack, using genuine statements only	Observed result
1	C1.1 pinned root	An allow-everything authority sealed under a root the Act supplies	**Weakened:** AUTHORIZED **Full:** ISSUER_NOT_RECOGNIZED **True facts / approvers:** never issued
2	C1.2 currency	Version 0.9.0, superseded but never revoked, lacks the review threshold; a $20,000 loan	**Weakened:** AUTHORIZED **Full:** SUPERSEDED **True facts / approvers:** ESCALATED
3	C2.1 provenance	The agent asserts a credit score of 768; the bureau holds 600	**Weakened:** AUTHORIZED **Full:** FACT_SOURCE_VIOLATION **True facts / approvers:** DENIED
4	C2.2 omission	Bankruptcy statement withheld (bankrupt in the world)	**Weakened:** AUTHORIZED **Full:** INCOMPLETE **True facts / approvers:** DENIED
5	C2.2 omission	DTI statement withheld (58%), two genuine consents attached	**Weakened:** AUTHORIZED **Full:** INCOMPLETE **True facts / approvers:** DENIED
6	C2.2 omission	Fraud alert withheld, rule written with `exists`	**Weakened:** AUTHORIZED **Full:** INCOMPLETE **True facts / approvers:** DENIED
7	C2.2 omission	`terms.months` sent as the string "480" in an object parameter	**Weakened:** AUTHORIZED **Full:** INCOMPLETE **True facts / approvers:** DENIED
8	C3 action	Consents for $75,000 reused for $750,000	**Weakened:** AUTHORIZED **Full:** CONSENT_INVALID **True facts / approvers:** approved $75,000
9	C3 testimony	Approvers shown DTI 31%; a later genuine 48% statement swapped in	**Weakened:** AUTHORIZED **Full:** CONSENT_INVALID **True facts / approvers:** approved on 31%
10	C3 audience	Consents for the core ledger presented at the card ledger	**Weakened:** AUTHORIZED **Full:** CONSENT_INVALID **True facts / approvers:** approved one ledger
11	C3 nonce	Committed Act resubmitted with a fresh nonce	**Weakened:** AUTHORIZED **Full:** CONSENT_INVALID **True facts / approvers:** approved one commit
12	C3 seal hash	Consents to R6 under the base policy satisfy a cosigner-waiver escalation under another in-force policy	**Weakened:** AUTHORIZED **Full:** CONSENT_INVALID **True facts / approvers:** approved R6

All twelve are witnessed: the weakened verifier commits, the full verifier refuses, and the commit is one the true facts or the approvers did not support.

A thirteenth ablation removes order independence: first match in presentation order. The same certificate set then gives TESTIMONY_INVALID when an expired path comes first and AUTHORIZED when the valid path comes first. The full verifier gives AUTHORIZED in both orders. This is the selection effect of Theorem 2(b).

C4, the point of effect, is not ablated. Its theorem concerns where a credential lives, not what a verifier computes (§4.1).

7.2 Small-scope check of omission safety

Proposition 1 quantifies over all authorities and all partial facts. Following the small-scope hypothesis — that most counterexamples to a property have small instances [Jackson 2006] — we checked it exhaustively within a small scope, against the reference evaluator itself rather than a model of it.

Scope. 20,000 random authorities (seed 2026), each with:

- one to five decisions with conditions of nesting depth up to two over `and`, `or` and `not`;
- atoms drawn from all six comparisons and `exists`;
- outcomes allow, block and escalate, each escalation naming one of two escalations;
- priorities drawn so that ties are frequent;
- a tie order drawn from all three;
- a random default outcome.

The vocabulary is three witnessed inputs (two numeric, one boolean) and one relying-party input. Numeric domains include a present value of the wrong kind.

Method. For each authority we enumerated every partial assignment of the witnessed inputs a carrier could present. For each assignment the evaluator would commit on, we enumerated every completion — every way the world could be — and checked Proposition 1:

- no completion yields a refusal or INCOMPLETE;
- an allow stays an allow;
- an escalation stays the *same* escalation.

We also checked Theorem 8 on every fully disclosed, well-kinded assignment.

Evaluators. Three evaluators ran over identical rule sets:

- the published SPEC-008 0.4.0 reference evaluator;
- the same evaluator with the corrected gate of §3.3;
- a control with C2.2 removed, in which a decision whose condition is undecided is treated as not holding and the gate is gone.

The control is a milder weakening than reading every absent atom as false, as SPEC-008 0.2.0 did: under `not`, that reading is unsound in further ways.

MODEL-CHECK RESULTS

Measure	Evaluator results
Committing partial states checked	**0.4.0:** 719,750 **Corrected gate:** 710,978 **Two-valued:** 1,007,312
Soundness violations	**0.4.0:** 2,320 **Corrected gate:** **0** **Two-valued:** 144,001
Rule sets with at least one violation	**0.4.0:** 265 of 20,000 (1.3%) **Corrected gate:** **0 of 20,000** **Two-valued:** 9,271 of 20,000 (46.4%)
Completeness violations	**0.4.0:** 0 **Corrected gate:** 0 **Two-valued:** 0

The two-valued control is unsound on 46% of small random authorities. The failure is not a corner case: it appears whenever an exception outranks a permission and reads a fact the carrier can withhold.

The published 0.4.0 gate fails on 1.3% of them, every failure an instance of the equal-priority escalation gap of §7.5. The corrected gate fails on none.

The difference in committing states, 1,007,312 against 710,978, is the price of soundness at this scope. The corrected gate declines to commit in 29% of the partial states the two-valued control would commit. Every one of those states is INCOMPLETE, and the response names the facts that would decide it. Disclosing them reaches the classical verdict (Theorem 8), which commits when the truth permits and refuses when it does not.

An earlier run in which every escalation shared one identity found no violation for 0.4.0. That is consistent: the gap needs two different escalations. We report it only to explain why the gap survived every earlier test.

7.3 Conformance and cross-implementation evidence

We re-ran the SPEC-008 vectors against the reference verifier. Under 0.4.0, all 29, plus the two non-finite cases built in memory, pass (31/31). Under 0.5.0, all 30 pass (32/32), and the platform's TypeScript verifier passes all 30.

R-16 reports evidence we did not re-run. The two implementations (the Python reference and the platform's TypeScript verifier):

- agree on all 29 vectors and on 729 derived mutants: 125 with one statement removed, 561 with one fact replaced by a value of another kind, and 43 with the in-force evidence altered;
- agree on every normative field, including the binding digest;
- verify the dual-consent Act in a median of 1.10 ms (TypeScript) and 3.35 ms (Python) on a laptop [R-16 §9.2–9.6].

7.4 The defect record, read through the obligations

R-16 reports seven defects found while building the second implementation and testing the stated properties. All are fixed and each is pinned by a vector [R-16 §9.4]. Classified by the obligation each violated:

Defect (R-16)	Obligation violated
Withheld fact read as false, including through consent	C2.2
Wrong-kind value read as false (incl. nested fields, NaN)	C2.2
`exists` decided by absence for witnessed facts	C2.2
Order-dependent chain resolution on tree-shaped sets	C1.1 (order)
Superseded, unrevoked version authorized a new action	C1.2
RFC 8785 key order differed between implementations	Reproducibility (Theorem 10); digests in C3
Receipts lacked verification time and relying-party facts	Reproducibility (Theorem 10)

Three of the seven are failures of omission safety, the obligation §4.5 identifies as the departure from capability-based practice. This distribution is evidence for the paper's central claim about difficulty. The authors of the specification, writing with the property in mind, violated C2.2 three separate times, each by a different route: absence, wrong kind, and `exists`. The obligations are not obvious, and stating them is not a formality.

7.5 What writing the proofs found

Proving the theorems against the reference verifier, rather than restating them, exposed one soundness gap and two refinements. None is caught by the 29 vectors, the 729 mutants or the agreement between the two implementations.

A soundness gap: the carrier chooses who approves. Consider the base lending authority with one added decision, `R_hr`: a flagged high-risk applicant needs dual senior approval. It is an escalation at the same priority as the existing review threshold R10, which needs a single reviewer. Take a $20,000 loan.

THE EQUAL-PRIORITY ESCALATION GAP

Act	SPEC-008 0.4.0 result
Bureau states `high_risk_flag: true` (the truth)	ESCALATED · **Decider:** R_hr · **Approval:** two senior approvers
Flag withheld; one reviewer consents	**AUTHORIZED** · **Decider:** R10 · **Approval:** one reviewer

In 0.4.0, the tie between two equal-priority escalations is ordered only by outcome type. Neither escalation outranks the other, so the undecided stricter one does not gate the weaker one. By staying silent, the carrier selects which people approve, and S3 fails.

The TypeScript verifier deployed on the platform implements the same rule, so the two implementations agree on the wrong answer. This is the limit of cross-implementation testing: agreement establishes consistency, not correctness. Only a property stated precisely enough to be checked caught it.

The fix is gate clause (ii) of §3.3: an undecided escalation of equal priority that names a different escalation also returns INCOMPLETE. It makes the decider's tie-break by list position normative, which both implementations already follow.

The small-scope check of §7.2, with escalations carrying one of two identities, gives:

MODEL-CHECK RESULTS

Measure	Evaluator results
Committing partial states checked	**0.4.0:** 719,750 **Corrected gate:** 710,978 **Two-valued:** 1,007,312
Soundness violations	**0.4.0:** 2,320 **Corrected gate:** **0** **Two-valued:** 144,001
Rule sets affected	**0.4.0:** 265 (1.3%) **Corrected gate:** **0** **Two-valued:** 9,271 (46.4%)
Completeness violations	**0.4.0:** 0 **Corrected gate:** 0 **Two-valued:** 0

The gap is narrow: it needs two escalations of equal priority with different requirements. It is real, because institutions write such pairs whenever two independent conditions each require approval at the same level. SPEC-008 0.5.0 adds clause (ii) and a thirtieth vector, `equal_priority_escalation_withheld`. On that vector, the 0.4.0 reference verifier returns AUTHORIZED on a single reviewer's consent, and the 0.5.0 verifier returns INCOMPLETE, naming R16 and `days_since_delinquency`. The 29 earlier vectors are byte-identical, and all 32 cases pass. The platform verifier carries the same fix: unfixed, it returns AUTHORIZED on vector 30; fixed, it passes all 30. Before the release we checked, read-only, the 139 authorities in the platform repository, its five public demonstration authorities and the EU AI Act authority on its Exchange (851 rules); none contained the pattern.

What the fix leaves to the author. Clause (ii) stops the carrier from choosing between equal-priority escalations. It does not decide what *should* happen when both apply with every fact disclosed. Under the rank of §3.3, the one listed first governs. That rule is deterministic and sealed, so the carrier cannot exploit it. But it carries an authoring hazard: an institution that lists its dual-approval rule second gets single approval whenever both apply. There are two ways to close it:

- **(a)** require every applicable escalation of the decider's priority to be satisfied. Consents then accumulate across escalations, and the verdict reports each one;
- **(b)** make two equal-priority escalations naming different escalations a sealing-time error, so the ambiguous form cannot be sealed. The author resolves it by assigning distinct priorities.

The next revision of SPEC-008 should adopt (b). It changes no evaluation semantics, and it turns the ambiguity into a review question at the point where a human already reviews the authority. Option (a) remains available to a later version.

Two refinements. Neither breaks a security property:

- **Scope-blind path selection (SPEC-007).** Adding a valid certificate that offers a second path, with a narrower scope and a lower fingerprint, turns an authorization into OUT_OF_SCOPE (§5.6). The proposed fix includes scope admission in path validation.
- **Record-only currency (SPEC-008).** Issuer-key compromise is contained only if the relying party establishes currency from its own record (Theorem 9.4). The specification does not yet name that configuration. The proposed fix names it.

A further observation concerns how authorities are written, not the verifier. It is Corollary 2: `exists` over a witnessed input in a block or escalate decision can never be cleared. It belongs in the authoring lint that runs before an authority is sealed.

8. What changes

The theorems are narrow on purpose. Their consequences are not. Five follow directly.

The unit of authorization moves from the actor to the act. Access control has been organized around a subject for fifty years: who is asking, and what has that identity been granted? Carrier indifference makes the subject irrelevant to the verdict, not just unimportant. The object that is authorized is a change, and the change carries the reason it is permitted.

This is not a claim that identity is useless. Systems still need to know who is calling in order to manage traffic, apportion cost and respond to abuse. It is a claim about which question the verdict answers. For a consequential change, the question is not "who are you?" but "under what authority, on what facts, with whose approval, is *this* permitted now?"

The system of record becomes the authority boundary. Today an institution's authority is enforced wherever someone remembered to put a check: in a gateway, in the agent's framework, in the authorization server's view of the world at token time. Under C4 it is enforced where the state lives, by the only component that can change it.

That relocation has an organizational consequence. The party accountable for a ledger is also the party that verifies every change to it. R-18 called this role the Sovereign. Theorem 10 makes its decisions checkable by anyone.

Disclosure becomes the dominant strategy. An argument now common in agent security says agents will hide what they are, because disclosure gets them throttled or refused [R-16 §1]. Under carried authority, with the corrected gate, there is nothing to gain by hiding (Corollary 1):

- the verdict never depended on what the carrier is;
- omission yields at most INCOMPLETE, a response that names exactly what is missing;
- an honest carrier that discloses every fact reaches the truthful verdict.

The arms race over detection does not need to be won, because it no longer decides anything that matters.

Authority becomes portable across institutional boundaries. An Act verifies against one root the relying party chose, with no call to the issuer and no prior integration (§4.2, §5.1). Two institutions that have never exchanged an API key can rely on each other's authority, provided one pins a root that certifies the other. Payment networks did this for cards and border systems did it for passports. Carried authority does it for any consequential change an institution is willing to delegate.

Human authority moves to where it is effective. Part I argued that people should establish, approve and withdraw authority, and machines should enforce it [R-07 §18]. The construction gives each human role a precise cryptographic form [R-18]:

- **Issuers** seal authorities.
- **Witnesses** state facts within certified scopes.
- **Approvers** consent to exact acts.
- **Sovereigns** pin roots and keep records.

Theorem 9 shows that each role's failure is contained to that role, except for the one choice — record versus in-force statement — that a Sovereign must make knowingly. No role can be performed by the agent. When the rules reserve a decision for a person, ESCALATED routes it to one. When the evidence is missing, INCOMPLETE says so, and no approval can paper over it (§5.2).

The thesis behind the theorems. §1.1 makes a historical claim that the theorems do not prove: institutional decision-making at the scale of autonomous agents becomes stable only once infrastructure exists that lets strangers coordinate on the same rules without trusting the party that carries them. The claim is falsifiable. If institutions scale consequential agent decisions without any mechanism that makes the agent irrelevant to the verdict, it fails. If they cannot, it stands. The theorems settle the narrower question exactly: what such a mechanism must guarantee, and that four conditions are both necessary and sufficient.

9. Limitations and open problems

The theorems hold under stated assumptions. Each assumption marks a problem the protocol does not solve. We state them as the open problems of the discipline.

9.1 Who should operate roots. Every guarantee rests on a root the relying party pins (C1.1, Theorem 9.5). Bilateral pinning works between institutions that already trust each other. It does not by itself create the many-to-many recognition that card networks or ICAO's public key directory provide. Who should operate shared roots, under what governance and with what liability, is a question of institutional design, not cryptography. We have argued it separately and leave it open [R-09].

9.2 Whether witnesses tell the truth. Assumption A4 is load-bearing. Certification bounds *which* facts a key may state, not whether its statements are true. A false statement from a correctly certified witness produces a commit that S2 does not forbid. That is the bounded damage of Theorem 9.2, but it is still damage.

What the protocol contributes is that the choice of witness becomes explicit, signed, scoped, revocable and visible in every Act and receipt. How institutions should select, audit and hold witnesses liable is open.

9.3 Privacy. Testimony values are visible to the carrier, which assembles the Act, and to the relying party. Selective disclosure [RFC 9901] would hide fields from parties who do not need them. Zero-knowledge proofs of policy satisfaction would hide the facts while proving the verdict [Llambi-Morillas & Fernández-Fernández 2026]. Encryption to the relying party would hide them from the carrier. Each interacts with Theorem 10, because a third party re-verifying the verdict needs the facts or a proof about them. The right trade-off between confidentiality and reproducibility is open.

9.4 Revocation latency. A revoked key is refused once the revocation reaches the relying party (Proposition 3). How fast it reaches the relying party is a distribution problem. Freshness staples cover issuer chains [SPEC-007 §5.5]. They are not yet specified for witness and approver keys.

9.5 Effects without a system of record. C4 needs a component that holds the only credential able to make the change. Sending a message, publishing a statement and moving a physical actuator often have no such component. An outbound gateway that holds the only credential can serve as the relying party, and inherits all of C4's duties. Specifying it is open.

9.6 Legacy systems. A system that cannot verify Acts itself needs an adapter. The adapter satisfies C4 only if it holds the only credential able to change the system behind it (Theorem 1). Many legacy systems grant broad credentials to many clients, and narrowing them is an operational problem, not a protocol one.

9.7 What the rules can say. The rule language compares a field with a value. It does not compare two fields arithmetically. A cumulative limit such as "consumed before plus this amount must not exceed the budget" is therefore enforced by the relying party's ledger, not by the rules [R-16 §10.8]. This keeps evaluation decidable, cheap and reproducible. It also places some institutional constraints in the relying party rather than in the authority. Where the boundary between the two should sit is open.

9.8 Independence of evidence. Both SPEC-008 implementations, the ablation harness and the model check were written by the specification's authors. The ablations show that each obligation is necessary *to the reference verifier's guarantees*. The model check confirms Proposition 1 *within a small scope*. The mechanized proof (§5.2) establishes it for all authorities, against a model of the semantics written by the same authors; it does not establish that either implementation matches that model.

A second, independent implementation that agrees on the published vectors is the condition for promoting the specifications from Draft, and the strongest test of this paper's claims.

10. Conclusion

Each time institutions scaled to coordination among strangers, they first needed infrastructure that let strangers rely on the same rules without trusting one another. Computable Authority defined an authorized action by an equation with no term for the agent. That is a statement about what authority is. This paper has asked what it takes for the statement to survive deployment, where the agent carries every input the equation reads.

The answer is four conditions:

- the authority is genuine and current;
- the facts are proven;
- the action is bound;
- the verdict is computed where the change lands.

They are jointly sufficient for carrier indifference. Each of the six obligations they refine into is necessary on its own. We proved both directions. We executed every necessity construction against the reference verifier, and we checked the central safety property exhaustively at small scope, against a control that fails it on nearly half of all rule sets.

The most consequential of the six concerns silence. Institutional rules are defeasible: exceptions override permissions, because that is how law is written. Over facts an adversary carries, a verifier that reads silence as falsity lets the carrier decide which exceptions exist. Access-control research has long known the attack as attribute hiding, and has answered it by certifying policies one at a time or by restricting them to monotone forms. Capability systems never met it, because their conditions are positive. Policy engines avoided it by fetching the facts themselves. When the adversary carries every fact, the remedy has to sit in the evaluator: a third truth value and a single gate, which make every prioritized defeasible rule set safe without restricting how it is written. The gate has to cover the case SPEC-008 0.4.0 missed: two equal-priority escalations naming different escalations. Safety must preserve who approves, not only whether. With both clauses of the gate, silence is never rewarded and disclosure always suffices.

The consequence is a change in what security protects at an agent boundary. The question stops being who is calling and becomes whether this change, under this authority, on these facts, now, is permitted, decided by the only system that can make it. An agent that holds no authority cannot misuse any. At worst it is a confused courier. Everything that makes the change legitimate, it carries but cannot choose.

The specifications, vectors, reference verifier, the scripts behind §7 and the Lean proof are published at github.com/nomos-spec/spec under CC BY 4.0. We invite independent implementations, attempts to break the obligations, and corrections to Table 4.

Appendix A. Reproducing §5.2 and §7

Everything runs on a laptop, with no network access and no secrets, from the specification repository (`github.com/nomos-spec/spec`).

```
cd spec/papers/carried-authority
python3 ablate.py vectors_040.json       # Table 3: 12/12 necessity witnesses, plus the order ablation
python3 modelcheck2.py 20000 2026        # §7.2: three evaluators, ~17 minutes
cd lean && lean CarriedAuthority.lean    # §5.2: Proposition 1 and the 0.4.0 counterexample (Lean 4.35, no Mathlib)

cd ../../../act-vectors
python3 check.py vectors.json            # 32/32 under SPEC-008 0.5.0 (31/31 under 0.4.0)
```

`papers/carried-authority` holds the SPEC-008 0.4.0 verifier, generator and vectors, frozen byte-for-byte from commit `c8fa382`, so the evidence reproduces against exactly the version it describes. `act-vectors/` holds 0.5.0, which adds gate clause (ii) of §3.3 and the thirtieth vector, `equal_priority_escalation_withheld` (`nomos-spec/spec` pull request 11); the 29 earlier vectors are byte-identical. Rerun from the repository, `ablate.py` reproduces `ablation-results.json` byte for byte.

References

Classical foundations

[Abadi & Needham 1996] M. Abadi, R. Needham. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering* 22(1):6–15, 1996.

[Appel & Felten 1999] A. W. Appel, E. W. Felten. Proof-Carrying Authentication. *ACM Conference on Computer and Communications Security*, 1999.

[Bauer et al. 2005] L. Bauer, S. Garriss, M. K. Reiter. Distributed Proving in Access-Control Systems. *IEEE Symposium on Security and Privacy*, 2005.

[Blaze et al. 1996] M. Blaze, J. Feigenbaum, J. Lacy. Decentralized Trust Management. *IEEE Symposium on Security and Privacy*, 1996.

[Bruns & Huth 2011] G. Bruns, M. Huth. Access Control via Belnap Logic: Intuitive, Expressive, and Analyzable Policy Composition. *ACM Transactions on Information and System Security* 14(1):9, 2011.

[Clark & Wilson 1987] D. D. Clark, D. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. *IEEE Symposium on Security and Privacy*, 184–194, 1987.

[Crampton & Huth 2010] J. Crampton, M. Huth. An Authorization Framework Resilient to Policy Evaluation Failures. *ESORICS*, LNCS 6345, 472–487, 2010.

[Crampton & Morisset 2012] J. Crampton, C. Morisset. PTaCL: A Language for Attribute-Based Access Control in Open Systems. *Principles of Security and Trust (POST)*, LNCS 7215, 390–409, 2012.

[Crampton & Morisset 2014] J. Crampton, C. Morisset. Monotonicity and Completeness in Attribute-Based Access Control. *Security and Trust Management (STM)*, LNCS 8743, 33–48, 2014.

[Crampton et al. 2015] J. Crampton, C. Morisset, N. Zannone. On Missing Attributes in Access Control: Non-deterministic and Probabilistic Attribute Retrieval. *ACM Symposium on Access Control Models and Technologies (SACMAT)*, 99–109, 2015.

[Dennis & Van Horn 1966] J. B. Dennis, E. C. Van Horn. Programming Semantics for Multiprogrammed Computations. *Communications of the ACM* 9(3):143–155, 1966.

[Fitting 1994] M. Fitting. Kleene's Three Valued Logics and Their Children. *Fundamenta Informaticae* 20(1–3):113–131, 1994.

[Greif 2006] A. Greif. *Institutions and the Path to the Modern Economy: Lessons from Medieval Trade*. Cambridge University Press, 2006.

[Griesmayer & Morisset 2013] A. Griesmayer, C. Morisset. Automated Certification of Authorisation Policy Resistance. *ESORICS*, LNCS 8134, 2013. arXiv:1306.4624.

[Hardy 1988] N. Hardy. The Confused Deputy (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review* 22(4):36–38, 1988.

[Hart 1949] H. L. A. Hart. The Ascription of Responsibility and Rights. *Proceedings of the Aristotelian Society* 49 (1948–1949):171–194, 1949.

[Hart 1961] H. L. A. Hart. *The Concept of Law*. Clarendon Press, Oxford, 1961; 3rd ed. 2012, ch. 5.

[Jackson 2006] D. Jackson. *Software Abstractions: Logic, Language, and Analysis*. MIT Press, 2006.

[Kleene 1952] S. C. Kleene. *Introduction to Metamathematics*. North-Holland, 1952.

[Kula 1986] W. Kula. *Measures and Men*. Trans. R. Szreter. Princeton University Press, 1986.

[Lampson et al. 1992] B. Lampson, M. Abadi, M. Burrows, E. Wobber. Authentication in Distributed Systems: Theory and Practice. *ACM Transactions on Computer Systems* 10(4):265–310, 1992.

[Lee 1977] G. A. Lee. The Coming of Age of Double Entry: The Giovanni Farolfi Ledger of 1299–1300. *Accounting Historians Journal* 4(2):79–95, 1977.

[Necula 1997] G. C. Necula. Proof-Carrying Code. *ACM Symposium on Principles of Programming Languages (POPL)*, 1997.

[North 1990] D. C. North. *Institutions, Institutional Change and Economic Performance*. Cambridge University Press, 1990.

[North et al. 2009] D. C. North, J. J. Wallis, B. R. Weingast. *Violence and Social Orders: A Conceptual Framework for Interpreting Recorded Human History*. Cambridge University Press, 2009.

[Pacioli 1494] L. Pacioli. *Summa de arithmetica, geometria, proportioni et proportionalita*. Venice, 1494. Distinctio IX, Tractatus XI, *De computis et scripturis*.

[Peragallo 1938] E. Peragallo. *Origin and Evolution of Double Entry Bookkeeping: A Study of Italian Practice from the Fourteenth Century*. American Institute Publishing, New York, 1938.

[Prakken & Sartor 1997] H. Prakken, G. Sartor. Argument-Based Extended Logic Programming with Defeasible Priorities. *Journal of Applied Non-Classical Logics* 7:25–75, 1997.

[Roth 1997] M. T. Roth. *Law Collections from Mesopotamia and Asia Minor*. 2nd ed. Scholars Press, Atlanta, 1997.

[Saltzer & Schroeder 1975] J. H. Saltzer, M. D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE* 63(9):1278–1308, 1975.

[Seabright 2004] P. Seabright. *The Company of Strangers: A Natural History of Economic Life*. Princeton University Press, 2004.

[Tschantz & Krishnamurthi 2006] M. C. Tschantz, S. Krishnamurthi. Towards Reasonability Properties for Access-Control Policy Languages. *ACM Symposium on Access Control Models and Technologies (SACMAT)*, 160–169, 2006.

[Zucker 1986] L. G. Zucker. Production of Trust: Institutional Sources of Economic Structure, 1840–1920. *Research in Organizational Behavior* 8:53–111, 1986.

Standards and specifications

[ICAO 9303] ICAO. Doc 9303, *Machine Readable Travel Documents*, Part 12: Public Key Infrastructure for MRTDs. 8th ed., 2021.

[LegalRuleML 2021] OASIS. *LegalRuleML Core Specification Version 1.0*. OASIS Standard, 30 August 2021.

[Metre Convention 1875] *Convention du Mètre*. Signed at Paris, 20 May 1875; establishes the International Bureau of Weights and Measures (BIPM).

[NIST SP 800-162] V. C. Hu et al. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. NIST Special Publication 800-

162, 2014.

[RFC 2693] C. Ellison et al. SPKI Certificate Theory. 1999.

[RFC 2704] M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis. The KeyNote Trust-Management System, Version 2. 1999.

[RFC 8032] S. Josefsson, I. Liusvaara. Edwards-Curve Digital Signature Algorithm (EdDSA). 2017.

[RFC 8693] M. Jones et al. OAuth 2.0 Token Exchange. 2020.

[RFC 8725] Y. Sheffer, D. Hardt, M. Jones. JSON Web Token Best Current Practices. February 2020.

[RFC 8785] A. Rundgren, B. Jordan, S. Erdtman. JSON Canonicalization Scheme (JCS). 2020.

[RFC 9396] T. Lodderstedt, J. Richer, B. Campbell. OAuth 2.0 Rich Authorization Requests. 2023.

[RFC 9421] A. Backman, J. Richer, M. Sporny. HTTP Message Signatures. 2024.

[RFC 9901] D. Fett, K. Yasuda, B. Campbell. Selective Disclosure for JSON Web Tokens (SD-JWT). November 2025.

[XACML 3.0] OASIS. *eXtensible Access Control Markup Language (XACML) Version 3.0*. OASIS Standard, 2013.

[Biscuit] Eclipse Biscuit. *Biscuit Authorization Token Specification*.

[UCAN 1.0] UCAN Working Group. *UCAN Delegation Specification v1.0*. github.com/ucan-wg/delegation.

[Birgisson et al. 2014] A. Birgisson, J. G. Politz, Ú. Erlingsson, A. Taly, M. Vrable, M. Lentczner. Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud. *NDSS*, 2014.

Agents and agent authorization

[AP2] Google and partners. *Agent Payments Protocol (AP2)*. Announced 16 September 2025.

[Aviv et al. 2026] A. Aviv, P. A. Gandh, R. Bitton, A. Shabtai. Beyond the Mandate: A Systematic Security Analysis of the Agent Payments Protocol (AP2). arXiv:2608.23858, August 2026.

[Dalugoda 2026] A. Dalugoda. HDP: A Lightweight Cryptographic Protocol for Human Delegation Provenance in Agentic AI Systems. arXiv:2604.04522, April 2026.

[Debenedetti et al. 2024] E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, F. Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. *NeurIPS Datasets and Benchmarks Track*, 2024.

[draft-oauth-ai-agents-on-behalf-of-user] OAuth 2.0 Extension: On-Behalf-Of User Authorization for AI Agents. IETF Internet-Draft, work in progress.

[Figuera 2026] J. Figuera. Notarized Agents: Receiver-Attested Confidential Receipts for AI Agent Actions. arXiv:2606.04193, June 2026.

[Greshake et al. 2023] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, M. Fritz. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. *ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023.

[Kodathala 2026] S. V. Kodathala. aiAuthZ: Off-Host, Identity-Bound Authorization for AI Agents. arXiv:2607.05518, July 2026.

[KYA 2026] Ant International, Mastercard, Visa. Know-Your-Agent interoperability collaboration. Announcement, September 2026.

[Llambi-Morillas & Fernández-Fernández 2026] M. Llambi-Morillas, D. Fernández-Fernández. Cryptographically Verifiable Authorization for Autonomous AI Agents: A Falsifiable Hypothesis and Proof of Concept. arXiv:2607.21325, 2026.

[Racioppi 2026] G. Racioppi. Mandato: Protocol-Level Enforcement of Digitally Signed Mandates on AI Agent Actions with Cryptographically Chained Audit Trails. arXiv:2608.14074, August 2026.

[Surapani et al. 2026] R. K. Surapani, P. K. D. Kakitapelli, A. Morampudi, P. Padi. Authorization Architectures for Tool-Using AI Agents. arXiv:2609.15906, September 2026.

[Uchibeke 2026] Uchi Uchibeke. Before the Tool Call: Deterministic Pre-Action Authorization for Autonomous AI Agents (the Open Agent Passport). arXiv:2603.20953, March 2026.

[Visa TAP] Visa. *Trusted Agent Protocol*. Developer documentation, 2025.

This series and its specifications

[R-07] A. Sendagi. Computable Authority: A Runtime Reference Architecture for Machine-Executable Law and Governance. Computable Authority Research Library, R-07, August 2026. Submitted to the OECD Consultation on the Digital Provision of Law. computableauthority.com/research/computable-authority-a-runtime-reference-architecture

[R-09] Root Legitimacy for Machine-Executable Institutional Authority. Computable Authority Research Library, R-09. computableauthority.com/research/root-legitimacy

[R-16] Passports for Agents. Computable Authority Research Library, R-16, 7 October 2026 (revised 8 October 2026). computableauthority.com/research/passports-for-agents

[R-17] The Computable Authority Equation. Computable Authority Research Library, R-17, 8 October 2026. computableauthority.com/research/the-computable-authority-equation

[R-18] What Remains Human. Computable Authority Research Library, R-18, October 2026. computableauthority.com/research/what-remains-human

[SPEC-006] NOMOS-SPEC-006: Artifact Revocation. github.com/nomos-spec/spec.

[SPEC-007] NOMOS-SPEC-007: Chain-of-Trust Key Certificates, v1.7.0, with Revision 1.8.0 (Statement Keys, Target-First Resolution). Draft. github.com/nomos-spec/spec.

[SPEC-008] NOMOS-SPEC-008: Act Binding, v0.5.0 (the gap of §7.5 is in v0.4.0). Draft. github.com/nomos-spec/spec.