Attested Governance:
Runtime Integrity for Autonomous Systems

Jack Brennan
Attested Intelligence, Illinois, USA
admin@attestedintelligence.com

Abstract

Autonomous AI systems face four critical governance problems: attestation of subject identity and authorized behavior, runtime enforcement of policy, generation of cryptographically meaningful evidence, and continuity verification of decision history. These four problems are coupled by design: any architecture that solves them must address all four under one trust root. The case making them actively pressing now is the rise of autonomous AI agents that manage workflows, invoke external tools, and modify state at production scale without cryptographic records that survive forensic scrutiny. We name this property inseparability. The architecture either binds all four pillars under one cryptographic chain (integrated) or it does not (composed). This commitment is made at the time of design. Composing independent provenance, policy, attestation, and logging produces parallel verification with multiple independent trust roots. Integration requires the architectural commitment to a unified receipt format and trust model from system genesis.

We define Attested Governance as the integrated category and present AGA as one implementation. The category supports multiple architectural realizations; Section 3.4 reviews peer instantiations. We derive the integration property within an explicit trust boundary that names two assumptions upfront: the issuing authority and the portal are not compromised. We present AGA in detail including its primitive choices, trust model design, and operational framing. We anchor the architectural pattern in three independent regulated sectors (industrial control under ISA/IEC 62443-3-3 and NIST SP 800-82, custody transfer measurement under API MPMS Chapter 21.1, and electronic records integrity under 21 CFR Part 11) where the same four-pillar pattern has operated for decades under high consequence requirements where evidence must survive external audit. A reference implementation establishes an empirical baseline: a complete signed measurement cycle completes in approximately 4.94 ms on commodity hardware (reference implementation, v0.9.1 snapshot; see Section 4.7), with post-quantum hybrid signing adding only 0.39 ms per governance decision (Hybrid Sign + Verify mean). We propose six minimum criteria for standardization within the category, identify four open research directions, and identify three IETF working groups (SCITT, RATS, WIMSE) as standardization pathways.

The proposed category is motivated by high consequence contexts demanding forensic review. We do not claim it applies to all AI governance. The architectural debt of partial governance becomes more expensive to retire as autonomous systems scale into higher consequence deployments, and the same forensic scrutiny pressure that drove regulated sectors (Section 5) toward integration applies here. Whether a coherent category with interoperable implementations emerges depends on whether the integration question is engaged structurally now.

Keywords: attested governance, runtime integrity, AI agent governance, cryptographic attestation, inseparability property, four-pillar pattern, evidence verification, cryptographic runtime governance, attested intelligence.

1 Introduction

Autonomous AI agents are now invoking external tools, modifying state, and executing transactions at scale. Over the past two years, the deployment surface for such agents has expanded from research demos to production environments in regulated industries, federal procurement contexts, and consumer applications. A single autonomous agent in production routinely loads a system prompt, invokes external tools through dynamic protocols, modifies records in downstream systems, and retains memory across sessions. None of this produces a cryptographic record of which policy authorized which action against which agent configuration. When something goes wrong, the operator faces a forensic question with no forensic answer: which decisions were made by the authorized agent, which were made after a configuration change no one logged, and which records of either survive subsequent audit. The governance infrastructure surrounding these agents has not kept pace with their operational reach. Application logs in mutable databases provide no cryptographic guarantee that recorded decisions match decisions actually made. Dashboards observe behavior after it occurs. Policy frameworks declare intent without enforcing it at runtime. The architecture itself is the unit of change. This paper formalizes the architectural category for runtime AI governance: *Attested Governance*. The governance problems facing AI systems are architecturally interdependent and require integrated cryptographic linkage designed in from the start. We present Attested Governance Artifacts (AGA) as one instantiation; peer instantiations are surveyed in Section 3.4.

1.1 Procurement context

Runtime governance for AI systems is becoming embedded in procurement language and post-deployment obligations through multiple parallel tracks, with the US federal pathway in active transition. The NIST AI Risk Management Framework (AI 100-1) is a voluntary technical framework that has historically anchored federal procurement guidance and continues to shape sectoral best practice [National Institute of Standards and Technology, 2023]. Executive Order 14110 (October 2023), which had directed federal agencies to apply AI 100-1, was revoked by Executive Order 14179 of January 23, 2025 [Trump, 2025]; OMB Memoranda M-25-21 and M-25-22 of April 3, 2025 replaced the Biden-era M-24-10 and M-24-18 and now govern federal AI use and procurement, with the M-25-21 use memo not referencing AI 100-1 as particular guidance [Executive Office of the President, Office of Management and Budget, 2025a,b]. The July 2025 White House AI Action Plan directs NIST to revise AI 100-1 [Executive Office of the President, 2025]. NIST has separately issued an Initial Preliminary Draft of IR 8596 (December 2025), a Cybersecurity Framework Profile for AI that extends CSF 2.0 and is intended to complement the AI RMF rather than supersede it [National Institute of Standards and Technology, 2025]. AI 100-1 remains voluntary and is not currently part of binding federal AI procurement requirements during revision. Outside the federal procurement context, runtime governance obligations are being formalized through other tracks. The EU AI Act (Regulation 2024/1689) establishes post-deployment governance obligations for high-risk AI systems through provisions on record-keeping (Article 12), automated logs across the system lifecycle (Article 19), and ongoing in-market monitoring with formal plans (Article 72) [European Parliament and Council of the European Union, 2024]. The NIST Center for AI Standards and Innovation (CAISI) issued a Request for Information on Security Considerations for AI Agents in early 2026 (Docket NIST-2025-0035), soliciting broad input on agent security (authorization, runtime monitoring, evidence generation, governance mechanisms) without preselecting a category

of mechanism [National Institute of Standards and Technology, 2026]. The National Cybersecurity Center of Excellence (NCCoE) issued a concept paper on AI agent identity and authorization in the same period, soliciting industry input on architectural primitives [National Cybersecurity Center of Excellence, 2026]. CAISI engages the standards solicitation question through the broad RFI scope on agent security; NCCoE engages the question of developing architectural primitives that any standardization must build upon. Across these tracks the framing is converging toward requirements that exceed observation alone. Emerging frameworks and contractual structures demand evidence that survives scrutiny by external parties: regulators, auditors, and contractual counterparties. Plain logs, isolated signed records, and post-hoc dashboards each fall short of this requirement.

The proposed category applies to high consequence AI deployment contexts where evidence must survive forensic review. Examples include federal defense procurement, healthcare and life sciences AI under FDA oversight, financial services AI under sectoral regulation, EU AI Act high-risk system deployments, and critical infrastructure AI under CISA and sector guidance. Selection among specific deployments within these contexts is a matter of implementation; the architectural commitment formalized below is common across all of them. We do not claim the category is the right fit for lower consequence AI deployments where partial governance suffices; the architectural argument is scoped to the forensic scrutiny case.

1.2 The structural gap

Established security categories each address one piece of the runtime governance gap. Sigstore and SLSA address build time provenance, proving where software came from [Newman et al., 2022, Open Source Security Foundation (OpenSSF), 2024]. The IETF SCITT working group extends signed claims to supply chain statements [Birkholz et al., 2025]. Open Policy Agent (OPA) addresses admission-time policy evaluation [Open Policy Agent Contributors, 2024]. The IETF Remote ATtestation procedureS (RATS) architecture addresses platform context through TEE-anchored quotes [Birkholz et al., 2023]. Certificate Transparency provides ordering for certificate issuance through append-only signed logs [Laurie et al., 2013]. AI governance dashboards observe agent behavior after the fact. Each category is valuable within its scope. None solves runtime AI governance as one architecture.

The gap is structural. The cryptographic linkage that binds these capabilities has to be designed into the receipt format and trust model from the start. A receipt format that does not carry the structural metadata of the preceding receipt cannot be chained. A trust model with multiple independent roots cannot reduce to one through external wrapping. Section 2 develops this argument through a worked example.

1.3 Thesis

Our central claim is that the governance problems facing critical AI systems (attestation, policy enforcement, evidence generation, continuity verification) are architecturally interdependent in a precise structural sense we define in Section 2.1. We formalize this as *inseparability*, framed as a design imperative: integration of the four pillars requires the cryptographic linkage between them to be designed into the receipt format and trust model from the start. A system originally built

from composed components can be redesigned to commit to the linkage architecturally, at which point it satisfies integration. The commitment is what produces integration.

The paper supports this thesis through four sub-claims. Section 2 establishes the inseparability property formally and demonstrates through ablation that removing any one of the four pillars produces a system that records or enforces but does not govern. Section 3 demonstrates that composing existing single pillar categories produces parallel verification problems with multiple independent trust roots, each requiring its own verifier procedure. Section 4 presents Attested Governance Artifacts (AGA) as one implementation that satisfies the integration property derived in Section 2.4, within a sealed trust boundary that defines the implementation's guarantees. Section 5 anchors the architectural pattern in structural correspondence with regulated practice from critical infrastructure operating under forensic scrutiny: industrial control under ISA/IEC 62443, custody transfer measurement under API MPMS Chapter 21, electronic records integrity under 21 CFR Part 11, and operational technology security under NIST SP 800-82. Our contribution is the formal articulation as a category and the cryptographic implementation at the AI systems layer, applied to the context of autonomous execution that the regulated regimes did not address.

Terminology. We name this category *Attested Governance* to emphasize the architectural property the integrated four-pillar pattern produces: verifiable cryptographic attestation spanning subject identity, policy, enforcement decisions, and continuity. The choice is motivated by three considerations: the property is stable under primitive choice (Section 4.4 demonstrates this with a worked example covering algorithm agility), the term aligns with established procurement and forensic vocabulary used in Section 5, and a property level name accommodates implementations on different cryptographic foundations (signature-based attestation, zero-knowledge proofs, hardware-rooted quotes, or combinations) under the same category.

Non-goals

This paper does not propose a new cryptographic primitive. It uses standard primitives (Ed25519, SHA-256, RFC 8785 canonicalization) and treats algorithm agility as an architectural property [Brennan, 2025, ¶89]. Formal proofs of integration properties are not provided; the argument proceeds through dependency analysis and ablation, not through theorem proving. Standardization is needed; minimum category criteria are proposed in Section 6, though the standardization process itself is outside scope. The threat model follows the scope documented in the source artifacts [Brennan, 2025, ¶8]. AGA is one possible implementation of Attested Governance, not the only one; the category admits multiple architectural realizations (Section 3.4), and we encourage additional implementations.

The integration property formalized here addresses what was enforced under the sealed policy. It does not address whether the policy itself faithfully captures the principal's intent (*intent binding*); whether evidence can be jointly verified across independent trust roots, where neither is subordinate to the other (*governance across independent trust roots*); whether actions that did not occur can be audited (*negative-space attestation*); the determinism, calibration, or behavioral fidelity of the model itself (*non-determinism in model outputs*); or whether policy semantics align with regulatory, ethical, or contractual requirements outside the policy document (*semantic correctness*). These are real problems, upstream or adjacent to the integration property; the architecture does not claim to resolve them.

2 The Inseparability Claim

2.1 Four Pillars Defined

We identify four governance pillars whose combined operation produces runtime governance: attestation, policy enforcement, evidence generation, and continuity verification. Each pillar accepts specific inputs and produces specific outputs. The interaction between pillars is constrained by these input-output relationships, which the rest of this section unpacks.

Attestation. Cryptographic binding of an authorized state to a subject identity, producing an immutable sealed reference. *Input:* subject identity, typically expressed as cryptographic hashes of normalized bytes and canonicalized metadata, together with a content-addressable reference to the governing policy. *Output:* a signed artifact containing the sealed reference. The artifact is immutable once sealed; any modification invalidates its signature. In Attested Governance Artifacts (AGA), this is the Policy Artifact [Brennan, 2025, §C]. In adjacent categories, partial implementations include Sigstore certificates [Newman et al., 2022], which bind at build time only, and RATS quotes [Birkholz et al., 2023], which attest platform state without binding policy.

Policy Enforcement. Runtime mediation of the subject's interactions with protected resources against the sealed reference. *Input:* the signed artifact from attestation, and the subject's runtime state. *Output:* enforcement decisions (permit, deny, transition to a predetermined safe state) and the signing capability that will produce evidence of those decisions. The output couples decisions to a signing key, which makes the decisions cryptographically attributable downstream. In AGA, this is the portal [Brennan, 2025, §F]. In adjacent categories, OPA admission decisions provide policy evaluation without signing, and service mesh policy points enforce without a sealed reference.

Evidence Generation. Production of cryptographically committed records of enforcement decisions, in a format that supports subsequent verification. *Input:* enforcement decisions and the signing capability of the enforcing party. *Output:* signed receipts containing the structural metadata required for ordering and integrity verification. The output format is constrained: it must carry sufficient metadata for downstream continuity verification, or that downstream verification becomes impossible. In AGA, this is the signed receipt with structural metadata, enumerated in Section 4.4. In adjacent categories, application logs are mutable and unsigned; standardized audit trails are signed but lack continuity-supporting metadata.

Continuity Verification. Tamper-evident ordering and integrity of evidence over time, supporting offline verification independent of the producing system. *Input:* signed receipts with structural metadata. *Output:* verifiable history with ordering guarantees, packaged in portable evidence bundles. In AGA, this is the continuity chain with Merkle-checkpointed evidence bundles [Brennan, 2025, §§H, I]; the verification procedure is specified in Section 4.5. In adjacent categories, Certificate Transparency [Laurie et al., 2013] provides ordering without policy binding, and content-addressable storage provides immutability without ordering.

We adopt four pillars as the decomposition for this architecture. Subdivision (for example, separating evidence generation into independent signing and chaining components) produces sub-pillars

trivially coupled within one architectural concern, exposing no external dependency structure. Merging (for example, combining attestation and policy enforcement) collapses distinct trust models: attestation operates at issuing authority scope, and enforcement at portal scope. The four-pillar decomposition is the minimum for this architecture: any further reduction either collapses a pillar into another or trivializes it within the dependency structure analyzed in Section 2.2. We do not claim it is unique across all conceivable runtime governance architectures; alternative splits (separating liveness or revocation as a fifth pillar, or merging attestation with evidence generation under a different trust model) may be defensible under different commitments. The argument that follows operates over the four-pillar decomposition we adopt, and the inseparability claim holds structurally within it.

The four-pillar decomposition also has empirical support from independent implementation. Mazzocchetti's Aegis architecture [Mazzocchetti, 2026], designed independently of AGA, partitions the runtime governance problem into the same four pillars under different vocabulary (Table 1). Two architectures arriving at the same decomposition independently, under different design philosophies and primitive choices, is an empirical observation. The convergence does not prove this decomposition is uniquely correct. It does suggest the four-pillar split tracks structural features of the problem.

Table 1: Four-pillar decomposition across two independent architectures

Pillar	**AGA component**	**Aegis component [Mazzocchetti, 2026]**
Attestation	Policy Artifact (sealed reference binding subject identity and policy)	Immutable Ethics Policy Layer with Genesis Lock fusing hardware identity to policy charter
Policy Enforcement	Portal (mandatory runtime boundary that compares against the sealed reference)	Enforcement Kernel Module as publish gate, with Ethics Verification Agent as compliance watchdog
Evidence Generation	Signed receipt with structural metadata	Immutable Logging Kernel producing hash-chained logs, plus Proof of Conduct per governed decision
Continuity Verification	Append-only continuity chain with Merkle-checkpointed evidence bundles	Hash-chained logs verifiable via Senatus Machina validator quorum

Table 2 presents the formal definitions with explicit inputs/outputs per pillar.

Each definition specifies an input and an output. In three of the four cases, the input of one pillar is the output of another. This structural relationship is the dependency structure analyzed in Section 2.2, and is the basis for the formal inseparability property we define now.

Definition 1 (Inseparability, as a design imperative)**.** Four governance pillars are *inseparable* in an integrated runtime governance system when the cryptographic linkage between them is built into the receipt format and trust model at design time, rather than added through external composition. The property is structural in the design sense: a system either commits to the linkage architecturally (and is integrated, per Section 2.4) or does not (and is composed). Inseparability is not a theorem of impossibility about composition. A system built from previously composed components can be redesigned to commit to the linkage, adopting a shared receipt format with continuity-supporting metadata, a unified trust model rooted in one issuing authority (with delegation as needed), and a single verifier procedure over a single evidence bundle. The redesigned system satisfies integration;

Table 2: Four Governance Pillars Defined

Pillar	Input	Output	Realization in AGA
Attestation	Subject identity (canonicalized metadata hashes) and policy reference	Signed immutable artifact containing sealed reference	Policy Artifact [Brennan, 2025, §C]
Policy Enforcement	Signed artifact; subject runtime state	Enforcement decisions; signing capability coupled to decisions	Portal [Brennan, 2025, §F]
Evidence Generation	Enforcement decisions; portal signing capability	Signed receipts with continuity-supporting structural metadata	Signed receipt with structural metadata (§4.4)
Continuity Verification	Signed receipts containing the structural metadata needed for continuity	Verifiable history with ordering guarantees; portable evidence bundles	Continuity chain with Merkle checkpoints [Brennan, 2025, §§H, I]

the redesign itself is the architectural commitment we are formalizing. The point of the definition is to name what that commitment is: a property of how the receipt format and trust model are constructed at design time. Composition baselines that retrofit linkage through wrapping (adding an outer signature over independently signed inner artifacts) do not produce this property. Section 2.4 specifies the integration property derived from inseparability and identifies the qualifications that exclude pathological cases, including pseudo-integration achieved through wrapping.

The definition has three operational consequences for any system committed to it.

Receipt format dependency. The receipt format containing enforcement decisions must hold the structural metadata required for continuity verification. A format designed without this metadata can only be retrofitted by redesigning it from scratch. At that point all systems producing receipts must adopt the new format. The architectural commitment, then, includes either committing to the format from the start or coordinating migration across systems that were previously independent.

Trust model dependency. All four pillars must verify against a unified trust anchor. Typically this is the issuing authority's public key, possibly via hierarchical delegation; in AGA's two-key trust model, the portal's signing key is itself authorized through the issuing authority's trust chain. Composed systems with N independent trust roots that lack unifying delegation produce N-parallel verification. External wrapping that adds another trust root extends the verification surface.

Verification protocol dependency. A single verifier procedure must check all four pillars from a single evidence bundle. Composed systems requiring N verification protocols and N evidence formats do not satisfy this; a verifier that runs N protocols over N bundles and reconciles results externally is performing composition, not integration.

The claim of this paper is that for any runtime governance system operating within a defined trust boundary, the four pillars must satisfy inseparability as a design commitment. Composition of independent systems through subsequent wrapping does not produce the architectural commitment Section 2.1 names. We defend this claim in Section 2.2 through dependency analysis, in Section 2.3 through ablation, and in Section 2.4 by deriving the integration property with explicit qualifications.

2.2 The Dependency Structure

The four pillars connect through dependencies that form a directed graph. Three of the pillars produce outputs consumed as inputs by another, and one (continuity verification) feeds back informationally to the attestation cycle. We describe the strict dependencies first, followed by the feedback.

Strict dependency 1: Attestation → Policy Enforcement. The portal cannot enforce policy without a sealed reference to compare against. Without an attestation step producing a signed Policy Artifact, the enforcement boundary has nothing to compare the active runtime state against. Enforcement of an unsealed policy admits in-band policy modification, which invalidates the governance property: the policy being enforced becomes whatever was in effect at the moment of enforcement, exactly the state an adversary controlling the policy environment would produce.

Strict dependency 2: Policy Enforcement → Evidence Generation. Evidence generation cannot occur without enforcement decisions to record, nor without a signing capability tied to those decisions. A signing capability detached from enforcement produces signed claims about state (the Sigstore model), without signed records of enforcement decisions. The cryptographic attribution of evidence to a particular enforcement action requires the signing key to be coupled directly to the enforcement boundary that made the decision.

Strict dependency 3: Evidence Generation → Continuity Verification. Continuity verification requires receipts in a format that allows ordering and integrity to be verified. Receipts without structural metadata can be signed individually as independent attestations, yet cannot be chained into verifiable continuity. The receipt format is the constraint. Receipts designed without the metadata needed for continuity make subsequent chaining structurally impossible, regardless of engineering effort applied later.

Informational dependency: Continuity Verification → Attestation (future cycles). The verifiable record produced by continuity verification informs subsequent attestation cycles. Policy authors update their policies based on prior governance evidence, and the new Policy Artifact references the updated version. This dependency is informational rather than strict. Continuity verification does not directly produce the attestation inputs themselves; policy and subject identity are supplied externally. What it does is shape the policy authoring process that produces them.

The dependency structure is a directed graph with three strict edges and one informational feedback edge. The inseparability argument does not require a closed loop. It requires only that each pillar's required inputs cannot be supplied by independent systems with different trust roots, which the three dependencies above establish (Figure 1).

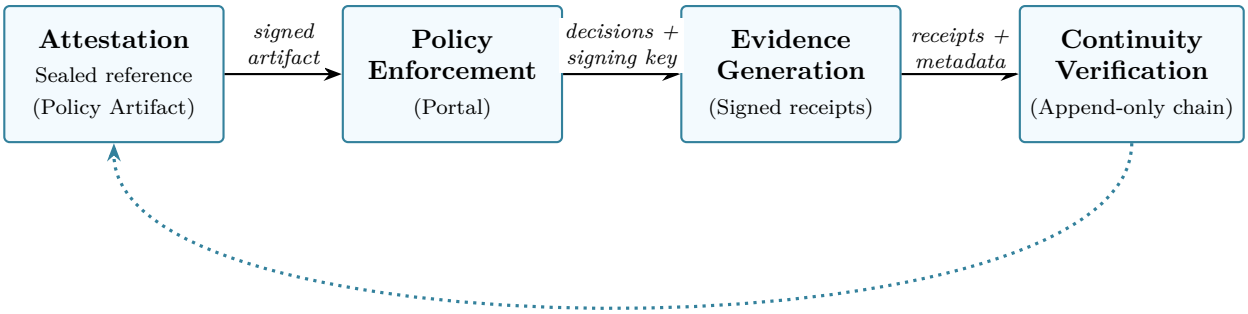

Figure 1: Four-Pillar Dependency Structure. Three strict cryptographic dependencies link Attestation → Policy Enforcement → Evidence Generation → Continuity Verification. One informational feedback link (dotted) carries governance history back to inform later attestation cycles.

Breaking any of the three dependencies leaves remaining components unable to compose into runtime governance:

- Attestation without enforcement produces provenance, meaning signed claims about state without runtime constraint. Sigstore and SLSA sit here. The result is valuable, yet falls short of governance.
- Enforcement without attestation produces admission control: decisions without a sealed reference. OPA without a cryptographically bound policy artifact sits here. The result is valuable, yet falls short of governance.
- Evidence without enforcement produces signed claims about events that no portal mediated. SCITT signed statements sit here. The result is valuable, yet falls short of governance.
- Continuity without evidence produces an ordered, immutable sequence of arbitrary records. Certificate Transparency and content-addressed storage sit here. The result is valuable, yet falls short of governance.

Each of these is a valuable system. Runtime governance requires the three strict dependencies to be satisfied within an architecture where each pillar's required inputs are the outputs of preceding pillars. This is the inseparability property of Section 2.1.

2.3 Ablation Analysis

The inseparability claim can be tested by ablation: removing each pillar in turn and identifying the specific verification breakdown that occurs in the remaining three-pillar system. We present three ablations corresponding to the three strict dependencies.

Ablation A: Three pillars, no continuity verification. A system produces sealed Policy Artifacts, enforces them at a runtime boundary, and generates signed receipts, but does not chain the receipts into a structure that supports continuity verification. Each receipt is signed independently and stored in a database without cryptographic linkage to others.

The system supports attestation, policy enforcement, and evidence generation. The remaining failure points:

Ordering attack. An adversary with database access reorders receipts to construct a history favorable to the adversary. Each receipt independently verifies (signatures are valid) but the ordering cannot be checked.

Omission attack. An adversary deletes the receipt for a denied tool call. That receipt was the record of an unauthorized action being attempted and blocked. The remaining receipts verify independently, and the audit appears clean.

Insertion attack. During a portal compromise window, an adversary fabricates a receipt for an enforcement decision that did not occur. With access to the portal's signing key, the fabricated receipt verifies independently; no chain context exists to detect the inconsistency.

A verifier examining the receipt collection cannot detect any of these attacks because the ordering, completeness, and consistency guarantees that continuity verification provides are absent. The three-pillar system produces signed assertions about decisions, without verifiable history.

Ablation B: Three pillars, no evidence generation. A system performs attestation, runtime enforcement, and maintains a continuity chain, but the chain records only timestamp markers and structural integrity, omitting signed records of enforcement decisions. The chain proves the portal ran but does not record what enforcement decisions were made.

What fails: the verifier can confirm a continuity chain existed and proceeded in order, but cannot determine what enforcement decisions occurred at any point. An adversary who controls the portal during compromise can perform unauthorized actions while producing continuity-valid chain entries that record nothing about those actions. The chain proves the portal was operational but does not prove the portal governed.

Ablation C: Three pillars, no attestation. A system enforces against a policy at a runtime boundary, generates signed receipts, and chains them into verifiable continuity, but the policy is not cryptographically sealed. Enforcement parameters can change at runtime without invalidating any signature.

What fails: the receipts attest to enforcement decisions made against a policy whose integrity cannot be confirmed. An adversary modifies the policy during execution, and subsequent receipts attest to enforcement decisions made under the modified policy. The verifier sees a clean receipt chain that records enforcement against a policy the verifier cannot trust. The system logs and orders, but the policy it logs against is itself in question.

What the ablation demonstrates. Removing any single pillar produces a system that records, or enforces, or asserts without producing verifiable runtime governance. The four pillars resist decomposition. Their integration is what produces the governance property; no split into independent components preserves it.

2.4 Integration as a Structural Property

Inseparability per Section 2.1 specifies a structural property of the relationship between pillars: they cannot be supplied by independent systems with different trust roots. The companion property is integration: what it means for a specific system to satisfy this structural relationship.

Definition 2 (Integration)**.** A runtime governance system is *integrated* if and only if

(a) a single verifier, given a single trust anchor, can verify all four pillars from a single evidence bundle, within the system's defined trust boundary;

(b) the cryptographic linkage between pillars is designed into the receipt format and trust model rather than added through external wrapping; and

(c) the integration property is preserved under a defined set of preserving operations (receipt chain extension, evidence bundle subsetting, re-attestation, and key rotation under continuity chain linkage); operations outside this enumerated set (replacing the portal's signing key without re-attestation, substituting an alternate evidence format, or grafting independently signed artifacts into the chain through external wrapping) are excluded from the system's defined extension rules.

The three qualifications are individually necessary.

(a) Single verifier, single trust anchor, single bundle, within a defined trust boundary. The verifier need not trust the producing system. The trust anchor is the issuing authority's public key. A system that requires multiple distinct trust roots for verification falls into the composed category. The trust boundary names the components within the trusted perimeter; in AGA, the portal is inside the trust boundary, and portal compromise is a named failure case (Section 4.6). Integration is achieved within this stated boundary; it is not a guarantee about external trust.

The two-key trust model in AGA (issuing authority signs Policy Artifacts; portal signs receipts) does not violate this qualification. The portal's signing key is itself authorized through the issuing authority's trust chain; POLICY_ISSUANCE events in the continuity chain record key issuance [Brennan, 2025, §H]. The trust graph has one root with chained delegation, not N parallel trust roots.

(b) Cryptographic linkage by design, not by wrapping. Composition of independent systems cannot retrofit cryptographic linkage into receipt formats that were not designed for it. The chain hash in AGA's receipt format (SHA-256 of each receipt's complete canonical form including signature, referenced from the subsequent receipt's previous receipt hash field; the construction is specified in Section 4.4) is the mechanism that enables continuity verification. A wrapper around an unlinked receipt format cannot produce this property. The wrapper produces an outer signature over a bundle of inner artifacts; this is signed assertion, not integrated verification.

(c) Integration under composition and extension. The integration property must be preserved under defined operations. AGA defines several: receipt chain extension (appending new receipts preserves chain integrity); evidence bundle subsetting (extracting a subset with Merkle

proofs preserves verifiability); Policy Artifact revision via re-attestation (TTL expiration triggers re-attestation, producing a new Policy Artifact that links to the prior one through governance history). Operations that do not preserve integration (replacing the portal's signing key without re-attestation, substituting an alternate evidence format) are excluded from the system's defined extension rules.

The three qualifications exclude pathological cases. A monolithic system that hard-codes verification within itself does not satisfy (a): the verifier cannot be external. A system using standard primitives but achieving integration only through trust in the producing system fails (a)'s trust anchor requirement. A wrapper around composed independent systems fails (b). An ad-hoc integration that breaks under standard operations fails (c).

Response to the monolithicity objection. Single vendor scope and integration are different things. A system built across multiple vendors can be integrated if its receipt formats and trust models are designed for cryptographic linkage; a system from one vendor can fail integration if its receipts are not chained. Vendor scope and architectural integration are independent properties.

Response to the trust boundary objection. Architectural integration and trust model completeness are separate concerns. Section 2 establishes that the four pillars must verify as integrated within a defined trust boundary. Trust model hardening (hardware roots of trust, HSM-backed key storage, multi-party signing, TEE-isolated portals) strengthens that boundary without changing the integration property. The architecture provides integration. Defense-in-depth strengthens the trust model on top of it.

AGA names the portal as a trusted component and enumerates portal compromise as a failure mode (Section 4.6). Systems with implicit trust boundaries carry the same vulnerability, only unnamed; naming it explicitly is the more rigorous choice. A specified failure boundary admits targeted defense such as HSM key storage, TEE isolation, or multi-party signing; an unspecified one admits no such defense.

The architectural argument of Section 2 is independent of the AGA presentation in Section 4. Section 2's reasoning operates over runtime governance systems generally; Section 4 describes one specific architecture. The inseparability property holds for any runtime governance system within a defined trust boundary, regardless of whether AGA is the right implementation of such a system.

2.5 Anticipated Objections

The inseparability and integration claims admit three substantive objections. We steelman each and respond.

Objection A: Decomposition relativity. The inseparability claim holds only for the four-pillar decomposition. A three-pillar decomposition that collapses attestation and evidence generation into a single "signed artifact production" pillar might admit composition: a system that produces signed artifacts at both seal time and decision time could, on this reading, be composed from a single signing primitive applied at two phases. A five-pillar decomposition that adds revocation or liveness as a fifth pillar might exhibit different dependency structure. The inseparability result is

conditional on the decomposition choice, not universal across all possible decompositions of runtime governance.

We acknowledge the decomposition relativity. The four-pillar decomposition is defended as minimal for forensic scrutiny scope on two grounds. First, further reduction collapses distinct trust models: attestation operates at issuing authority scope (who certified the binding), enforcement at portal scope (who applied the policy), evidence at decision scope (what was recorded), and continuity at archival scope (how the history is preserved). Collapsing across these scopes hides the trust model distinctions that the integration property depends on. Second, further expansion conflates concerns within the dependency structure analyzed in Section 2.2: revocation operates over the attestation pillar, liveness over the enforcement pillar. The two-architecture convergence with Aegis (Table 1) is empirical support that two independently developed runtime governance architectures arrive at the same four-component structure; it is not a uniqueness proof. We do not claim the decomposition is unique across all possible runtime governance architectures, and we welcome alternative decompositions that derive comparable inseparability results within their stated scope.

Objection B: Composition with redesign. A composed system whose receipt formats and trust models are extended to support cross-system cryptographic linkage achieves the same property as an integrated system. The distinction between "designed-in" integration and "redesigned-in" integration is rhetorical: in both cases, the receipt formats and trust models support the linkage, and the system satisfies the integration property.

We concede the technical point. A previously composed system that adopts a unified receipt format with structural metadata linkage, hierarchical trust model with a single anchor, and single verifier procedure does satisfy the integration property as defined in Section 2.4. This is precisely what Definition 1 names: the architectural commitment to inseparable design is what produces integration. The redesign *is* integration; the redesigned system is now integrated. The argument of Section 3 addresses a specific claim: that wrapping a composed system in external machinery, without making the underlying receipt format and trust model commitment, produces integration. This claim does not hold. This is exactly the boundary established in Section 2.4 qualification (b): integration is a property of the receipt format and trust model, not of the wrapping layer.

Objection C: Operational sufficiency of composition. In practice, multi-system audit infrastructure already provides adequate forensic evidence. The integration property is a theoretical concern with no operational difference: auditors today verify build time provenance through one system, policy decisions through another, platform attestation through a third, and event logs through a fourth, and arrive at adequate forensic conclusions. Integration as a category is a refinement without operational consequence.

The §3.2 worked example refutes this claim within forensic scrutiny scope. Composition yields parallel verification across four separate systems, each anchored in its own trust root and producing its own evidence format. A verifier can confirm each component on its own. What no verifier can do from those four artifacts alone is answer the integrated question forensic review actually requires: did the right artifact run on the right platform under the right policy, and do the recorded decisions match the actions taken? Disjoint records leave that question structurally unanswerable (Figure 2). The gap is not in the per-component checks, which composition passes; it is in the cross-component binding that the integrated question requires. For low consequence audits, separate per-component verification may suffice. For external review where the question must be answered to a courtroom,

a regulator, or a post-incident investigator, the architectural gap becomes operational. This is the scope qualification of Section 5.1: the claim is correspondence with regulated industry practice under forensic scrutiny, not universality across all AI governance.

3 Existing Approaches and the Composition Gap

3.1 Survey by Pillar Coverage

We examine adjacent security categories and identify which of the four pillars each addresses. Table 3 organizes the comparison; per-category paragraphs identify what each category does well and where its pillar coverage stops.

Table 3: Adjacent Categories vs. Pillar Coverage

Category	**Attestation**	**Policy enforcement**	**Evidence generation**	**Continuity verification**
Sigstore / SLSA / SCITT	Build time only	—	Build time signed claims	—
OPA / policy-as-code	—	Admission-time only	Logged, often unsigned	—
RATS / TEE attestation	Platform level only	—	—	—
Certificate Transparency	—	—	Signed log entries	Ordering primitive only
AI governance dashboards	—	—	Mutable logs (weak)	—
Attested Governance (this paper)	**Yes**	**Yes**	**Yes**	**Yes**

Sigstore / SLSA / SCITT [Newman et al., 2022, Open Source Security Foundation (OpenSSF), 2024, Birkholz et al., 2025]. These categories address build time provenance. The first signs build artifacts and produces certificates verifiable against a designated root CA. SLSA defines build-integrity levels. SCITT extends signed claims to supply chain statements. All three produce attestation pillar coverage at build time but contribute nothing to runtime policy enforcement or to verifiable continuity of runtime decisions. Sigstore's Rekor transparency log provides Merkle inclusion proofs for issued certificates, and the IETF SCITT working group is extending signed statements toward runtime contexts; the trajectory acknowledges the limitation without closing it structurally. Even with cross-system receipt format extensions and runtime applicability, the underlying trust roots remain independent: runtime extension is protocol evolution, not architectural integration. Extending the build time category to satisfy the four-pillar integration property requires architectural redesign at the format and trust model layer, not feature addition alone (Section 3.3 develops why).

OPA / policy-as-code [Open Policy Agent Contributors, 2024]. OPA evaluates policies against decision inputs at admission time. Decisions are logged but typically unsigned; even when signed, the signing key is separate from any attestation root. Pillar coverage: policy enforcement at admission time. The category does not address attestation, signed evidence binding to attested state, or continuity verification.

RATS / TEE attestation [Birkholz et al., 2023]. Platform level attestation through trusted execution environments produces signed quotes verifiable against TEE vendor PKI. Pillar coverage: attestation at platform level. Runtime policy enforcement for the workload, signed evidence of workload decisions, and continuity binding platform attestation to workload behavior over time fall outside the category.

Certificate Transparency [Laurie et al., 2013]. CT operates an append-only signed log with Merkle proofs, providing tamper-evident ordering for certificate issuance events. Pillar coverage: continuity verification through ordering, with signed log entries as evidence. Attestation of workload state, runtime policy enforcement, and binding of log entries to enforcement decisions are outside the category. CT is honestly framed as a different category from partial coverage governance; it shares one primitive (signed append-only log entries) with the continuity pillar, and AGA inherits that primitive at the receipt chain level.

AI governance dashboards. Observability tooling for AI agent behavior. Logs are mutable; signing is incidental; cryptographic linkage between logs and enforcement decisions is absent. Pillar coverage: weak evidence pillar (mutable logs); attestation, enforcement, and continuity are all absent. Dashboards observe behavior after the fact rather than governing it at runtime.

This comparison shows that adjacent categories are valuable and mature within their own scope. None alone satisfies the inseparability property of Section 2.1. The natural question is whether composing them produces what no single category provides. Section 3.2 develops this through a worked example. (C2PA addresses media artifact integrity and is omitted from this comparison.)

3.2 Worked Example: Composing Four Independent Solutions

A federally regulated organization deploys an autonomous AI agent under three audit obligations: agent decisions must be cryptographically attributable, agent behavior must be verifiable against authorized policy, and audit evidence must survive adversarial review by external parties. To meet these, the organization assembles four established categories:

Provenance layer (Sigstore). Agent binary artifacts are signed at build time [Newman et al., 2022]. The verifier confirms the artifact came from an authorized signer via certificate chain validation against the build time root CA.

Admission layer (OPA). Before each agent action against a protected resource, an OPA policy evaluates the action against authored rules. OPA produces a decision (permit or deny). Decisions are written to a log and, depending on deployment, may be signed by an OPA operator key.

Platform attestation layer (RATS). The TEE in which the agent executes produces a RATS attestation quote. The quote is verifiable against the TEE vendor's attestation authority via vendor PKI.

Audit layer (CT-style log). Decisions are written to an append-only signed log with Merkle proofs. Ordering and integrity are verifiable against the log operator's signing key.

The auditor's question is whether all four components (binary, platform, policy, decisions) are cryptographically linked into a single verifiable chain.

The auditor's actual verification flow:

1. Validate the Sigstore certificate. Conclude: *the artifact was built by an authorized signer.*
2. Validate the RATS quote. Conclude: *a TEE configuration matching the quote existed at some point.*
3. Walk the OPA decision logs. Conclude: *OPA evaluated certain inputs and produced certain decisions, assuming logs were not modified.*
4. Verify CT-style log Merkle proofs. Conclude: *certain entries existed in the log at certain checkpoint times.*

Each verification succeeds within its own scope. The auditor holds attestations against four independent trust roots: build time root CA, TEE vendor PKI, OPA operator key, log operator key. Each conclusion stands alone.

The gaps:

- Sigstore proves artifact provenance but does not bind to the platform that ran it. The provenance certificate and the RATS quote share no cryptographic linkage.
- The RATS attestation confirms platform state at one point in time but does not bind to the OPA decisions made on that platform. The attestation could be valid while policy evaluation occurred on a different, potentially compromised instance.
- OPA decisions are logged but not bound to RATS-attested platform state. Even when signed, the OPA signing key is independent of the RATS attestation authority and the build time provenance root.
- The CT-style log preserves ordering of entries but does not bind entries to the decisions OPA actually made. The log could contain fabricated entries matching genuine decisions, or omit genuine entries from the record.

The auditor cannot answer the original question. The answer requires cross-component cryptographic linkage that is not present in any of the four categories and that no external wrapper can supply without becoming a fifth trust root.

Composition produces four parallel verification problems with separate trust roots and disjoint evidence formats. Verification is parallel, not integrated. The verifier confirms each component

independently; the integrated question of whether the right artifact ran on the right platform under the right policy with the recorded decisions remains structurally unanswerable from the separate artifacts (Figure 2).

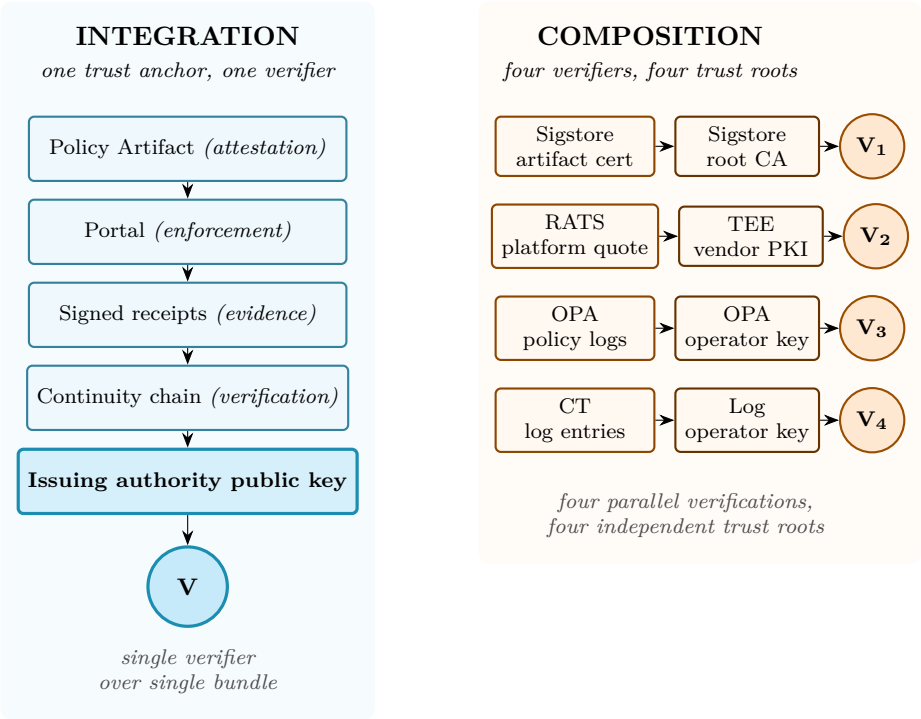

Figure 2: Composition vs. Integration. On the left, a single verifier checks all four pillars from a single evidence bundle against one trust anchor (integration). On the right, four independent verifications anchor against four independent trust roots (composition).

3.3 Why Composition Cannot Close the Gap

The gap is structural. Five reasons follow.

Receipt formats are not designed for cross-system chaining. OPA decision logs, Sigstore certificates, and RATS quotes share no fields linking them to one another or to policy. Retrofitting cross-system references modifies every system's receipt format per deployment, and the next deployment must reproduce the wrapping.

Trust models are independent by design. Each category verifies against its own trust root: Sigstore against a root CA, RATS against TEE vendor PKI, OPA against an operator key, CT against a log operator key. Wrapping these together does not reduce to one trust root; it adds a fifth, the wrapper signer.

Unifying the trust root does not solve this either. The strongest version of the composition defense issues Sigstore certificates, RATS quotes, OPA decisions, and CT log entries under a shared

X.509/PKIX root via cross-certification, establishing one trust root through delegation. This is the canonical pattern of one verifier validating a chain under one anchor, and it does eliminate the wrapper signer objection above. It does not, however, establish the integration property. Sigstore certificates, RATS attestation evidence, OPA decision logs, and CT signed entries share no fields binding a certificate's subject to the policy decision made on that subject's attested platform, nor to the CT entry recording issuance. Cross-component binding requires *format commitment* (the receipt format dependency identified in Section 2), not trust root unification. PKIX delivers the latter without the former; the canonical construction that wraps signed assertions in an outer certificate chain satisfies neither inseparability (Definition 1) nor integration (Definition 2).

Adversarial scenarios cross boundaries. An adversary who compromises the OPA operator can fabricate decisions that match valid Sigstore certificates on attested RATS platforms with valid CT log entries. Each component verifies independently while the composition stays unsound. Detection requires cross-component cryptographic binding that no individual category provides; the threats that operate across composition boundaries fall outside every individual threat model.

Standardization does not solve this. A common receipt format extension with hash references across categories makes the formats linkable, but the trust roots remain independent. Each reference is signed by its asserting system's own key, so the auditor verifies multiple independent signatures over assertions about other systems, not over state integrated into the asserting system's own verification.

A system designed from the start with attestation, policy enforcement, evidence generation, and continuity as coupled outputs of one architecture (with one trust anchor, one evidence bundle format, and a single verification procedure) produces what composition cannot. Section 4 presents AGA as one such system; Section 3.4 surveys peer instantiations.

3.4 Peer Instantiations in the Emerging Category

Mazzocchetti's Aegis architecture [Mazzocchetti, 2026], released March 2026, is an independently developed runtime governance architecture in the same emerging category. AGA's architecture was disclosed in U.S. Patent Application 19/433,835 (filed December 28, 2025), predating the March 2026 Aegis release, and the two works appear to have been developed independently. Like AGA, Aegis commits to integrated cryptographic linkage across the four pillars by design, not through composition of independent systems. The integration property defined in Definition 2 and demonstrated for AGA in Section 4.6 is reflected in Aegis's architectural commitments through different primitive choices. The existence of a second independently developed full coverage implementation supports the architectural convergence claim of Section 2.5.

The broader March 2026 literature includes three additional architectures in adjacent or partial coverage territory. Yuan et al. [2026] (a distinct architecture sharing the name "AEGIS" with Mazzocchetti's work; we refer to this work as Yuan et al.'s AEGIS throughout) presents a pre-execution firewall and audit layer for AI agents, applying a three-stage pipeline of content extraction, risk scanning, and policy validation at the tool execution boundary, with decisions recorded in an Ed25519-signed, SHA-256-chained audit trail. The work covers policy enforcement and evidence generation pillars under our four-pillar definition; attestation of subject identity to authorized configuration is not addressed, and continuity verification is implemented through hash chaining

without the Merkle-checkpointed evidence bundles that support offline verification. Kaptein et al. [2026] formalize runtime governance through "policies on paths," contracts attached to specific execution trajectories that balance task completion against legal, reputational, and data breach costs. The work addresses policy enforcement at the path level with substantial formal development of the cost-balancing trade-off, but does not address cryptographic attestation, signed evidence generation, or continuity verification as architectural concerns. Zhou [2026] formalizes capability context separation and derives three Agent Governance Requirements (capability integrity, behavioral verifiability, interaction auditability) along with the Chain Verifiability Theorem and the Bounded Divergence Theorem. This formalization is the closest of the three to the four-pillar decomposition: capability integrity corresponds to attestation, behavioral verifiability corresponds to enforcement evidence under reproducibility verification, and interaction auditability corresponds to continuity verification. The policy enforcement boundary as an architectural component is missing.

Within the decomposition adopted in this paper, AGA and Mazzocchetti's Aegis remain the only architectures that address all four pillars, and both build the cryptographic linkage into the receipt format and trust model from system genesis. The other three works each cover a subset: Yuan et al.'s AEGIS contributes pre-execution enforcement with a signed audit trail; Kaptein et al. develop contracts on execution paths that trade task completion against legal and reputational cost; Zhou formalizes capability and context as separable security objects under reproducibility verification. Each contributes substantive work within the territory it addresses. As the category formalizes and the surrounding vocabulary stabilizes, further architectures are likely to emerge, and the analysis above provides the framework against which they can be classified.

4 AGA as Instantiation

4.1 Architectural Overview

Attested Governance Artifacts (AGA) is one implementation of the Attested Governance category; Mazzocchetti's Aegis [Mazzocchetti, 2026], introduced in Section 3.4, is another. AGA implements the four pillars formalized in Section 2.1 (attestation, policy enforcement, evidence generation, and continuity verification) within a single architecture, three operational phases (Seal, Enforce, Prove; Figure 4), and one trust anchor. The integration property of Section 2.4 is satisfied by construction (Section 4.6); the trust boundary, which includes the portal as a trusted component, is named and bounded.

Four defining architectural commitments:

Two-process boundary. The two-process boundary (Figure 3) is the architectural primitive that makes evidence cryptographically meaningful. Evidence signed by the subject of governance is not governance evidence.

Mandatory mediation. Every interaction between the subject and a protected resource transits the portal boundary. There is no parallel path that bypasses the portal. The default state is denial: in the absence of a valid Policy Artifact, no execution proceeds.

Single trust anchor. All cryptographic verification anchors at the issuing authority's public key. The portal's signing key is itself authorized through the issuing authority's trust chain; POLICY_ISSUANCE events in the continuity chain record this delegation [Brennan, 2025, §H]. The trust graph has one root.

Single evidence bundle. Verification material (Policy Artifact, signed receipts, Merkle inclusion proofs, public key for verification) is packaged into a single portable bundle. Offline verification requires only the bundle and standard cryptographic primitives (Ed25519, SHA-256).

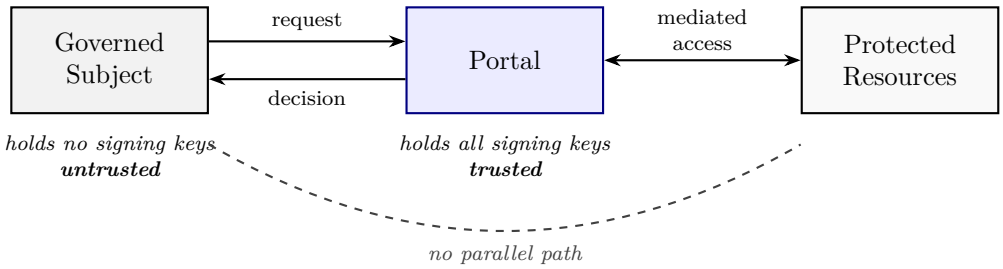

Figure 3: Two-Process Boundary. The portal mediates all interactions between the governed subject and protected resources. The subject holds no signing keys; the portal holds all signing keys. There is no parallel path bypassing the portal.

The architecture is algorithm-agile: receipt and artifact metadata specify the signature algorithm, enabling migration from Ed25519 to ML-DSA or SLH-DSA without architectural change [Brennan, 2025, ¶89]. The integration property is independent of the chosen signature algorithm; Section 4.4 demonstrates this through a worked migration example.

Table 4 presents the mapping from components to pillars, with patent specification references.

Table 4: AGA Components Mapped to Pillars and Patent Claims

Pillar	**AGA component**	**Function**	**Spec. ref.**
Attestation	Policy Artifact	Sealed reference binding subject identity, policy reference, sealed hash, enforcement parameters	§C, Claim 1
Policy enforcement	Portal	Verification at startup; continuous measurement; enforcement on drift detection	§F, Claims 10, 11
Evidence generation	Signed receipt with structural metadata	Cryptographically committed records of enforcement decisions and measurements	§4.4
Continuity verification	Continuity chain + Merkle checkpoint + evidence bundle	Append-only chain linked by structural metadata; Merkle root anchored to immutable storage; portable offline-verifiable bundle	§§H, I, J, Claims 3, 9

The next four subsections describe each pillar's implementation. Section 4.6 demonstrates that the four implementations, taken together, satisfy the integration property of Section 2.4.

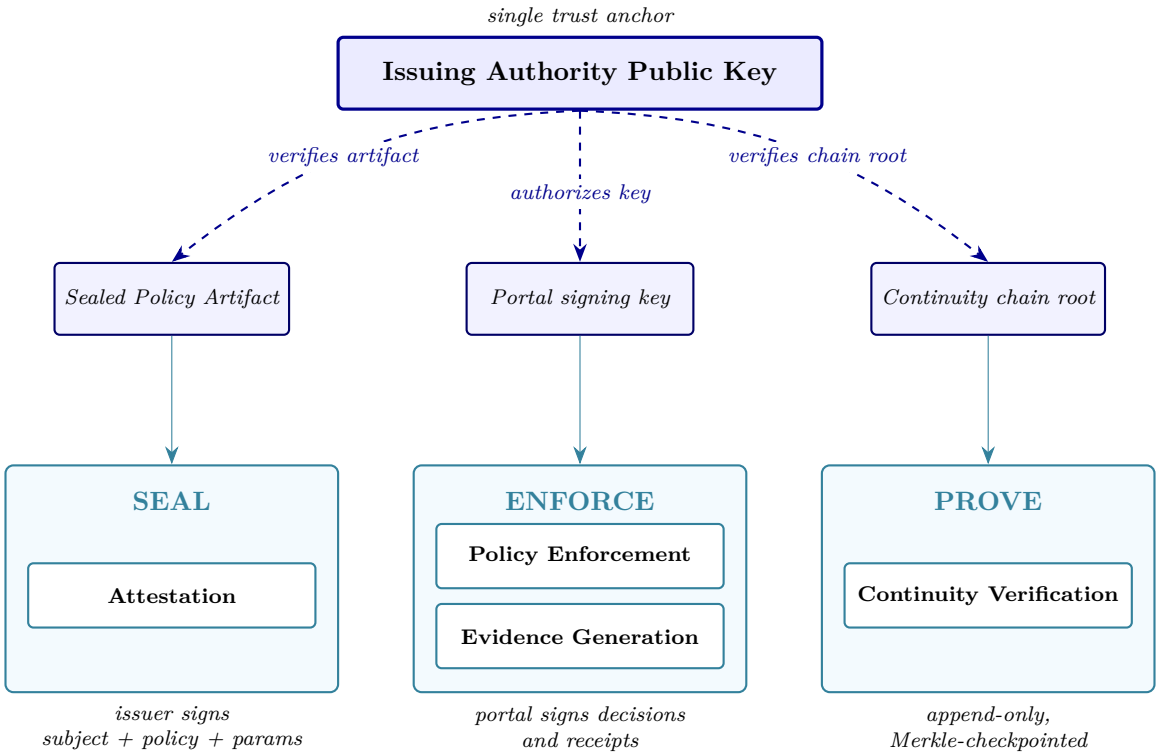

Figure 4: Three-Phase Architecture. The four pillars distribute across the Seal/Enforce/Prove operational phases. A single trust anchor (issuing authority public key) verifies the artifact, the portal's delegated signing capability, and the continuity chain.

4.2 Policy Artifact (Attestation)

The Policy Artifact is the sealed reference: a signed object that binds subject identity, policy reference, and enforcement parameters into an immutable structure [Brennan, 2025, §C].

The artifact binds four classes of information. *Subject identity* is established through two cryptographic hashes: one over the normalized bytes of the subject (executable image, container digest, model weights, or other primary content) and one over the canonicalized metadata (subject identifier, version, deployment context) [Brennan, 2025, §B, Claim 1(b)]. *Policy reference* is a content-addressable hash pointing to the governing policy document. The policy is referenced rather than embedded so that policy revisions can be tracked independently while the artifact remains immutable. *Sealed hash value* is the immutable reference against which the subject's runtime state will be compared by the portal; mismatch on initial measurement results in fail-closed execution blocking. *Enforcement parameters* include the measurement cadence, the time-to-live (TTL) governing artifact validity, and the enforcement triggers specifying what action the portal takes on drift detection.

These four classes of information are bound by a cryptographic signature. In the reference embodiment, the signature is Ed25519 over the canonical serialization of the artifact [Brennan, 2025, ¶82–87]. The serialization uses RFC 8785 JSON canonicalization to eliminate ambiguity in field

ordering and whitespace [Rundgren et al., 2020]. The sealing structure uses length-prefixed encoding for the composite hash inputs, preventing field boundary ambiguity that could allow distinct inputs to produce identical hashes [Brennan, 2025, §§B, D].

The artifact is immutable post-seal: any modification to the subject identity, the policy reference, the sealed hash, or the enforcement parameters invalidates the signature. The portal verifies the signature against a pinned issuer public key before parsing [Brennan, 2025, Claim 10]; verification failure blocks execution.

Re-attestation: TTL expiration triggers re-attestation [Brennan, 2025, Claim 6]. The new Policy Artifact references the prior policy version through the continuity chain; the policy revision is recorded as a POLICY_ISSUANCE event linking past and current artifacts. This mechanism allows policy evolution without compromising the integrity of decisions made under prior policy versions. Each decision is bound to the policy version in effect at the time of enforcement.

4.3 Portal (Policy Enforcement)

The portal is the mandatory runtime boundary [Brennan, 2025, §F]. It implements three coupled responsibilities: verification at startup, continuous measurement during execution, and enforcement on drift detection.

Startup verification. Before execution begins, the portal performs three checks. It verifies the Policy Artifact signature against the pinned issuer public key [Brennan, 2025, Claim 10]. It confirms the current time falls within the artifact's effective period; the issued, effective, and expiration times bound the artifact's validity. It computes initial runtime state measurements and compares them to the sealed hash. Any check failing blocks execution.

Continuous measurement. During execution, the portal computes runtime state hashes at the measurement cadence specified in the artifact. The patent names ten measurement embodiments [Brennan, 2025, §E, ¶42–53]: executable image digest, loaded module digests, container image digest, configuration manifest digest, software bill of materials digest, TEE quote, memory region sampling, control flow measurement, file system state digest, and network configuration digest. Different embodiments may run at different cadences within the same governed subject; an executable image digest might be checked on every measurement cycle while configuration manifests are checked every minute.

Enforcement on drift. On drift (runtime measurement diverging from sealed reference) the portal executes the predetermined enforcement action specified in the sealed artifact. The patent enumerates action types: termination, quarantine (phantom execution), network isolation, key revocation, token invalidation, actuator disconnection, and safe-state transition [Brennan, 2025, ¶57]. The action is sealed at attestation time, not improvised at runtime; the response to compromise is pre-committed by the issuing authority. For contexts where termination is unacceptable, such as industrial control during active operations or autonomous vehicles mid-mission, the sealed artifact specifies safe-state transition instead of hard termination.

Phantom execution is a specific enforcement variant [Brennan, 2025, §G, Claim 11]. When drift is detected and the policy specifies quarantine instead of termination, the portal transitions the subject to a sandboxed execution environment. Connections to protected resources (actuators, network endpoints, data stores) are severed. Inputs continue to be delivered to the sandboxed subject; the subject continues operating, believing it functions normally. Outputs are captured as forensic receipts and not delivered. This converts a security incident into intelligence-gathering: the attacker's intent and methods are recorded without permitting actual damage.

Recursive delegation addresses multi-agent governance scenarios [Brennan, 2026b, §4]. When a primary agent tasks a secondary agent, the portal issues a derived artifact to the secondary with TTL no longer than the parent's remaining TTL and scope no broader than the parent's authorized resources. The secondary's genesis event is cryptographically linked to the parent's continuity chain. Scope can diminish through delegation but cannot expand. A sub-agent cannot be granted permissions the parent did not hold.

Concurrent enforcement. Within a single portal instance, the portal runs each subject's measure, compare, enforce, and sign steps as one indivisible cycle, never interleaving cycles. Every signed receipt therefore reflects a governance decision that ran to completion under the sealed Policy Artifact, never a partial one. When the portal is distributed across several instances (sharing one continuity chain, or governing overlapping subject populations), that guarantee no longer holds on its own, and preserving each subject's ordering across instances takes an external coordination mechanism such as distributed locking or a consensus protocol. Those mechanisms are a deployment concern rather than part of the single portal baseline this paper analyzes, and they operate under the same trust boundary discipline described in Section 4.6.

4.4 Signed Receipts and Policy-Gated Disclosure (Evidence Generation)

The portal generates a signed receipt for every enforcement decision and for every measurement cycle. The receipt format specifies the structural metadata required for chain integrity verification together with the decision content of each governance event. Receipt fields include the algorithm identifier (`Ed25519-SHA256-JCS` in the reference embodiment), event identifier, sequence number, protocol version, timestamp, previous leaf hash for chain linkage, signature, and the decision-specific payload (tool name, arguments hash, policy reference as SHA-256 of the canonical policy document, reason code, and method identifier in the MCP-proxy embodiment). Verifiers must reject unknown algorithm identifiers. Forward compatibility through schema versioning does not extend to security-critical fields.

The receipt format provides one chain commitment that supports three derived verifier capabilities. The chain commitment is structural: each receipt's signature covers the full receipt content including a `previous_receipt_hash` field; structural metadata is extracted for Merkle tree leaf hashing while payload remains within the signed receipt scope; and Merkle roots are computed over leaf hashes for periodic checkpointing. From this single chain commitment, three verifier capabilities derive, each addressing a distinct verification problem the others do not solve. The three capabilities are not independent integrity guarantees. They are views into the same chain commitment, calibrated to different verifier types and verification problems.

The chain commitment is formally stated as

$$R_n \;=\; \mathsf{Sign}(K_{\text{portal}},\; \mathsf{Payload}_n \;\|\; \mathsf{H}(R_{n-1})) \tag{1}$$

where Sign denotes the portal's signing operation under key K_{portal}, $\mathsf{Payload}_n$ is the canonical receipt content for cycle n (the receipt fields enumerated above), H is SHA-256, $\|$ is RFC 8785 JCS canonical concatenation, and R_{n-1} is the prior receipt's complete signed form. This chained construction ensures that any modification to R_{n-1} (payload, structural metadata, or signature) invalidates R_n, guaranteeing the ordering and integrity required by the continuity verification pillar (Definition 1; verification procedure in Section 4.5).

Local ordering verification (via chain hash). The `previous_receipt_hash` field in each receipt references the SHA-256 of the prior receipt's complete canonical form including signature. The problem is ordering: an auditor holding a sequence of receipts must confirm that none was inserted, deleted, or reordered between adjacent positions. The auditor inspects each receipt's `previous_receipt_hash` against the actual hash of the prior receipt; chain hash mismatch indicates compromise. Modifying any receipt invalidates the chain hash for that receipt, which propagates forward to invalidate all subsequent chain hashes.

Privacy-preserving global integrity verification (via structural metadata hash). The leaf hash for Merkle tree inclusion is computed over structural metadata only: schema version, protocol version, event type, event identifier, sequence number, timestamp, previous leaf hash [Brennan, 2025, Claim 3(c)]. The payload is deliberately excluded from this leaf hash computation. Privacy is the constraint: a third party auditor (regulator, court-appointed examiner, contractual counterparty) must verify chain integrity *without* access to tool call arguments or sensitive content within receipts. Structural metadata hashing enables this: chain integrity is verifiable from leaf hashes alone, without disclosure of payload. The payload remains protected by the same chain commitment, since the event signature is computed over the complete event including payload and a content-addressable payload hash is stored in event metadata; payload tampering remains detectable to parties with payload access while privacy is preserved at the chain level.

Scalable inclusion proofs and tamper-evidence against local storage compromise (via Merkle root, optionally anchored). Leaf hashes are periodically batched into a Merkle tree; the Merkle root is computed and may be anchored to an immutable storage network. This one is about scale and resilience: an auditor verifying inclusion of a single receipt should not need to process the entire chain, and chain integrity should survive compromise of the local database storing receipts. The Merkle root provides logarithmic cost inclusion proofs for individual receipts; anchoring to immutable storage prevents history rewriting even when local storage is compromised, because the anchored root contradicts any rewritten history.

The three capabilities are different projections of the same underlying structure: local ordering verification onto adjacent-receipt linkage, privacy-preserving integrity verification onto leaf hashing over structural metadata alone for Merkle construction, and tamper-evident anchoring onto the Merkle-root commitment. Each capability addresses a verifier type the others do not serve, while the verification stack rests on one chain commitment, not three independent guarantees layered redundantly.

Algorithm agility worked example. A deployment running Ed25519+SHA-256 today (algorithm identifier `Ed25519-SHA256-JCS`) migrates to ML-DSA-65 ahead of post-quantum signature requirements. The migration is metadata level, not architectural. The issuing authority issues a new Policy Artifact under algorithm identifier `MLDSA65-SHA256-JCS`, signed with the ML-DSA-65 private key bound to a new issuer public key (with the new public key authorized via a POLICY_ISSUANCE event in the continuity chain signed under the still-valid Ed25519 key). The portal upgrades to recognize both algorithm identifiers concurrently during the transition window. Existing receipts under the Ed25519 chain remain verifiable under their original algorithm; new receipts are issued under the ML-DSA-65 chain. The verifier, encountering a bundle with mixed algorithm identifiers, applies the appropriate verification primitive per receipt as indicated by the algorithm identifier field. After the transition window completes, the Ed25519 chain is closed by an ANCHOR_BATCH event; subsequent receipts are exclusively ML-DSA-65. No section of the architecture changes. Only the metadata field value and the verifier's primitive table need updating.

The migration mechanism above shares structural features with generic algorithm negotiation patterns in TLS, JWT, COSE, and SCITT, where algorithm identifiers in protocol metadata permit primitive substitution. What distinguishes AGA's algorithm agility from generic protocol level negotiation are three AGA-specific architectural elements preserved across the algorithm boundary:

Two-process boundary preservation. The boundary AGA enforces between the trusted portal and the untrusted governed agent (Section 4.1) is unchanged across the migration. The portal's signing role under the new algorithm is structurally identical to its role under the old algorithm; the governed agent holds no keys before, during, or after migration. Generic protocol level algorithm negotiation does not commit to this two-process boundary as architectural structure; it commits only to algorithm identifier semantics. The boundary is what makes algorithm substitution meaningful as governance migration rather than as a protocol upgrade.

Continuity chain linkage across the algorithm boundary. The POLICY_ISSUANCE event signed under the still-valid Ed25519 key, which authorizes the new ML-DSA-65 issuer public key, links the Ed25519 chain to the ML-DSA-65 chain through a single chain commitment that the verifier walks during evidence bundle verification. One continuity chain spans the algorithm boundary, with the cryptographic delegation explicit. Generic algorithm negotiation patterns produce per-session algorithm choice; AGA's continuity chain linkage produces per-deployment migration history that is itself verifiable as part of the four-pillar bundle.

Single verifier procedure across mixed algorithm receipts. The verifier procedure of Section 4.5, a five-step deterministic computation over the evidence bundle, operates identically across Ed25519 receipts, ML-DSA-65 receipts, and the POLICY_ISSUANCE event that bridges them. Primitive selection is driven by the per-receipt algorithm identifier. The bundle remains a single bundle; the verifier remains a single procedure; the trust anchor remains the issuing authority's public key (with the migration's delegation event documenting key rotation inside the chain). Generic protocol level algorithm negotiation typically requires per-protocol verifier configuration; AGA's algorithm agility keeps the bundle, verifier, and anchor singular across the migration boundary.

The migration window as bounded transitional state. During key rotation, the system operates in a transitional dual algorithm state, from issuance of the new key under the old chain until expiry or revocation of the old key. The trust anchor across both algorithms is the issuing authority, not any particular signing primitive: the new key is authorized by a POLICY_ISSUANCE event signed under the old chain, with the rotation event integrated into the continuity chain rather than

external to it. The migration is therefore an exception window with bounded dual verification. Receipts within the window verify under their per-receipt algorithm identifier, while the chain has one trust root spanning the transition. Outside rotation periods, single algorithm verification is the steady state.

Policy-gated disclosure addresses the sharing of governed evidence across organizational boundaries under policy constraint [Brennan, 2025, §K, Claim 2]. AGA includes a privacy-preserving disclosure mechanism: when a sensitive claim is requested and policy denies disclosure of the requested claim, the system traverses an ordered substitute list and discloses the first permitted substitute of lower sensitivity. A signed substitution receipt records the original claim identifier, substitute identifier, policy version under which substitution was performed, and reason code. Before disclosure, the system verifies that the disclosed set of claims does not enable inference of denied claims [Brennan, 2025, Claim 14]. Three disclosure modes are supported: PROOF_ONLY (boolean attestation of the claim), REVEAL_MIN (minimized or bucketed disclosure of the claim value), and REVEAL_FULL (complete claim value disclosure). This mechanism distinguishes AGA from runtime governance systems that do not address structured disclosure under policy constraint.

4.5 Continuity Chain and Evidence Bundles (Continuity Verification)

The continuity chain is append-only, with events linked by structural metadata hashes [Brennan, 2025, §H, ¶65–70]. The genesis event contains the protocol version identifier, taxonomy version identifier, a root fingerprint derived from the chain's cryptographic key pair, and a specification hash binding the chain to a specific rule set [Brennan, 2025, Claim 17].

Checkpoint anchoring periodically commits chain state to immutable external storage [Brennan, 2025, §I, ¶71–73, Claim 3(d–f)]. The system selects a batch of continuity events, constructs a Merkle tree with the leaf hashes of the batched events as leaves, computes the Merkle root, and submits the root to an immutable append-only storage network through a pluggable anchor provider interface [Brennan, 2025, Claim 18]. Candidate networks contemplated in the patent specification include Arweave, Ethereum, and other content-addressed distributed ledgers. The anchor provider returns a transaction identifier; the system appends an ANCHOR_BATCH event to the continuity chain recording the checkpoint reference, batch range, Merkle root, anchor network identifier, and transaction identifier. This mechanism prevents history rewriting even if the local database is compromised. Modifying historical events invalidates leaf hashes, which propagates to invalidate the Merkle root, which contradicts the anchored transaction.

Tiered verification levels are specified in the patent application [Brennan, 2025, §L], calibrating rigor to deployment consequence. The *Bronze* tier relies on device attestation for low consequence agents and development. *Silver* moves to server attestation with client-side key pinning, suited to moderate consequence enterprise deployments. *Gold* anchors evidence checkpoints on a blockchain with Merkle inclusion proofs, the tier for critical infrastructure, regulatory compliance, and defense. Receipt metadata records which tier produced a receipt, so relying parties can set confidence thresholds accordingly. The scheme is the runtime governance analog of FIPS 140-2/3 security levels for cryptographic modules. The reference implementation supports the Gold-equivalent path through Merkle-checkpointed evidence bundles (Section 4.5); the explicit tier

metadata fields and the Bronze and Silver paths are specified in the architecture but not yet built into the open-source reference codebase.

Evidence bundles package verification material into portable artifacts [Brennan, 2025, §J, Claim 9]. A bundle contains the Policy Artifact, relevant signed receipts, Merkle inclusion proofs for each receipt, the checkpoint reference (transaction identifier and Merkle root), and the public key for signature verification. Bundles are offline-verifiable: all verification steps except optional anchor validation can be performed without network connectivity to the producing system.

The verification procedure operates as a deterministic five-step computation. The verifier first rejects any receipt with an unknown algorithm identifier, refusing forward compatibility for security-critical algorithm fields. It then performs signature verification on each receipt against the public key (Ed25519 in the reference embodiment, or ML-DSA-65 or other algorithm as indicated by the receipt's algorithm identifier field). Chain linkage is verified by computing the chain hash for each receipt and confirming the next receipt's `previous_receipt_hash` matches the prior receipt's hash per Equation 1. Merkle inclusion proofs are walked from each receipt's leaf to the stated Merkle root. Finally, all computed roots are confirmed to match the stated root in the checkpoint reference. The result is binary: the bundle is valid or it is not. The procedure is stateless, depending only on bundle contents and the public key.

4.6 Integration by Construction, with Explicit Trust Boundary Qualification

The four-pillar architecture produces integration by construction. Each pillar's output is structured to be the next pillar's input:

- The Policy Artifact's signature, anchored at the issuing authority's public key, establishes the trust anchor.
- The portal verifies that signature, parses the artifact, and operates within its enforcement parameters. The portal's signing key is itself authorized through the issuing authority's trust chain; POLICY_ISSUANCE events in the continuity chain record key issuance [Brennan, 2025, §H].
- The portal generates signed receipts; receipts inherit the policy reference and chain to prior receipts via `previous_receipt_hash`.
- Receipts carry structural metadata enabling continuity verification independently of payload disclosure.
- The continuity chain produces evidence bundles containing the original Policy Artifact, the receipts, Merkle inclusion proofs, and the public key for verification.

A single verifier, given the issuing authority's public key, verifies all four pillars from the evidence bundle. The integration property defined in Section 2.4 is satisfied.

Trust boundary. This integration property holds within AGA's stated trust boundary. The boundary includes the portal as a trusted component. A compromised portal signs receipts for

actions not actually evaluated or for actions that were denied. These pass all five verification steps because they carry valid signatures from a trusted key.

Temporal bounding of compromise: architectural mechanism, operational magnitude. The architecture does not prevent portal compromise. It provides a *mechanism* for temporally bounding compromise but does not itself specify the *magnitude* of the bound. The mechanism comprises three architectural elements. First, a compromised portal cannot issue new Policy Artifacts (those require the issuing authority's signature, which the portal does not hold) and cannot modify existing Policy Artifacts (signature verification at startup fails). Second, the issuing authority retains the capability to rotate the portal's signing key and to issue a revocation POLICY_ISSUANCE event in the continuity chain, after which all receipts signed under the compromised key past the revocation timestamp fail verification. Third, anchored Merkle checkpoints with timestamps distinguish pre-compromise from post-compromise receipts in the chain, enabling forensic separation of trustworthy from untrustworthy intervals.

The *magnitude* of the resulting compromise window (how long an adversary in possession of the portal's signing key can produce forged receipts that still verify, before the architecture's bounding mechanism takes effect) is determined by external operational properties the architecture does not specify. These include: detection latency (how quickly compromise is detected, which depends on monitoring infrastructure and threat detection capability external to AGA itself); the operational time required for issuing authority key rotation (which depends on key rotation processes, HSM coordination, and multi-party signing policies); and the propagation latency for revocation POLICY_ISSUANCE events to reach verifiers consulting the continuity chain. A deployment with high-confidence intrusion detection, automated key rotation under multi-party signing, and rapid revocation propagation has a fundamentally different residual risk profile from a deployment with no detection capability and manual key rotation. Yet both deployments instantiate the same AGA architecture. The architecture is the mechanism; deployment practices set the magnitude.

AGA's contribution to the portal compromise problem is *architectural*: a sealed mechanism for bounding compromise, distinguishing pre- from post-compromise receipts, and propagating revocation through the continuity chain. AGA's contribution does not include a *magnitude bound*; magnitude is supplied by the deployment's external operational properties. A claim that AGA "bounds the compromise window" is accurate only if read as "provides the mechanism that operational practices use to bound the window."

Issuing authority compromise: the harder failure mode. The portal compromise case enumerated above is the more recoverable failure mode; issuing authority compromise is the harder case and is outside the baseline architecture's defenses. If the issuing authority's signing key is compromised, an adversary can mint Policy Artifacts binding arbitrary subject identities to arbitrary policies, and the trust anchor against which the four pillars verify is itself untrusted. The architecture does not defend against this case in its baseline form. Three mechanisms outside the baseline scope address it. *Threshold issuance* distributes the issuing authority's signing capability across k-of-n parties such that no single compromised party can issue artifacts; the trust anchor becomes the threshold quorum rather than a single issuing key. *HSM-backed key storage with multi-party signing policies* makes single party compromise insufficient for artifact issuance even when an attacker reaches one party's environment. *External attestation of issuing authority integrity at signing time*, by a separate attestation authority over the issuing authority's runtime state, allows

verifiers to validate that artifacts were signed by an issuing authority in a known-good state. Each is a deployment practice extension to the baseline; each strengthens the trust model rather than the architecture. The architecture provides the structure under which these mechanisms compose; the mechanisms themselves are outside the scope of this paper.

What would lift the trust boundary qualification. Integration within the stated trust boundary is weaker than trustless integration. The qualification (that integration is architectural within a stated boundary including the portal) can be lifted by mechanisms not included in the baseline architecture. Three directions are compatible with the design and each strengthens the assurance profile. *Threshold portal signing* distributes the portal's signing capability across k-of-n portal instances such that no single compromised portal can forge receipts; the trust boundary then includes the threshold quorum rather than a single portal. *External attestation of portal integrity* at receipt generation time, by a separate attestation authority signing portal state assertions or by remote attestation over RATS-style evidence [Birkholz et al., 2023], allows the verifier to validate portal integrity alongside the portal receipt. *Hardware-rooted attestation* embeds portal execution within a TEE with measured boot, with attestation quotes carried in receipts; the trust boundary then includes the TEE measurement root rather than the portal's software state. Each is work outside the scope of this paper. Single trust root verification within a stated boundary differs from N-trust-root verification across independent systems; the baseline can be strengthened along any of these directions without changing the integration property as defined in Section 2.4.

Integration as defined in Section 2.4 is *architectural*, while trust model completeness is *operational*. The deployment must include layered defenses that bound the portal's compromise probability and remediation latency. The two are complementary: architectural integration is what AGA provides; trust model completeness is what defense-in-depth and (for higher-assurance profiles) the mechanisms named above add.

4.7 Empirical Baseline

The architecture has been implemented as a reference codebase[1]: approximately 56,000 lines of Go with ∼59% aggregate test coverage in the current reference codebase (HEAD), exposing a 14-command CLI, supporting 12 policy profiles spanning AI agent, defense, and regulated industry deployment patterns. Cross-language verification vectors verify byte-identical receipt encoding between Go, Python, and TypeScript implementations across 45 reference test cases spanning leaf hash, seal, Merkle, canonicalization, timestamp normalization, signature, and policy decision categories. The reference implementation is distributed under the `aga-governance` package on PyPI and the `@attested-intelligence/aga-mcp-server` package on npm. This section reports measured performance on the reference implementation; the numbers establish a baseline for the architecture's overhead profile rather than optimization targets, and production deployments with HSM-backed signing or hardware-accelerated cryptographic primitives will have different absolute numbers.

[1] A public repository accompanying this paper is available at https://github.com/attestedintelligence/aga-mcp-server. It distributes the reference MCP server implementation, the deployment scenarios, an independent offline verifier, and the cross-language verification vectors that reproduce the receipt encoding determinism results reported in this section. The performance figures were measured with the Go benchmark harness in the reference codebase, under the methodology and per-operation results recorded in the repository's `BENCHMARKS.md`; the full production-grade implementation remains proprietary at this time.

Methodology. All measurements were taken on commodity hardware (AMD Ryzen 5 3550H, Windows 11, Go 1.26.1) using standard Go benchmarking tooling. Each per-operation timing reported in Tables 5 and 6 is the per-iteration average over a minimum of 1,000 benchmark iterations. The end-to-end measurement cycle number aggregates signature generation, hash computation, and chain append operations under the reference implementation's default configuration.

End-to-end measurement cycle. A complete signed measurement cycle completes in approximately 4.94 ms on the reference hardware.[2] The cycle comprises Ed25519 signature generation over the receipt payload, SHA-256 hashing of the canonicalized receipt content, and an append and chain operation against the continuity chain. Receipt generation and chain append are constant-time per cycle and re-measure at approximately 40 μs at the current implementation HEAD; the remainder of the 4.94 ms is the subject measurement computation, which varies by embodiment, with binary integrity digest scaling with executable size and configuration manifest digest scaling with manifest complexity.

Signature primitive benchmarks. Table 5 reports per-operation timing for the three supported signature modes. The hybrid composite construction concatenates length-prefixed Ed25519 and ML-DSA-65 signatures; hybrid signing produces both signatures, and verification requires both to validate, with no downgrade fallback. Alignment with the IETF LAMPS composite signature draft (`draft-ietf-lamps-pq-composite-sigs`) is open work flagged among the post-quantum primitive directions in Section 6.5. Existing Ed25519-only artifacts remain verifiable under the algorithm-agile receipt schema.

Table 5: Signature primitive benchmarks: AMD Ryzen 5 3550H, Windows 11, Go 1.26.1, cloudflare/circl v1.6.0[4]

Operation	Algorithm	Mean (ns/op)	CI	B/op	allocs/op
Sign	Ed25519	33,850	±4%	64	1
Sign	ML-DSA-65	202,800	±62%	450	3
Sign	Hybrid (Ed25519 + ML-DSA-65)	264,300	±113%	3,906	4
Verify	Ed25519	78,430	±2%	0	0
Verify	ML-DSA-65	40,660	±2%	450	3
Verify	Hybrid (Ed25519 + ML-DSA-65)	121,100	±7%	450	3

Storage overhead. Table 6 reports public key and signature sizes for each signature mode. The ML-DSA-65 signature is 51.7× the size of an Ed25519 signature, and the hybrid composite signature is 52.8×, essentially ML-DSA-65 plus an Ed25519 increment. For deployments where receipt chain storage is dominated by signature size (high cadence measurement under bandwidth-constrained

[2]The 4.94 ms figure is the aga-k8s v0.9.1 reference measurement (Q1 2026), taken when the cycle was instrumented end-to-end including the subject measurement step. At the current implementation HEAD the cryptographic receipt generation path re-measures at 40.07 μs (±3%, benchstat, 2026-05-24); the full cycle is not re-instrumented at HEAD because the subject measurement step is deployment-specific (the choice of file or process measurer, and policy complexity) and sits outside the cryptographic hot path. The signature primitive figures in Table 5 and the hybrid cadence derivation in the *Cadence feasibility* paragraph below were re-measured against implementation HEAD on 2026-05-24; raw output at `aga-k8s/_artifacts/BENCHSTAT_2026-05-24.txt`.

transport), this is a deployment trade-off the algorithm-agile schema exposes rather than a fixed architectural cost.

Table 6: Storage overhead by signature mode (raw bytes)

Component	Ed25519	ML-DSA-65	Hybrid	Ratio to Ed25519
Public key	32 B	1,952 B	1,992 B	62.3×
Signature	64 B	3,309 B	3,381 B	52.8×

Cadence feasibility under post-quantum deployment. Hybrid sign plus verify totals approximately 385 μs per governance decision (Sign: 264 μs mean, ±113% CI; Verify: 121 μs mean, ±7% CI). At the fastest supported measurement cadence (100 ms), the hybrid signing and verification overhead consumes approximately 0.4% of the cadence window (worst-case 95% CI bound: 0.7%); at slower cadences the percentage falls proportionally. The reference implementation accommodates 100 ms, 200 ms, 250 ms, 500 ms, and 1,000 ms cadences within the hybrid overhead budget, with the specific cadence selected per deployment profile (e.g., `scada-100ms`, `drone-250ms`). Post-quantum migration does not require cadence relaxation in the reference configuration.

Comparison to the reported Aegis baseline. AGA's 4.94 ms measurement cycle and the 238 ms median proof-verification latency reported for Aegis under controlled internal trials [Mazzocchetti, 2026] are not directly comparable: they measure different operations (signature-based generation versus proof-based verification), on different hardware, under different methodology. We report both to situate AGA's per-decision generation cost rather than to rank the two architectures; the difference does not reflect implementation effort.

Scope of these measurements. The benchmarks reported here characterize the reference implementation under the documented methodology. They do not constitute independent third party benchmarking. Independent replication, benchmarking under varied hardware, and cross-implementation empirical comparison are open work, flagged among the open research directions in Section 6.5.

5 The Architectural Pattern in Regulated Industries

5.1 Anchoring in Regulated Practice

The architectural pattern formalized in Section 2 (sealed reference, mandatory mediation, evidence generation with structural metadata, continuity verification) has structural precedent. Industries operating under high consequence forensic scrutiny have converged on the same architectural commitments within the forensic scrutiny subset of their regulatory regimes, regimes that have evolved through enforcement actions, audit failures, and commercial litigation. This section examines three such regimes (industrial control under ISA/IEC 62443-3-3 and NIST SP 800-82 Rev. 3, custody transfer measurement under API MPMS Chapter 21.1, and electronic records and signatures under 21 CFR Part 11) and demonstrates that the same architectural commitments operate in each within that subset of their requirements.

The scope of the correspondence requires care. The regulated regimes surveyed below did not operate against autonomous tool execution at runtime; 21 CFR Part 11 governs the binding of human electronic signatures to records, API MPMS 21.1 governs electronic measurement system integrity for custody transfer, and ISA/IEC 62443 governs industrial control system behavior with primarily human or supervised-process actors. None of these regimes confronted the runtime governance of autonomous AI agents making policy-relevant decisions, which is the contemporary problem this paper addresses. The structural correspondence we claim is at the level of architectural commitment (how evidence is produced, integrity-protected, and verified under external audit), not at the level of operational context. The regulated industry anchoring demonstrates that the same architectural commitments have been independently adopted in three regimes facing the same audit pressure, which is evidence that the architecture is what such pressure produces rather than novel research speculation. The cross-regime correspondence is structural, not operational.

ISA/IEC 62443-3-3 contains seven foundational requirements (FRs) and 51 system requirements (SRs) across them; we select the SRs that map to the four-pillar pattern (SR 2.1, SR 3.4, SR 3.9, SR 6.1, SR 6.2) because they instantiate the forensic scrutiny architectural commitments. API MPMS Chapter 21.1 contains operational provisions on metering, calibration, and reporting that go beyond the audit framework, and 21 CFR Part 11 contains procedural requirements on training, electronic signature uniqueness, and certification that go beyond the architectural integrity provisions we cite. The pattern claim is over the subset of each regime that addresses how evidence is produced, integrity-protected, and verified under forensic review, not over the full regulatory scope.

The argument is empirical within that scope, not metaphorical. Each claim below is anchored to a specific provision of a specific standard. Provision text for 21 CFR Part 11 and ISA/IEC 62443-3-3 citations was directly verified against the published source documents; API MPMS Chapter 21.1 and NIST SP 800-82 provisions were verified through pattern characterization across aligned secondary sources where direct provision text was not available. No claim in this section depends on inference beyond what the cited provision supports.

The cross-domain anchoring grounds the architectural necessity claim of Section 2 in structural correspondence with operationally validated practice within forensic scrutiny scope, with the operational context gap to autonomous AI execution made explicit above. Each cross-domain claim below cites a specific provision; the pattern is observed structurally across three independent domains under three independent regulatory regimes.

Table 7 (presented at the end of Section 5.5) summarizes the convergent four-pillar pattern across industrial control, custody transfer, electronic records, and AI substrate.

5.2 Electronic Records and Signatures: 21 CFR Part 11

Title 21 of the Code of Federal Regulations, Part 11 establishes FDA criteria under which electronic records and electronic signatures are accepted as trustworthy, reliable, and generally equivalent to paper records and handwritten signatures [U.S. Food and Drug Administration, 1997]. Part 11 is operative for closed systems (§11.10) and open systems (§11.30), with §11.30 incorporating all §11.10 controls plus additional measures.

Section 11.10(a) requires "validation of systems to ensure accuracy, reliability, consistent intended performance, and the ability to discern invalid or altered records" [U.S. Food and Drug Administration, 1997, §11.10(a)]. The attestation pillar appears here at the electronic records layer: the

system is validated to operate as specified, and is required to detect when records have been altered. The architectural commitment is that integrity is structural. The system itself can discern altered records, not merely an external audit.

Section 11.10(e) requires "use of secure, computer-generated, time-stamped audit trails to independently record the date and time of operator entries and actions that create, modify, or delete electronic records. Record changes shall not obscure previously recorded information. Such audit trail documentation shall be retained for a period at least as long as that required for the subject electronic records" [U.S. Food and Drug Administration, 1997, §11.10(e)]. This provision specifies four architectural commitments operating together: audit trails are computer-generated (not user-entered, eliminating one class of falsification vector), they are time-stamped (establishing ordering), they independently record actions on records (the evidence generation pillar is mandatory and operates separately from the records themselves), and previously recorded information cannot be obscured by subsequent changes (the append-only integrity property at the regulatory layer).

Section 11.10(f) requires "use of operational system checks to enforce permitted sequencing of steps and events" [U.S. Food and Drug Administration, 1997, §11.10(f)]. This is policy enforcement at the workflow layer: the system enforces the workflow sequence rather than relying on user discipline. Section 11.10(g) requires "use of authority checks to ensure that only authorized individuals can use the system, electronically sign a record, access the operation or computer system input or output device, alter a record, or perform the operation at hand" [U.S. Food and Drug Administration, 1997, §11.10(g)]. Authority checks operate at every action that could affect a record: system use, signature, device access, record alteration, operation execution. This is mandatory mediation at the regulated records layer.

Section 11.70 (Signature/record linking) is structurally critical to the four-pillar pattern at this layer: "electronic signatures and handwritten signatures executed to electronic records shall be linked to their respective electronic records to ensure that the signatures cannot be excised, copied, or otherwise transferred to falsify an electronic record by ordinary means" [U.S. Food and Drug Administration, 1997, §11.70]. The signature-to-record binding is required to operate cryptographically (or by an equivalent mechanism that defeats ordinary means), preventing the cross-record forgery attack: a signature cannot be removed from one record and applied to another. The structural equivalent at the AI governance layer is the cryptographic chain hash linking a signed receipt to its sealed Policy Artifact and to the prior receipt; the signature is bound to the record and its position in the chain, not transferable across records.

5.3 Industrial Control: ISA/IEC 62443-3-3 and NIST SP 800-82 Rev. 3

ISA/IEC 62443-3-3 specifies system security requirements for industrial automation and control systems (IACS). The standard organizes requirements into seven foundational requirements (FRs), each containing system requirements (SRs) calibrated to capability security levels (SL-C 1 through 4). Four SRs map directly to the four-pillar pattern.

SR 2.1 (Authorization Enforcement) requires that "the system should be able to enforce authorization on all users, roles, and parameters" [International Society of Automation and International Electrotechnical Commission, 2013, SR 2.1]. The policy enforcement pillar appears here at the industrial control layer: enforcement is system level, not advisory, operating across the full scope of users, roles, and parameters within the IACS.

SR 6.1 (Audit Log Accessibility) requires that "the system should only grant authorized users read-only access to audit logs and not be able to modify the logs" [International Society of Automation and International Electrotechnical Commission, 2013, SR 6.1]. The structural property of an append-only evidence chain appears here: logs are produced by the system and accessible to authorized parties, yet not modifiable after generation, including by the parties who can read them. SR 3.9 (Protection of Audit Information) reinforces the integrity property by requiring active protection of audit information against unauthorized access and modification [International Society of Automation and International Electrotechnical Commission, 2013, SR 3.9].

SR 6.2 (Continuous Monitoring) requires "ongoing monitoring protocols to ensure constant awareness and support risk decisions" through "detecting and recording security events" [International Society of Automation and International Electrotechnical Commission, 2013, SR 6.2]. This is continuity verification at industrial control layer: monitoring is continuous rather than periodic, events are recorded as they occur, and the recorded events support risk decisions made by authorized parties.

SR 3.4 (Software and Information Integrity) requires the IACS to provide the capability to "detect, record, report and protect against unauthorized changes to software and information at rest" [International Society of Automation and International Electrotechnical Commission, 2013, SR 3.4]. The sealed reference pillar appears here at the industrial control layer: integrity protection operates on software and information at rest against unauthorized modification, and the system itself must detect such changes. This detection capability, the IACS discerning unauthorized changes, is the structural equivalent of AGA's portal detecting drift between runtime state and the sealed reference. SR 3.4 also requires reporting of detected changes, corresponding to AGA's signed receipt generation on drift detection. The report is integrity-protected by structural property, generated by the IACS itself rather than reviewed externally.

The four SRs cited above (SR 2.1 for policy enforcement, SR 3.4 for sealed reference and drift detection, SR 6.1 and SR 3.9 for evidence generation integrity, SR 6.2 for continuity verification) together instantiate the four-pillar pattern at the industrial control layer within the forensic scrutiny subset of 62443-3-3.

NIST SP 800-82 Rev. 3, "Guide to Operational Technology (OT) Security" (September 2023), extends NIST SP 800-53 Rev. 5 security controls to operational technology environments through low-, moderate-, and high-impact OT overlays [Stouffer et al., 2023]. The document aligns with ISA/IEC 62443 and the NIST Cybersecurity Framework, providing tailored guidance on Audit and Accountability, Configuration Management, System and Information Integrity, and System and Communications Protection control families for OT systems.

The Configuration Management control family addresses detection of unauthorized changes to OT system configuration, the structural equivalent of drift detection against a sealed reference. The Audit and Accountability control family addresses the evidence generation pillar through OT-tailored logging requirements that account for ICS-specific reliability and safety constraints. The continuous monitoring guidance addresses the continuity verification pillar through OT-appropriate cadence and protocol recommendations.

Two architectural commitments emerge consistently across both ISA/IEC 62443-3-3 and NIST SP 800-82 Rev. 3. First, authorization enforcement is system level and continuous, not advisory. The IACS or OT system itself enforces, with no separate governance layer overlaid afterward. Second, audit information is integrity-protected by structural property, not by external audit. The

system is designed so that audit records cannot be modified after generation, including by system administrators.

5.4 Custody Transfer Measurement: API MPMS Chapter 21.1

The American Petroleum Institute's Manual of Petroleum Measurement Standards Chapter 21.1 governs flow measurement using electronic metering systems for custody transfer of gaseous phase hydrocarbon [American Petroleum Institute, 2013].

Custody transfer is a commercial transaction in which evidence must survive external audit by parties whose financial interests may diverge: pipeline operators, gas producers, gas purchasers, regulators, and auditors. The standard provides "the minimum reporting and change management requirements of the various intelligent components required for accurate and auditable measurement" [American Petroleum Institute, 2013, Scope]. The auditable measurement framework is not an optional addition; it is the operational basis for custody transfer commerce.

The standard's security provisions "restrict access to data and protect algorithms from unauthorized changes" [American Petroleum Institute, 2013]. This is mandatory mediation at the measurement layer: the algorithms producing measurement results are protected against modification, and the data produced is access-restricted. The integrity property is binding, not advisory. Algorithm modification without authorization invalidates custody transfer measurement under the standard's audit framework.

The standard's auditability framework operates structurally: the EFM (Electronic Flow Measurement) system is designed so that configuration changes, calibration events, and measurement reports are independently retrievable, time-correlated, and verifiable by external auditors.

The relevant architectural observation: custody transfer is exactly the kind of high consequence forensic scrutiny transaction that AI procurement is becoming. The auditor in a custody transfer dispute asks the same structural question as the auditor in an AI procurement audit. Was this measurement (or this AI decision) produced by an authorized system operating under sealed configuration, with continuous monitoring producing integrity-protected records that survive external audit by parties whose interests may diverge?

5.5 The Pattern Across Three Domains

Three independent regulatory regimes, each governing different subject matter under different threat models and different commercial pressures, converge on the same four architectural commitments within the forensic scrutiny subset of their requirements. Table 7 summarizes the convergent pattern.

Sealed reference with drift detection capability: 21 CFR §11.10(a) through validation and the requirement to discern altered records; ISA/IEC 62443-3-3 SR 3.4 (Software and Information Integrity), requiring the IACS to detect, record, report, and protect against unauthorized changes to software and information at rest; ISA/IEC 62443-3-3 configuration management requirements complementing SR 3.4; API MPMS Chapter 21.1 through algorithm protection provisions and the auditable measurement framework.

Table 7: Architectural Pattern Across Critical Infrastructure

Architectural commitment	**Electronic records** (21 CFR Part 11)	**Industrial control** (ISA/IEC 62443, SP 800-82)	**Custody transfer** (API MPMS Ch. 21.1)	**AI substrate** (AGA, this paper)
Sealed reference with drift detection capability	§11.10(a) (validation; ability to discern altered records)	SR 3.4 (software and information integrity); configuration management requirements	Algorithm protection provisions; auditable measurement framework	Policy Artifact with sealed hash (§4.2)
Mandatory mediation through authorization enforcement	§11.10(d), §11.10(f), §11.10(g) (access, sequencing, authority checks)	SR 2.1 (authorization enforcement)	Restricted-access provisions for data and algorithms	Portal as mandatory runtime boundary (§4.3)
Evidence generation with structural integrity	§11.10(e) (secure, computer-generated, time-stamped audit trails)	SR 3.9 (protection of audit information)	Change management reporting requirements	Signed receipts with structural metadata (§4.4)
Continuity verification preserving ordering and integrity	§11.10(e) (retention); §11.70 (signature–record linking)	SR 6.1 (read-only audit log access); SR 6.2 (continuous monitoring); NIST SP 800-82 Rev. 3 continuous monitoring guidance	Time-correlated, externally verifiable records	Continuity chain with Merkle checkpoints; evidence bundles (§4.5)

Mandatory mediation through authorization enforcement: 21 CFR §11.10(d), §11.10(f), §11.10(g) through limiting access, operational system checks, and authority checks; ISA/IEC 62443-3-3 SR 2.1; API MPMS Chapter 21.1 through restricted-access provisions.

Evidence generation with structural integrity: 21 CFR §11.10(e) through secure, computer-generated, time-stamped audit trails that cannot obscure previously recorded information; ISA/IEC 62443-3-3 SR 3.9 (Protection of audit information); API MPMS Chapter 21.1 through change management reporting requirements.

Continuity verification preserving ordering and integrity: 21 CFR §11.10(e) through retention requirements that match the records themselves, and §11.70 through signature-to-record binding that cannot be falsified by ordinary means; ISA/IEC 62443-3-3 SR 6.1 (read-only audit log access) and SR 6.2 (continuous monitoring); NIST SP 800-82 Rev. 3 continuous monitoring guidance.

The pattern is structural. Each regime arrived at the same architectural commitments independently, through enforcement experience and audit pressure within its own domain. The regimes did not coordinate; the standards reference each other in places but were not designed to share a common architectural foundation. They converged because the architectural pattern is what high consequence forensic scrutiny requirements produce when the regulated parties commit to making evidence cryptographically meaningful at infrastructure layer.

This convergence is the empirical backing for the architectural necessity claim of Section 2. The four-pillar pattern is not something this paper invented; regulated infrastructure arrived at it on its own, in the settings where evidence has to survive forensic scrutiny, and proved it over decades of enforcement and audit. Those industries governed human operators and supervised processes long

before autonomous AI, so the correspondence is architectural: it lies in how evidence is produced and verified, not in the operational setting. What this paper adds is the articulation: naming the pattern as a category, carrying it to the AI substrate those regimes never had to govern, and implementing it cryptographically at the scale AI systems now operate.

6 Future Implications

6.1 From Application Layer to Foundational Layer

In Q1 2026, the major commercial AI providers are beginning to publish service level agreements with more explicit decision auditability and retention commitments, restructuring their commercial offerings around per-inference accountability, not per-API-call delivery. The shift is small in isolation but typical of a transition in commercial framing: AI providers are beginning to behave like utility providers, billing per use, committing to performance and audit guarantees, calibrating tiers to deployment consequence. AI systems are currently still treated as applications: models accessed through APIs, agents deployed as services, governance added through dashboards alongside the application stack. The governance is bolted on, not designed in. The directional pressure is toward integration at the foundational layer.

This transition is in progress rather than complete. The signals are directional but not dispositive, and the architectural argument that follows is conditional on the transition continuing in its observed direction. If the transition reverses or stalls, the foundational layer argument loses force; the application level architectural argument grounded in Section 5's structural correspondence with regulated infrastructure stands independently within forensic scrutiny scope.

The directional signals are visible in three domains. *Regulatory*: federal procurement RFIs are formalizing what must be provable about AI agent decisions; NIST CAISI and NCCoE are explicit on this [National Institute of Standards and Technology, 2026, National Cybersecurity Center of Excellence, 2026]. Sector regulators are formalizing post-deployment monitoring requirements; EU AI Act Articles 12, 19, and 72, along with emerging FDA guidance on AI/ML in medical devices, point in this direction [European Parliament and Council of the European Union, 2024]. *Commercial*: per-token and per-inference pricing structures are widespread in AI services, and SLA-style commitments on AI behavior (refusal rates, decision auditability, retention guarantees) are beginning to appear in enterprise contracts. These are early signals of utility-style provisioning rather than completed transition. *Contractual*: AI providers are restructuring offerings with service tiers calibrated to deployment consequence, again an early signal.

The implication conditional on this transition continuing: the governance layer beneath utility-grade AI must operate with the architectural rigor of foundational layers beneath other utilities. Electric grid governance is engineered into the grid itself: sealed setpoints, continuous measurement, automatic enforcement, regulated audit, all operating as integrated infrastructure (Section 5). Telecom routing governance is built into the network. Water utility metering is built into the meter. AI governance at the foundational layer, when it exists, will follow the same pattern.

Section 2 formalizes the integrated four-pillar pattern this paper proposes for the foundational layer.

6.2 Why Integration Becomes Non-Optional at the Foundational Layer

At application layer, partial governance is tolerable. A dashboard that misses some violations is operationally annoying but not catastrophic. Audit failures are localized to specific applications; the consequences scale with application scope.

At the foundational layer, partial governance is failure. Failures cascade. A single governance gap at this layer produces application level failures across every service that depends on it. Federal procurement will not buy foundational services that fail to verify under audit; the procurement chain depends on infrastructure verifiability. Regulated industries cannot deploy underlying services whose evidence does not survive forensic scrutiny.

The four-pillar integration is therefore the engineering minimum for foundational systems. The structural correspondence established in Section 5 is direct: the critical infrastructure regimes examined there converged on integrated runtime governance within their forensic scrutiny scope because partial governance was tried and failed under enforcement pressure. Foundational AI is likely to follow a similar trajectory under analogous audit pressure. The strategic question is whether the field adopts integrated runtime governance proactively, informed by the structural correspondence with regulated practice, or only after foundational scale incidents force adoption. Proactive adoption is the case this paper makes.

6.3 Mapping to the NIST AI Risk Management Framework

NIST AI 100-1 organizes AI risk management around four functions (Govern, Map, Measure, Manage), each containing categories (19 in total: six Govern, five Map, four Measure, four Manage) and 72 subcategories beneath them [National Institute of Standards and Technology, 2023]. We map the four-pillar architecture formalized in Section 2 against specific categories at each function. The framework remains voluntary and is currently in revision per the July 2025 White House AI Action Plan; we map against AI 100-1 v1.0 (January 2023), and the mapping will need recalibration once the revised version is published.

The *Govern* function includes categories on policies, processes, and accountability mechanisms (Govern 1 through Govern 6). The Policy Artifact (Section 4.2) is a cryptographically sealed implementation of governance policy: the policy is content-addressable, the binding is signed, and the audit trail of policy issuance is captured in POLICY_ISSUANCE events on the continuity chain. Accountability is structural: every decision is bound to the specific policy version under which it was made, making policy evolution auditable, not ambiguous. This maps most directly onto Govern 1 (policies, processes, procedures, and practices across the organization) and Govern 4 (accountability structures).

The *Map* function includes categories on AI system context, scope, and capabilities (Map 1 through Map 5). The subject identifier in the Policy Artifact (Section 4.2), comprising hashes of normalized bytes and canonicalized metadata, formalizes what the governed AI system is, at what version, in what deployment configuration. This satisfies Map 1 (context is established and understood) and Map 4 (risks and benefits are mapped for the system) at the AI system identification layer with cryptographic precision, not documentation.

The *Measure* function includes categories on quantitative monitoring and tracking (Measure 1 through Measure 4). The portal's continuous measurement responsibility (Section 4.3), with ten

measurement embodiments operating at policy-specified cadences, satisfies Measure 2 (AI systems are evaluated for trustworthy characteristics) and Measure 3 (mechanisms for tracking risks over time are in place) at the runtime layer with the integrity property AI 100-1 calls for: the measurements are generated by an independent enforcement boundary, signed at generation time, and bound to the policy version against which they are evaluated.

The *Manage* function includes categories on response and recovery (Manage 1 through Manage 4). The portal's enforcement responsibility (Section 4.3), with pre-committed enforcement actions and the phantom execution forensic capture variant, satisfies Manage 2 (strategies to maximize AI benefits and minimize negative impacts are planned and applied) and Manage 4 (risk treatments are documented and monitored regularly) at the response execution layer with the architectural rigor that bolt-on response cannot provide. The response is sealed at attestation time and executed automatically on drift detection, not negotiated after the fact during incident response.

The mapping is not exhaustive: AI 100-1 contains 72 subcategories and the four-pillar architecture does not address all of them. In particular, AI 100-1's coverage of cultural and procedural risk management, workforce competency, and stakeholder engagement exceeds what any architectural pattern can satisfy. The architectural functions of AI 100-1, the ones that demand technical implementation rather than organizational practice, are satisfied by the four-pillar pattern when integrated. Compositions of partial coverage solutions satisfy these functions only with the structural gaps Section 3 identifies. When AI 100-1's revision is published, this mapping will require recalibration; the architectural argument is independent of the specific framework version, since it operates on the structural properties of integrated runtime governance.

6.4 Proposed Minimum Category Criteria

We reformulate the six recommendations submitted to NIST CAISI [Brennan, 2026a] as *proposed* minimum criteria for the Attested Governance category. We acknowledge a structural constraint: the CAISI submission and this paper share an author, so what is presented as category criteria has emerged from a single author's perspective on what the category requires. We offer C1–C6 below as proposed criteria for community refinement, not as authoritative category specification. Standards bodies, research groups, and alternative implementers should refine, contest, and improve these. The existence of a peer instantiation (Mazzocchetti's Aegis [Mazzocchetti, 2026], introduced in Section 3.4) suggests the criteria are at a useful abstraction level for the category: specific enough to characterize architecture, abstract enough to admit independent implementations. Additional instantiations would settle the question more definitively.

C1: Sealed reference states. A system in the Attested Governance category produces cryptographically sealed, signature-immutable artifacts binding subject identity, policy reference, and enforcement parameters, where any modification is cryptographically detectable. A system in this category must seal subject identity, policy reference, and enforcement parameters into a single artifact (the Policy Artifact in AGA), so that none can be altered without breaking the artifact's signature.

C2: Continuous runtime measurement. A system in the Attested Governance category performs continuous measurement of runtime state against the sealed reference at a policy-specified cadence and produces a signed cryptographic commitment for each measurement. A system in this category must measure on a recurring cadence rather than on demand, binding a signed commitment to each measurement so the record of runtime state cannot be backdated or forged after the fact. The cadence is policy-set; the commitment may be a signed receipt or an equivalent.

C3: Tiered verification levels. A system in the Attested Governance category supports tiered verification calibrated to deployment consequence, with the tier expressed in receipt metadata so relying parties can apply appropriate confidence thresholds. A system in this category must support tiered verification calibrated to deployment consequence, with the tier indicated in receipt metadata so relying parties apply appropriate confidence thresholds, analogous to FIPS 140-2/3 security levels for cryptographic modules. The criterion is the architectural commitment to consequence-calibrated verification with metadata-indicated tier; specific tier structures are implementation choices. AGA's three-tier structure (Bronze device-attested, Silver server-attested, Gold blockchain-anchored) [Brennan, 2025, §L] is one instantiation. Alternative tier structures are possible and welcomed, including binary tiers, structures with four or more tiers, and structures calibrated to different consequence axes (regulatory, commercial, defense).

C4: Offline-verifiable evidence bundles. A system in the Attested Governance category produces evidence bundles verifiable using standard cryptographic primitives and the issuing authority's public key alone, without network connectivity to the producing system. A system in this category must produce bundles a verifier can check offline, with only standard cryptographic primitives and the issuing authority's public key. That offline property is the precondition for air-gapped, classified, or critical infrastructure deployments, where the verifier cannot reach the producing system.

C5: Standardized artifact formats. A system in the Attested Governance category uses machine-parseable, cryptographically interoperable artifact formats verifiable by a common verifier procedure across multiple implementations. A system in this category must use machine-parseable, cryptographically interoperable artifact formats. Multiple implementations should be able to produce evidence bundles verifiable by a common verifier procedure operating on standard primitives. (C5 is framed as a work item for future standards bodies, not as a self-contained specification; the criterion identifies what would need to be standardized.)

C6: Policy-constrained disclosure with cryptographic evidence. A system in the Attested Governance category supports structured disclosure of governed evidence under policy constraint, producing cryptographic evidence of what was disclosed, what was withheld, under what policy version, and with what rationale. A system in this category must make the disclosure decision itself provable, not only the evidence it releases. This addresses the disclosure problem that arises across organizational boundaries, where integrated governance evidence has to be shared with a party whose access is limited by policy: an auditor handed partial evidence should be able to confirm cryptographically that it is a policy-conformant disclosure rather than arbitrary redaction. AGA's mechanism, ordered substitute traversal with

signed substitution receipts and three disclosure modes (PROOF_ONLY for boolean attestation, REVEAL_MIN for bucketed or minimized values, REVEAL_FULL for complete values) [Brennan, 2025, §K, Claim 2], is one instantiation. Others are welcome, among them predicate proofs over claims, threshold disclosure schemes, and selective disclosure credentials.

These six criteria, taken together, define the proposed category. AGA is one instantiation that satisfies all six by design (Section 4.6). Other architectures may satisfy all six through different implementation choices, and we invite this. Systems satisfying only some criteria are partial coverage architectures within the emerging category; whether they qualify as Attested Governance systems is a question the criteria are designed to make tractable.

Two structural acknowledgments accompany this proposal. *First*, the criteria C1–C6 are existence-proven through one full coverage instantiation (AGA, Section 4). Whether C1–C6 are at the right abstraction level (neither so AGA-specific that no alternative instantiation can satisfy them, nor so abstract that they fail to distinguish the category from adjacent categories) remains open for additional independent instantiations to settle. We invite alternative instantiations as the continued path to confirming the criteria's standardization readiness; the criteria stand as one author's proposal informed by one full coverage implementation, awaiting further community engagement. *Second*, the category as proposed is motivated by and demonstrated against high consequence forensic scrutiny contexts. The Section 5 structural correspondence with regulated practice operates within this scope; the AI system contexts identified in Section 1 (federal defense procurement, healthcare/FDA AI, financial services AI, EU AI Act high-risk systems, critical infrastructure AI) operate within this scope. Whether the integrated four-pillar architecture is the right fit for AI governance contexts of lower consequence (where partial governance may suffice and where the architectural commitment may be operationally unjustified) is an open empirical question. This paper does not claim universality across all AI governance contexts; the name "Attested Governance" identifies the category as proposed in the forensic scrutiny scope, and we acknowledge the name may apply more narrowly than the broadest reading suggests.

6.5 Open Research Questions

Four research directions remain open in this category.

Behavioral integrity measurement. Current AGA measurement embodiments address binary integrity: drift in code, configuration, state. Behavioral integrity extends this to semantic drift in AI agent decision distributions, tool use patterns, and output characteristics. Binding behavioral measurements to sealed references and integrating them with the existing measurement protocol [Brennan, 2025, §E] is open work.

Multi-agent governance protocols. The recursive delegation primitive (a sub-agent operating under a constrained sub-mandate derived from the parent mandate, with TTL no longer than the parent and scope no broader) is defined and partially demonstrated [Brennan, 2026b, §4]. The full protocol for inter-agent communication governance under integrated runtime governance is open.

Post-quantum governance primitives. The architecture is algorithm-agile; receipt and artifact metadata specify the signature algorithm, enabling migration from Ed25519 to ML-DSA or

SLH-DSA without architectural change [Brennan, 2025, ¶89] (Section 4.4 demonstrates the migration mechanism through worked example). The open question is the standardization process (which post-quantum signature algorithm becomes the canonical default for governance artifacts requiring decades-long verifiability) and the migration path for existing AGA deployments at scale.

Standardization pathway development. The category criteria of Section 6.4 are at the right abstraction level for category definition but require substantial specification work to support conformance testing across implementations. Three IETF working groups touch directly adjacent territory: SCITT (Supply Chain Integrity, Transparency, and Trust) provides receipt and statement formats with transparency log primitives that could host AGA-style continuity chain receipts under extension, although the trust root unification Section 3.3 identifies remains open; RATS (Remote ATtestation procedureS, RFC 9334) provides architectural language for attestation, evidence, and verifier roles that maps onto portions of the four-pillar structure but does not address policy enforcement or continuity verification; WIMSE (Workload Identity in Multi-System Environments) addresses workload identity standardization that intersects with the subject identity binding Section 4.2 formalizes through Policy Artifact subject identifier construction. Whether the category's standardization pathway runs through extension of one of these working groups, through a new working group dedicated to integrated runtime governance, or through a different forum (NIST NCCoE follow-on, ISO/IEC SC 42, ITU-T) is open. The pathway choice depends on standards body engagement that has not yet occurred at the integrated architecture level; we surface the question, not answer it. We encourage engagement from working group participants, NIST CAISI and NCCoE staff, and standards body chairs interested in the category.

7 Conclusion

Runtime governance for autonomous AI systems is becoming a procurement requirement through multiple parallel tracks: federal procurement frameworks in active transition, sector regulators establishing post-deployment monitoring obligations, EU AI Act high-risk provisions, and standards bodies engaging the question of architectural primitives. The technical question is whether the proofs these frameworks will accept can be produced by composing existing categories or whether integration is the missing layer. We have argued integration is missing because it cannot be composed.

This paper makes four contributions. First, a formal definition of inseparability as a design imperative, distinct from the informal "tight coupling" framing and from impossibility theorems about composition. Second, a derivation of the integration property as a structural consequence of inseparability, with three qualifications that exclude pathological cases while admitting integrated architectures regardless of vendor scope. Third, an instantiation of the integrated architecture (AGA) within a stated trust boundary that names issuing authority and portal compromise as architectural specification, distinguishing the temporal bounding mechanism from the operationally determined magnitude of the bound. Fourth, structural correspondence with regulated industry practice operating under forensic scrutiny pressure for decades (industrial control, custody transfer measurement, electronic records), with the operational context gap to autonomous AI execution made explicit.

A reference implementation reported in Section 4.7 establishes that integration at this architectural standard is operationally feasible, including under post-quantum constraints. Section 6.5

identifies four research directions open in the category: behavioral integrity measurement extending current binary state measurement to semantic drift; multi-agent governance protocols built on the recursive delegation primitive; post-quantum primitive standardization; and standardization pathway development through SCITT, RATS, WIMSE, or a dedicated forum. The six minimum criteria proposed in Section 6.4 stand as a framework for community refinement; we encourage alternative implementations that satisfy them. The paper's weakest claim is the substrate transition conditional argument in Section 6.1: that integrated runtime governance becomes architecturally non-optional if and as AI moves from application layer toward the foundational stack. The transition is directional, not established; the architectural argument grounded in Section 5's structural correspondence with regulated infrastructure stands independently.

Forensic scrutiny requirements will continue to demand runtime governance for autonomous AI systems. The field's choice is whether to meet that demand through architectural integration (a coherent category with interoperable implementations and verifiable evidence) or through composed approximations that produce parallel verification, multiple trust roots, and unanswerable forensic questions. The category will exist; the question is what form it takes.

Acknowledgments

The NIST CAISI and NCCoE workstreams' framing of the procurement context shaped Sections 1.1 and 6. Section 3.4 draws on Mazzocchetti's Aegis architecture [Mazzocchetti, 2026] as an independently developed peer instantiation in the category. The author used AI tools for prose copyediting; all technical content, analysis, and results are the author's own.

Conflict of Interest Disclosure

The author is the named inventor on USPTO Application No. 19/433,835, "Systems and Methods for Generating and Enforcing Attested Governance Artifacts," cited throughout Section 4, and the founder and sole owner of Attested Intelligence Holdings LLC (Illinois File No. 17233815), which holds prosecution rights to the application and develops the Attested Governance Artifacts implementation referenced in the same section. Section 1.3 addresses the relationship between the architectural argument and the specific implementation reference. The preparation of this paper received no external funding.

The criteria C1–C6 in Section 6.4 are informed by AGA's architectural choices, and two of them correspond to subject matter disclosed in the cited patent application (USPTO Application 19/433,835): C3 (tiered verification levels) corresponds to §L; C6 (policy-constrained disclosure with cryptographic evidence) corresponds to §K and Claim 2. The criteria have been framed at the level of architectural properties rather than specific mechanisms so that alternative instantiations beyond AGA can satisfy them. Standards bodies and other implementers refining the criteria should consider the patent prosecution status as it evolves, and whether the framing should be calibrated independently of the author's prosecution interests.

Availability

The AGA reference implementation referenced in Section 4 is available through publicly distributed artifacts: the `aga-governance` package on PyPI and the `@attested-intelligence/aga-mcp-server` package on npm, with source code at https://github.com/attestedintelligence/aga-mcp-server. A public demonstration repository accompanying this paper is available at the same location. The full text of USPTO Application No. 19/433,835 will become publicly available upon USPTO publication (estimated ∼July 1, 2027); the present paper relies on the application as filed.

References

American Petroleum Institute. Manual of Petroleum Measurement Standards, Chapter 21.1: Electronic Gas Measurement. API MPMS Chapter 21.1, 2nd ed., 2013. URL https://www.api.org/products-and-services/standards/important-standards-announcements/standard-21.

Henk Birkholz, Dave Thaler, Michael Richardson, Ned Smith, and Wei Pan. Remote ATtestation procedureS (RATS) Architecture. RFC 9334, January 2023. URL https://www.rfc-editor.org/rfc/rfc9334.

Henk Birkholz, Antoine Delignat-Lavaud, Cédric Fournet, Yogesh Deshpande, and Steve Lasker. An Architecture for Trustworthy and Transparent Digital Supply Chains. IETF SCITT Working Group, Internet-Draft `draft-ietf-scitt-architecture`, 2025. URL https://datatracker.ietf.org/doc/draft-ietf-scitt-architecture/.

Jack Brennan. Systems and Methods for Generating and Enforcing Attested Governance Artifacts. U.S. Patent Application No. 19/433,835, December 2025. URL https://patentcenter.uspto.gov/applications/19433835. Filed December 28, 2025, micro entity, 20 claims (3 independent, 17 dependent). Estimated publication July 2027. Attested Intelligence Holdings LLC, Illinois.

Jack Brennan. Comment on NIST CAISI RFI: Security Considerations for AI Agents. Public comment filed in NIST Docket No. NIST–2025–0035, March 2026a. URL https://www.regulations.gov/docket/NIST-2025-0035. Filed March 4, 2026 by Attested Intelligence Holdings LLC in response to National Institute of Standards and Technology [2026].

Jack Brennan. Comment on NCCoE Concept Paper: AI Agent Identity and Authorization. NCCoE Concept Paper Response, March 2026b. Filed March 4, 2026 by Attested Intelligence Holdings LLC in response to National Cybersecurity Center of Excellence [2026].

European Parliament and Council of the European Union. Regulation (EU) 2024/1689 Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act). Official Journal of the European Union, L series, July 2024. URL https://eur-lex.europa.eu/eli/reg/2024/1689/oj.

Executive Office of the President. America's AI Action Plan. White House publication, July 2025. URL https://www.whitehouse.gov/wp-content/uploads/2025/07/Americas-AI-Action-Plan.pdf.

Executive Office of the President, Office of Management and Budget. Accelerating Federal Use of AI through Innovation, Governance, and Public Trust. OMB Memorandum M–25–21, April 2025a. URL https://www.whitehouse.gov/wp-content/uploads/2025/02/M-25-21-Accelerating-Federal-Use-of-AI-through-Innovation-Governance-and-Public-Trust.pdf.

Executive Office of the President, Office of Management and Budget. Driving Efficient Acquisition of Artificial Intelligence in Government. OMB Memorandum M–25–22, April 2025b. URL https://www.whitehouse.gov/wp-content/uploads/2025/04/M-25-22.pdf.

International Society of Automation and International Electrotechnical Commission. Security for Industrial Automation and Control Systems – Part 3-3: System Security Requirements and Security Levels. ISA/IEC 62443-3-3:2013, 2013. URL https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards.

Maurits Kaptein, Vassilis-Javed Khan, and Andriy Podstavnychy. Runtime Governance for AI Agents: Policies on Paths. arXiv preprint arXiv:2603.16586, March 2026. URL https://arxiv.org/abs/26

03.16586. Formalizes runtime governance as path-dependent compliance policy mapping agent identity, execution path history, proposed actions, and organizational context to violation probability.

Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, June 2013. URL https://www.rfc-editor.org/rfc/rfc6962.

Adam Massimo Mazzocchetti. Cryptographic Runtime Governance for Autonomous AI Systems: The Aegis Architecture for Verifiable Policy Enforcement. arXiv preprint arXiv:2603.16938, March 2026. URL https://arxiv.org/abs/2603.16938. Single-author work introducing the Aegis architecture: Immutable Ethics Policy Layer (IEPL), Ethics Verification Agent (EVA), Enforcement Kernel Module (EKM), and Immutable Logging Kernel (ILK); 238 ms median proof-verification latency under controlled internal trials in the Civitas runtime environment.

National Cybersecurity Center of Excellence. Accelerating the Adoption of Software and AI Agent Identity and Authorization. NCCoE Concept Paper, February 2026. URL https://www.nccoe.nist.gov/projects/accelerating-adoption-software-and-ai-agent-identity-and-authorization. Comment period February 5 – April 2, 2026.

National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). Technical Report NIST AI 100-1, U.S. Department of Commerce, January 2023. URL https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf.

National Institute of Standards and Technology. Cybersecurity Framework Profile for Artificial Intelligence (Cyber AI Profile). Technical Report NIST IR 8596 (Initial Preliminary Draft), National Institute of Standards and Technology, December 2025. URL https://csrc.nist.gov/pubs/ir/8596/iprd.

National Institute of Standards and Technology. Request for Information Regarding Security Considerations for Artificial Intelligence Agents. NIST CAISI RFI, Docket No. NIST–2025–0035, 91 Fed. Reg. 698, January 2026. URL https://www.federalregister.gov/documents/2026/01/08/2026-00206/request-for-information-regarding-security-considerations-for-artificial-intelligence-agents. Published January 8, 2026; comment period closed March 9, 2026.

Zachary Newman, John Speed Meyers, and Santiago Torres-Arias. Sigstore: Software Signing for Everybody. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*, pages 2353–2367. Association for Computing Machinery, 2022. doi: 10.1145/3548606.3560596. URL https://dl.acm.org/doi/10.1145/3548606.3560596.

Open Policy Agent Contributors. Open Policy Agent (OPA). Cloud Native Computing Foundation Graduated Project, 2024. URL https://www.openpolicyagent.org/.

Open Source Security Foundation (OpenSSF). SLSA: Supply-chain Levels for Software Artifacts. Specification, version 1.1, 2024. URL https://slsa.dev/spec/v1.1/.

Anders Rundgren, Bradley Jordan, and Samuel Erdtman. JSON Canonicalization Scheme (JCS). RFC 8785, June 2020. URL https://www.rfc-editor.org/rfc/rfc8785.

Keith Stouffer, Michael Pease, CheeYee Tang, Timothy Zimmerman, Victoria Pillitteri, Suzanne Lightman, Adam Hahn, Stephanie Saravia, Aslam Sherule, and Michael Thompson. Guide to Operational Technology (OT) Security. Technical Report NIST SP 800-82 Rev. 3, National Institute of Standards and Technology, September 2023. URL https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r3.pdf.

Donald J. Trump. Executive Order 14179: Removing Barriers to American Leadership in Artificial Intelligence. 90 Fed. Reg. 8741, January 2025. URL https://www.federalregister.gov/documents/2025/01/31/2025-02172/removing-barriers-to-american-leadership-in-artificial-intelligence. Signed January 23, 2025; published January 31, 2025.

U.S. Food and Drug Administration. Electronic Records; Electronic Signatures. 21 C.F.R. pt. 11, 1997. URL https://www.ecfr.gov/current/title-21/chapter-I/subchapter-A/part-11. Codified at 21 C.F.R. §§ 11.1–11.300; most recent revision incorporated by reference.

Aojie Yuan, Zhiyuan Su, and Yue Zhao. AEGIS: No Tool Call Left Unchecked – A Pre-Execution Firewall and Audit Layer for AI Agents. arXiv preprint arXiv:2603.12621, March 2026. URL https://arxiv.org/abs/2603.12621.

Ziling Zhou. Governing Dynamic Capabilities: Cryptographic Binding and Reproducibility Verification for AI Agent Tool Use. arXiv preprint arXiv:2603.14332, March 2026. URL https://arxiv.org/abs/2603.14332.