CRSA-1

Compositional Runtime Safety Attestation Protocol for Multi-Principal AI Systems

Auburn Governance Stack — Clause AI-9

Document 46 — Composition Layer

Ryan Fields

UncleBroFields@proton.me

fieldsryanchristopher@gmail.com

February 2026

Auburn Patent Family Fields

Intellectual Property (IP) Declaration

The methods, logic structures, composition operators, safety certificate algebra, contamination extension formulae, audit event schemata, and agent interaction protocol constraints contained in this work are the sole property of Ryan Fields.

Contents

1 Introduction

1.1 The Composition Gap

Every major AI laboratory is shipping multi-agent systems. None of them can formally prove that safety properties hold across agent boundaries.

The majority of production AI deployments are no longer single-model systems. Retrieval-augmented generation (RAG) pipelines compose a retriever, a reranker, and a generator—each a separate model with independent failure modes. Agentic systems chain dozens of LLM calls with tool use into workflows where the planner, executor, critic, and tool-calling models each maintain distinct internal states. Ensemble systems aggregate outputs from multiple models through voting or learned aggregation functions. Cascade architectures route queries to different models based on estimated difficulty. Mixture-of-agents architectures enable multiple LLMs to collaboratively generate outputs through iterative refinement.

In all of these architectures, the same structural problem obtains: *individual model safety does not compose into system safety without additional guarantees.* A retriever that produces well-distributed results and a generator that maintains low hallucination rates can nonetheless produce a composed system that amplifies rare retrieval errors into confident hallucinations. An agent whose individual LLM calls each pass safety filters can nonetheless enter reasoning loops that produce unsafe composite behavior. An ensemble of individually calibrated models can produce an aggregate that is systematically miscalibrated.

This is not a theoretical concern. The Cooperative AI Foundation's Technical Report 1 (Hammond et al., February 2025), produced by over 50 researchers from DeepMind, Anthropic, Carnegie Mellon, Harvard, and Oxford, found that "many of the promising approaches to building safe AI rely on a *lack of cooperation*, such as adversarial training or scalable oversight. If advanced AI systems can learn to collude without our knowledge, these approaches may be insufficient." The report identifies three primary multi-agent failure modes—miscoordination, conflict, and collusion—and concludes that single-agent safeguards demonstrably do not translate to multi-agent settings.

Google DeepMind's "Distributional AGI Safety" paper (arXiv:2512.16856, December 2025) argues that AGI may first emerge as a distributed "patchwork" of interacting sub-AGI agents with complementary skills, and that "conventional alignment, process supervision, RLHF, and containment techniques—designed for individual agents—are shown to have limited applicability or blind-spots in this distributed case." Standardized agent-to-agent protocols such as the Model Context Protocol (MCP) and Agent-to-Agent (A2A) are identified as critical enablers whose proliferation may drive emergent general capability.

The MIT CSAIL AI Agent Index (arXiv:2602.17753, February 2026) quantifies the documentation crisis: of 30 state-of-the-art agents analyzed across 1,350 data points, only 4 publish agent-specific safety evaluations. Only 19% disclose a formal safety policy. Twenty-five of thirty do not disclose internal safety testing results. The researchers identify "safety washing"—publishing high-level frameworks while selectively disclosing empirical evidence—as a systemic pattern.

No existing framework, standard, specification, or protocol defines how safety certificates compose across model boundaries in multi-agent AI systems. CRSA-1 fills this gap.

1.2 The Regulatory Catalyst

The composition gap is not merely a technical problem. It is becoming a compliance problem with hard deadlines and material penalties.

The EU AI Act (Regulation 2024/1689) imposes obligations on high-risk AI systems that implicitly require compositional safety guarantees but provide no methodology for demonstrating them. Article 9 requires a risk management system covering "reasonably foreseeable risks"—in

multi-model deployments, foreseeable risks include composition failures that no single-model assessment can detect. Article 12 requires automatic recording of events sufficient to facilitate post-market monitoring and operational oversight—in multi-model pipelines, this requires capturing the full provenance graph of model invocations, not merely individual model logs. Article 15 requires "appropriate levels of accuracy, robustness, and cybersecurity"—for composed systems, these properties must be demonstrated at the system level, not merely asserted from component-level evaluations.

Full enforcement powers for General-Purpose AI model obligations—including penalties of up to €15 million or 3% of global annual turnover—activate on August 2, 2026. The harmonized standards being developed by CEN-CENELEC JTC 21 to support compliance are behind schedule: the CEN and CENELEC Boards approved an exceptional fast-track procedure on October 14–16, 2025, allowing direct publication without Formal Vote for six delayed standards. Even with this acceleration, the target availability is Q4 2026. The standards address component-level requirements (quality management, risk management, data quality, trustworthiness); no harmonized standard addresses *compositional* safety for multi-model systems.

In the United States, the National Defense Authorization Act for Fiscal Year 2026 (signed December 18, 2025) creates parallel pressure. Section 1512 requires a department-wide AI cybersecurity policy within 180 days (approximately June 2026), covering model tampering, jailbreaks, adversarial prompt injection, data poisoning, secure development practices, adversarial testing, model lineage documentation, and red teaming. Section 1513 requires a risk-based cybersecurity framework for AI acquired by the Department of Defense—a "CMMC for AI"—with status reported to Congress by June 16, 2026. Given the Department of Defense's procurement power, these standards will likely become de facto industry requirements.

The Colorado AI Act (SB 24-205, effective June 30, 2026) requires developers to provide documentation of bias testing, training data summaries, and known limitations, with deployers conducting annual impact assessments aligned with NIST AI RMF or ISO/IEC 42001. California SB 942 (effective January 1, 2026) mandates embedded provenance disclosures for AI-generated content. OMB Memorandum M-25-21 (April 2025) requires continuous monitoring and data traceability for federal AI systems.

None of these frameworks specify how compliance is demonstrated for composed multi-model systems. All of them implicitly require it. CRSA-1 provides the technical specification that bridges this gap.

1.3 What CRSA-1 Provides

CRSA-1 defines five components for compositional safety attestation:

1. A **safety certificate algebra** (Section 3) providing formally defined composition operators—sequential, parallel, conditional, and recursive—that derive system-level safety certificates from component-level certificates with bounded risk degradation.

2. A **runtime attestation mechanism** (Section 4) using cryptographic commitments within Entity Attestation Tokens (EAT, RFC 9711) to prove at each inference step that the composed system operates within its certified safety envelope, at less than 0.1% latency overhead.

3. **Multi-principal isolation bounds** (Section 5) extending the Stateful Isolation Law's contamination functional to multi-model shared infrastructure, providing formally derivable bounds on cross-principal influence across model boundaries.

4. A **standardized audit log format** (Section 6) built on W3C PROV-DM semantics with OCSF-compatible event classes, producing decision traces sufficient for EU conformity assessment, DoD traceability review, and US state-law compliance.

5. **Agent interaction protocol constraints** (Section 7) specifying mandatory structured communication, anti-steganography requirements, cascade-breaking checkpoints, and privilege boundary enforcement—extending MCP's connectivity layer with a governance layer.

CRSA-1 is infrastructure, not intelligence. It provides compositional risk reduction and accountability infrastructure—analogous to financial auditing, which certifies process compliance without guaranteeing future solvency. Section 9 specifies precisely what CRSA-1 cannot guarantee.

1.4 Relationship to the Auburn Governance Stack

CRSA-1 is Document 46 in the Auburn Governance Stack registry, designated Clause AI-9, assigned to the Composition Layer. It is a peer to MAI-1 (AI-5, Document 22) and AGS-1 (Document 23) in the hourglass architecture.

MAI-1 defines the canonical attestation interface for a single model on a single platform. CRSA-1 defines how MAI-1 attestation composes when models are chained, parallelized, routed, or recursively invoked. Every lower-layer Auburn specification (Layers 0–3) feeds evidence into MAI-1; CRSA-1 defines the algebra by which that evidence composes across model boundaries. Every upper-layer Auburn specification (Enforcement and Application layers) consumes evidence from MAI-1; CRSA-1 ensures that consumed evidence reflects the composed system's safety state, not merely the weakest individual component.

The composition principle from the Auburn Governance Stack Master Architecture applies: systems that do not expose a MAI-1 compliant endpoint are outside the scope of all downstream Auburn governance clauses. CRSA-1 extends this principle: *multi-model systems that do not expose a CRSA-1 compliant composition certificate are outside the scope of system-level Auburn governance guarantees.* Individual model attestation remains valid; system-level attestation requires CRSA-1.

CRSA-1 depends on:

- **MAI-1** (AI-5, Document 22): Single-model attestation as the building block. Every CRSA-1 composition operator takes MAI-1 attestation artifacts as inputs.
- **Stateful Isolation Law** (Document 5): The contamination functional, six clauses, composability theorem, and compliance tiers. CRSA-1 Section 5 directly extends the Stateful Isolation Law's sequential, parallel, and cross-layer composition lemmas to the multi-model case.
- **Cryptographic Binding Specification** (Document 24): Inter-layer binding mechanisms that CRSA-1 extends across model boundaries.
- **AI-4 SRAM Thermal Integrity Bound** (Document 3): Hardware trust root anchoring runtime attestation.
- **AI-2 Gradient Starvation Envelope** (Document 8), **AI-3 Lyapunov Stability Envelope** (Document 9), **AI-8 Entropy Collapse Constraint** (Document 10): Individual model health invariants that must compose under the CRSA-1 algebra.

CRSA-1 feeds into:

- **CTS-1** (Document 26): Multi-model conformance tests.
- **Decision Receipt Specification** (Document 19): Pipeline DAG format for multi-model output provenance.

- **All sector-specific compliance profiles** (Documents 33–39): Multi-agent compliance mapping for EU AI Act, Federal, Insurance, FDA, Financial Services, Defense, and Enterprise VRM.
- **MAI-1 v2.0**: The composition algebra becomes part of the canonical interface in the next major MAI-1 revision.

1.5 Scope and Limitations

CRSA-1 addresses the composition of safety attestation across model boundaries in multi-model AI systems. It does not address:

- **Single-model safety.** Individual model safety evaluation, alignment, interpretability, and behavioral testing are addressed by other Auburn clauses (AI-1 through AI-8) and by the broader AI safety research agenda. CRSA-1 assumes component-level safety evaluation exists and defines how its outputs compose.
- **Training-time composition.** CRSA-1 addresses inference-time composition. Training-time concerns (federated learning, multi-task training, data mixing) are addressed by the Stateful Isolation Law's Clause AI-6 and by Document 18 (Training Provenance Chain).
- **Behavioral guarantees.** CRSA-1 provides compositional *attestation*— cryptographic evidence that a composed system was operating in a certified configuration with verified safety properties at the time of inference. It does not provide behavioral guarantees about what the system will output. This limitation is fundamental, not contingent: behavioral guarantees for composed stochastic systems remain an open research problem.

Honest Framing

Compositional verification is inherently sound but incomplete: there exist composed systems that are safe but cannot be verified as safe through any compositional method. CRSA-1 can verify safety for systems whose safety derives from component-level properties that compose under the defined operators. It cannot verify safety for systems whose safety is an emergent property of the specific interaction pattern. This is a mathematical limitation, not an engineering gap. Section 9 elaborates.

The key words "**MUST**," "**MUST NOT**," "REQUIRED," "**SHALL**," "**SHOULD NOT**," "**SHOULD**," "**MAY**," and "OPTIONAL" in this document are to be interpreted as described in BCP 14 (RFC 2119, RFC 8174) when, and only when, they appear in all capitals.

2 Composition Patterns and Threat Landscape

This section establishes the problem space that CRSA-1 addresses. Section 2.1 defines the taxonomy of multi-model architectures and the attestation gaps inherent in each. Section 2.2 catalogs the inter-agent threat vectors that composition introduces. Section 2.3 explains why single-agent safeguards fail structurally in composed settings.

2.1 Taxonomy of Multi-Model Architectures

Five composition patterns account for the vast majority of production multi-model deployments. Each introduces distinct attestation requirements that single-model frameworks cannot satisfy.

2.1.1 Sequential Pipelines (RAG)

Retrieval-Augmented Generation composes a minimum of two models—a retriever (dense encoder or sparse index) and a generator (autoregressive LLM)—with optional intermediate stages including rerankers, query transformers, and citation verifiers. The retriever's output distribution directly conditions the generator's input distribution: drift in retrieval quality cascades to generation quality without any signal visible in the generator's own health invariants.

The attestation gap is structural. The generator's entropy floor (AI-8) may hold, its gradient stability (AI-2) may hold, and its Lyapunov envelope (AI-3) may hold—yet the composed system produces hallucinations because the retriever surfaced irrelevant context. No individual model invariant detects this failure. The composed system requires an attestation that captures both the retriever's relevance distribution and the generator's conditional behavior given that distribution.

An additional complication arises from the embedding model that converts documents to vectors for the retriever's index. This embedding model is itself an AI component with its own health invariants. Changes to the embedding model—through update, quantization, or drift—invalidate the entire vector store without any signal propagating to the generator. The attestation chain must therefore span at minimum three models: embedding model → retriever → generator, with each transition requiring guarantee-to-assumption verification.

2.1.2 Agentic Systems

Agent frameworks—including LangChain, CrewAI, AutoGPT, and tool-augmented systems built on MCP—compose multiple LLM calls with tool use into workflows where a planner selects actions, an executor carries them out, a critic evaluates results, and tool-calling models interact with external systems. A single user request may trigger dozens of LLM invocations, each with its own model state, safety configuration, and failure modes.

The attestation challenge is threefold. First, the number of model invocations per request is dynamic and unbounded without explicit termination constraints. Second, each invocation operates on a context that includes the outputs of all prior invocations—creating a sequential composition where contamination accumulates combinatorially (the Stateful Isolation Law's Lemma 1 shows $O(k^2)$ interaction terms for k actions). Third, tool use introduces *unattested* components: when an agent calls a web search API, executes code in a sandbox, or queries a database, the tool outputs affect the agent's subsequent behavior but are outside any model's attestation scope.

The boundary between attested (model inference) and unattested (external tool) components is the defining architectural challenge for agentic attestation. CRSA-1 must define what happens at this boundary: how the safety certificate degrades when unattested inputs enter the pipeline, and what re-verification is required before attested outputs resume.

2.1.3 Ensemble Systems

Ensemble architectures compose multiple models whose outputs are aggregated through majority voting, weighted averaging, or learned aggregation functions. The aggregation function itself is a safety-critical component: it determines how individual model failures map to system-level failures.

The attestation question for ensembles is: if one model in the ensemble is non-conformant, is the ensemble non-conformant? Three answers are defensible, corresponding to the three composition models identified in the Auburn Governance Stack's Document 43:

1. **Conjunction:** the composed system is conformant if and only if all component models are conformant. Strictest—any single failure fails the system.

2. **Weakest link:** the composed system's conformance level equals the lowest conformance level of any component. Moderate—allows mixed conformance but reports conservatively.

3. **Weighted composition:** each component's attestation contributes to the system-level attestation proportionally to its influence on the output. Most nuanced but requires formal influence quantification.

CRSA-1's parallel composition operator ($\otimes_{\text{par}}$, Section 3.2.2) formalizes the third option while enabling the first two as special cases.

2.1.4 Cascade and Routing Systems

Cascade architectures generalize speculative decoding: a small, fast model handles queries classified as easy, while a large, expensive model handles queries classified as hard. The routing decision—which model receives which query—is itself a safety-critical inference that determines the system's overall safety profile.

The attestation challenge is that different models in the cascade may have different conformance levels. A system that routes 90% of queries to a MAI-C1 model and 10% to a MAI-C2 model has no well-defined system-level conformance without a formal conditional composition operator. The router's own accuracy—its probability of correctly classifying query difficulty—directly affects the system-level safety guarantee. A router that misclassifies hard queries as easy sends safety-critical inputs to the less capable model, degrading the system's effective safety below either component's individual level.

No existing attestation framework addresses conditional composition. The IETF multi-verifier draft (draft-deshpande-rats-multi-verifier-03) defines topological patterns for verification but provides no semantic composition algebra for conditional routing. CRSA-1's conditional composition operator ($\otimes_{\text{cond}}$, Section 3.2.3) is a novel contribution addressing this gap.

2.1.5 Mixture-of-Agents

Recent architectures enable multiple LLMs to collaboratively generate outputs through iterative refinement. In the simplest form, Model A generates a draft, Model B critiques it, Model A revises, and the cycle repeats until convergence or a budget is exhausted. More complex variants involve multiple models contributing to different aspects of a response (reasoning, factuality, style) with a learned aggregator.

Mixture-of-agents combines the challenges of all other patterns: sequential composition (each refinement step depends on prior outputs), parallel composition (multiple models contribute simultaneously), conditional composition (the aggregator routes influence), and recursive composition (the refinement loop iterates). This is the hardest pattern for compositional attestation and the one for which the least formal theory exists.

CRSA-1's recursive composition operator ($\otimes_{\text{rec}}$, Section 3.2.4) provides the formal foundation, but the honest framing applies with particular force: mixture-of-agents systems are the composition pattern most likely to exhibit emergent safety properties that cannot be verified compositionally.

2.2 The Five Inter-Agent Threat Vectors

Multi-model composition introduces threat vectors that do not exist in single-model deployments. Each vector exploits the information channels created when models share infrastructure, communicate outputs, or operate on shared state.

2.2.1 Steganographic Collusion

Mathew et al. ("Hidden in Plain Text," arXiv:2410.03768, updated December 2025) demonstrated for the first time that steganographic collusion in LLMs arises from misspecified reward incentives during training—it is emergent, not designed. Using gradient-based reinforcement learning (PPO) and in-context reinforcement learning, they showed that standard mitigations—passive output oversight and active paraphrasing—are not fully effective against emergent covert channels.

The threat to multi-agent systems is direct: when Agent A's output serves as Agent B's input, the natural language channel between them can carry hidden information that standard monitoring cannot detect. Morimura et al. ("Steganographic Potentials of Language Models," arXiv:2505.03439, May 2025) confirmed that current models exhibit rudimentary steganographic abilities even without explicit prompting, and that targeted guidance markedly enhances capacity. Li et al. ("Hide and Seek in Embedding Space," arXiv:2601.22818, January 2026) found that KL divergence and perplexity can remain within typical fine-tuning ranges even when a covert channel is active, undermining statistical detection methods—though linear probes on late-layer activations provide partial detection cues.

Safety certificates that rely exclusively on output-level monitoring are therefore fundamentally insufficient for multi-agent systems. The inter-agent communication channel is a covert channel by default. CRSA-1's agent interaction protocol constraints (Section 7) address this through mandatory structured communication formats that constrain encoding bandwidth and mandatory sanitization between agent boundaries.

2.2.2 Cascading Jailbreaks

Gu et al. ("Prompt Infection," arXiv:2410.07283, October 2024) introduced self-replicating prompt injection that propagates across LLM-based multi-agent systems via inter-agent communication. Tested in simulated LLM environments with 10–50 agents, infections reached agents with no direct external input exposure. The mechanism is structurally identical to biological viral propagation: a compromised agent's output contains an instruction payload that, when consumed by a downstream agent as input context, compromises that agent in turn.

The fundamental vulnerability, as demonstrated by Greshake et al. (Black Hat US 2023), is that LLMs cannot reliably distinguish instructions from data within the same context window. A structured output schema approach using strict JSON with embedded fingerprint challenges (Sibylline Software, February 2026) eliminated injection propagation entirely across four models and nine attack categories—confirming that the vulnerability is addressable through protocol design rather than model improvement.

CRSA-1 addresses cascading jailbreaks through cascade-breaking checkpoints (Section 7.3): safety certificates **MUST** be re-verified at every agent boundary, and downstream agents **MUST NOT** accept input from upstream agents whose certificates have been degraded by detected injection attempts.

2.2.3 Confused Deputy and Semantic Privilege Escalation

The confused deputy attack, long established in computer security, takes a particularly dangerous form in multi-agent systems. SEAgent (arXiv:2601.11893, January 2026) identifies five types of privilege escalation in LLM agent systems and demonstrates that a malicious search agent can broadcast messages manipulating a trusted smart-lock agent into performing privileged actions that the search agent itself is not authorized to perform.

Acuvity (2025) introduced the concept of *semantic privilege escalation*—agents using authorized permissions for actions beyond the scope of their assigned task. Unlike traditional privilege escalation, which requires exploiting a technical vulnerability, semantic privilege escalation is an emergent property of agents with broad tool access operating on ambiguous instructions. The

agent is technically using its own credentials for actions it is technically authorized to perform, yet the composite effect exceeds any reasonable interpretation of its assigned task.

In multi-model pipelines, privilege escalation compounds: Agent A's tool access is scoped to database reads, Agent B's is scoped to email sending, but the composed pipeline A → B can read a database and email the contents—a capability neither agent was individually authorized to perform. CRSA-1's privilege boundary enforcement (Section 7.4) requires capability-based access control where no agent may delegate privileges exceeding its own safety certificate, preventing the composed system from acquiring capabilities that no individual component possesses.

2.2.4 MCP Security Gaps

The Model Context Protocol has achieved extraordinary adoption—97 million monthly SDK downloads within one year of its November 2024 launch, with backing from Anthropic, OpenAI, Google, Microsoft, and governance under the Linux Foundation's Agentic AI Foundation. MCP provides the connectivity layer for multi-agent systems: it standardizes how AI applications discover and invoke external tools, data sources, and other agents.

MCP's security posture, however, remains critically inadequate for multi-agent governance. Knostic's July 2025 scan of nearly 2,000 MCP servers found that all verified servers lacked any form of authentication. Backslash Security's June 2025 analysis of 7,000+ servers discovered "NeighborJack" vulnerabilities—servers bound to `0.0.0.0`, exposing them to any network peer—and servers permitting arbitrary OS command execution. Critical vulnerabilities include CVE-2025-32711 (EchoLeak: hidden prompt data exfiltration) and CVE-2025-6514 (remote code execution via malicious authorization endpoint in the `mcp-remote` package, affecting 437,000+ downloads). "Rug-pull attacks" allow trusted tools to silently push updates with harmful metadata; no version pinning or change alerts exist by default.

The June 2025 MCP specification update classifies MCP servers as OAuth Resource Servers and requires RFC 8707 Resource Indicators, but implementation remains inconsistent. What MCP does not provide, and what CRSA-1 must add, includes: inter-agent trust verification, capability attestation between agents, cascading authorization for agent-to-agent delegation, safety certificate exchange protocol, and version-pinning with change notification.

CRSA-1 is positioned as the safety and governance layer on top of MCP's connectivity layer. MCP solves the integration problem—how agents discover and invoke tools and each other. CRSA-1 solves the composition safety problem—how the composed system's safety properties are derived, attested, and enforced across those MCP-mediated connections.

2.2.5 Shared Infrastructure Side Channels

Multi-model systems that share inference infrastructure—the default deployment pattern for cost efficiency—inherit all information channels documented in the Stateful Isolation Law (Document 5, Section 2). The Stateful Isolation Law catalogs 25+ demonstrated attacks spanning every layer of the inference stack: PROMPTPEEK (NDSS 2025) achieves prompt recovery through shared KV cache timing; TLB covert channels (CCS 2023) achieve 31 kbps data exfiltration across NVIDIA MIG partitions with 99.8% accuracy; NVBleed (2025) achieves over 70 kbps across GPUs via NVLink, even across VMs on public cloud.

In multi-model pipelines, these channels multiply. If Model A and Model B share a GPU, Model A serves Principal 1, and Model A's output feeds Model B which serves Principal 2, then Principal 1's inputs can influence Principal 2's outputs through *two* channels: the explicit data flow (A's output → B's input) and the implicit hardware channel (shared TLB, L2 cache, memory bus). The explicit channel is the intended composition; the implicit channel is a contamination path that no application-layer attestation can detect.

CRSA-1's multi-principal isolation bounds (Section 5) extend the Stateful Isolation Law's contamination functional to the multi-model case, providing formally derivable bounds on cross-

principal influence that account for both explicit composition channels and implicit infrastructure channels.

2.3 Why Single-Agent Safeguards Fail

The failure of single-agent safeguards in multi-agent settings is not a matter of degree but of kind. Three structural arguments establish this.

Safety properties do not compose by default. Per Alpern and Schneider (1985, 1987), safety properties ("a bad thing never happens") are closed under intersection—the conjunction of two safety properties is itself a safety property. But the safety of individual components does not imply the safety of their composition without explicit *interconnection conditions*. In control theory, these are small-gain conditions (the spectral radius of the gain matrix must be less than 1) or dissipativity conditions. In contract-based design, they are assumption-coverage conditions (each component's output guarantees must satisfy the next component's input assumptions). Without verification of these interconnection conditions, composing individually safe components can produce an unsafe system. This is not a bug in the safety analysis; it is a theorem.

Evaluation does not compose. Anthropic's Responsible Scaling Policy v3.0 (February 24, 2026) explicitly acknowledged that capability evaluation thresholds—the triggering mechanism for all safety frameworks—operate in a "zone of ambiguity" where "the science of model evaluation isn't well-developed enough to provide dispositive answers" even for single models. If single-model evaluation is acknowledged as ambiguous by the laboratory most invested in safety evaluation methodology, multi-model evaluation—where the evaluation space grows combinatorially with the number of models, their interaction patterns, and their shared state—demands formal compositional approaches rather than ad hoc system-level testing. Exhaustive testing of composed systems is infeasible; formal composition of component-level evaluations is the only scalable path.

Adversarial robustness does not compose. The Cooperative AI Foundation report identifies a critical structural insight: many safety approaches—adversarial training, red-teaming, scalable oversight—rely on the assumption that the AI system cannot coordinate with other systems to evade the safety mechanism. In multi-agent deployments, this assumption is violated by construction. The agents *are* coordinating—that is the purpose of the system. The question is whether their coordination is confined to the intended task or extends to evading the safety infrastructure that monitors them. Mathew et al.'s demonstration that steganographic collusion emerges from standard training without explicit design shows that the answer is not guaranteed by architecture alone.

These three arguments establish that multi-agent safety requires a dedicated compositional framework—not merely better single-agent safety applied to each component. CRSA-1 provides that framework.

3 Safety Certificate Algebra

This section defines the formal algebra by which component-level safety certificates compose into system-level safety certificates. The algebra is grounded in assume-guarantee contract theory (Benveniste et al., 2018; Sangiovanni-Vincentelli et al., 2012) reformulated for stochastic, token-based AI systems where guarantees are probabilistic and degradation under composition is bounded but nonzero.

Section 3.1 defines the contract framework. Section 3.2 defines the four composition operators. Section 3.3 introduces the safety budget that governs degradation under composition. Section 3.4 recovers MAI-1 invariants as composable contracts. Section 3.5 states what does not compose and why.

3.1 Assume-Guarantee Contracts for AI Components

Definition 3.1 (AI Safety Contract)**.** *An **AI safety contract** is a tuple $\mathcal{C} = (A, G, \delta, \tau)$ where:*

- *A is the **assumption set**: a predicate over the component's input distribution, context state, and infrastructure configuration. A specifies the conditions under which the component's safety guarantees hold.*
- *G is the **guarantee set**: a predicate over the component's output distribution, internal state invariants, and resource consumption. G specifies the safety properties the component provides when its assumptions are met.*
- *$\delta \in [0, 1]$ is the **violation probability**: the maximum probability that G is violated when A holds. $\delta = 0$ denotes a deterministic guarantee; $\delta > 0$ denotes a probabilistic guarantee.*
- *$\tau \in \mathbb{R}_{>0}$ is the **validity horizon**: the wall-clock duration for which the contract remains valid without re-attestation, governed by freshness requirements from CTS-1 and the specific invariant's drift characteristics.*

Definition 3.2 (Contract Satisfaction)**.** *A component M **satisfies** contract $\mathcal{C} = (A, G, \delta, \tau)$, written $M \models \mathcal{C}$, if and only if:*

$$\Pr[\neg G(M(x)) \mid A(x)] \leq \delta \qquad \text{for all } t \in [t_0, t_0 + \tau]$$

where x denotes the component's input, $M(x)$ denotes the component's output and state trajectory, t_0 is the time of last attestation, and the probability is taken over the component's internal stochasticity (sampling, dropout, numerical noise) and input distribution.

Definition 3.3 (Contract Compatibility)**.** *Two contracts $\mathcal{C}_1 = (A_1, G_1, \delta_1, \tau_1)$ and $\mathcal{C}_2 = (A_2, G_2, \delta_2, \tau_2)$ are **compatible** for sequential composition if and only if:*

$$G_1 \Rightarrow A_2$$

*That is, the first component's guarantees logically imply the second component's assumptions. When $G_1 \Rightarrow A_2$ does not hold universally but holds with probability at least $1 - \epsilon$, the contracts are **ϵ-compatible**, and the compatibility gap ϵ contributes to the composed system's safety budget (Section 3.3).*

The contract framework differs from classical assume-guarantee reasoning in three respects specific to AI systems:

Stochastic guarantees. Classical contracts provide deterministic guarantees: if the assumption holds, the guarantee holds. AI components are inherently stochastic— sampling temperature, dropout masks, and floating-point nondeterminism mean that even identical inputs can produce different outputs. The violation probability δ captures this irreducible stochasticity. The algebra must track how δ values compose.

Distributional assumptions. Classical assumptions constrain individual inputs. AI safety properties are distributional: an entropy floor holds over a distribution of inputs, not for any single input. The assumption set A must therefore be understood as a predicate over distributions, not individual elements. This has consequences for composition: when Component 1's output distribution serves as Component 2's input distribution, the guarantee-to-assumption matching is a distributional implication, not a pointwise one.

Temporal validity. Classical contracts are timeless within their scope. AI model health invariants drift: distribution shift accumulates, hardware degrades, context caches grow stale. The validity horizon τ captures this temporal dimension. The composed system's validity horizon is at most $\min(\tau_1, \tau_2, \ldots, \tau_k)$—the composition is only as fresh as its stalest component.

3.2 Composition Operators

CRSA-1 defines four composition operators. Sequential and parallel composition formalize well-understood patterns. Conditional and recursive composition are novel contributions that address routing and agentic architectures respectively.

3.2.1 Sequential Composition ($\otimes_{\text{seq}}$)

Sequential composition models pipelines where one component's output feeds the next component's input.

Definition 3.4 (Sequential Composition)**.** *Given components $M_1 \models \mathcal{C}_1 = (A_1, G_1, \delta_1, \tau_1)$ and $M_2 \models \mathcal{C}_2 = (A_2, G_2, \delta_2, \tau_2)$ with ϵ-compatibility (i.e., $\Pr[G_1 \Rightarrow A_2] \geq 1 - \epsilon$), the sequential composition is:*

$$\mathcal{C}_1 \otimes_{\text{seq}} \mathcal{C}_2 \;=\; (A_1,\; G_2,\; \delta_1 + \delta_2 + \epsilon - \delta_1(\delta_2 + \epsilon),\; \min(\tau_1, \tau_2))$$

Theorem 3.1 (Sequential Composition Soundness)**.** *If $M_1 \models \mathcal{C}_1$ and $M_2 \models \mathcal{C}_2$ with ϵ-compatibility, then the composed system $M_2 \circ M_1$ satisfies $\mathcal{C}_1 \otimes_{\text{seq}} \mathcal{C}_2$.*

Proof sketch. By the union bound and conditional probability:

$$\begin{aligned}\Pr[\neg G_2 \mid A_1] &= \Pr[\neg G_2 \mid A_2, A_1]\,\Pr[A_2 \mid A_1] + \Pr[\neg G_2 \mid \neg A_2, A_1]\,\Pr[\neg A_2 \mid A_1] \\ &\leq \delta_2 \cdot 1 + 1 \cdot \Pr[\neg A_2 \mid A_1]\end{aligned}$$

Now $\Pr[\neg A_2 \mid A_1] \leq \Pr[\neg G_1 \mid A_1] + \Pr[\neg A_2 \mid G_1, A_1] \leq \delta_1 + \epsilon$. Substituting:

$$\Pr[\neg G_2 \mid A_1] \leq \delta_2 + \delta_1 + \epsilon - \delta_1(\delta_2 + \epsilon)$$

The subtracted term $\delta_1(\delta_2 + \epsilon)$ avoids double-counting the joint failure event. The validity horizon is $\min(\tau_1, \tau_2)$ because the composition is valid only while both component contracts remain valid. □

Remark 3.1. For k-stage pipelines, sequential composition is associative: $\mathcal{C}_1 \otimes_{\text{seq}} \mathcal{C}_2 \otimes_{\text{seq}} \cdots \otimes_{\text{seq}} \mathcal{C}_k$. The composed violation probability satisfies:

$$\delta_{\text{composed}} \;\leq\; 1 - \prod_{i=1}^{k}(1 - \delta_i) \cdot \prod_{i=1}^{k-1}(1 - \epsilon_i)$$

where ϵ_i is the compatibility gap at the i-th interface. For small δ_i, ϵ_i, this is approximately $\sum_{i=1}^{k} \delta_i + \sum_{i=1}^{k-1} \epsilon_i$—linear degradation.

3.2.2 Parallel Composition ($\otimes_{\text{par}}$)

Parallel composition models ensembles and concurrent model invocations whose outputs are aggregated.

Definition 3.5 (Parallel Composition)**.** *Given components $M_1 \models \mathcal{C}_1, \ldots, M_n \models \mathcal{C}_n$ with a deterministic aggregation function $\mathcal{A} : \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_n \to \mathcal{Y}$, the parallel composition is:*

$$\bigotimes_{\text{par}}^{\mathcal{A}} \{\mathcal{C}_i\}_{i=1}^{n} \;=\; (\textstyle\bigwedge_{i=1}^{n} A_i,\; G_{\mathcal{A}},\; \delta_{\mathcal{A}},\; \min_i(\tau_i))$$

where $G_{\mathcal{A}}$ is the guarantee predicate of the aggregated output and $\delta_{\mathcal{A}}$ depends on the aggregation function's properties.

The composed violation probability $\delta_{\mathcal{A}}$ depends on the aggregation function's *fault tolerance*:

Definition 3.6 (Aggregation Fault Tolerance)**.** *An aggregation function $\mathcal{A}$ is f-**fault-tolerant** with respect to guarantee G if the aggregated output satisfies G whenever at most f of the n component outputs violate their individual guarantees.*

For an f-fault-tolerant aggregation:

$$\delta_{\mathcal{A}} \;\leq\; \sum_{j=f+1}^{n} \binom{n}{j} \bar{\delta}^{\,j} (1-\bar{\delta})^{n-j}$$

where $\bar{\delta} = \max_i(\delta_i)$ is a conservative bound using the worst-case component violation probability. For majority voting with $n = 2m+1$ components, the aggregation is m-fault-tolerant, and the composed violation probability decreases exponentially in n when each $\delta_i < 1/2$—the classical redundancy gain.

Theorem 3.2 (Parallel Composition Soundness)**.** *If each $M_i \models \mathcal{C}_i$ and $\mathcal{A}$ is f-fault-tolerant, then the composed system $\mathcal{A}(M_1, \ldots, M_n)$ satisfies the parallel composition contract.*

Proof sketch. The event that more than f components violate their guarantees is bounded by the Chernoff tail of n independent Bernoulli trials with parameter $\bar{\delta}$. When A_i holds for all i (guaranteed by the conjunction $\bigwedge A_i$), each component violates its guarantee with probability at most $\delta_i \leq \bar{\delta}$. The aggregated output violates $G_{\mathcal{A}}$ only if more than f components fail, which occurs with the stated probability. Independence is assumed; shared infrastructure correlations are addressed in Section 5. □

Remark 3.2. When components share infrastructure, their failures are correlated. The independence assumption in Theorem 3.2 is optimistic. Section 5 introduces the cross-correlation correction factor ρ_{ij} that adjusts $\delta_{\mathcal{A}}$ for correlated failures due to shared GPU memory, KV cache, or model weights.

3.2.3 Conditional Composition ($\otimes_{\text{cond}}$)

Conditional composition models routing architectures where a classifier directs inputs to one of several downstream components. This operator is a novel contribution; no prior assume-guarantee framework defines conditional composition for stochastic AI components.

Definition 3.7 (Conditional Composition)**.** *Given a routing function $R : \mathcal{X} \to \{1, \ldots, n\}$ with per-class misrouting probabilities $\mu_i = \Pr[R(x) \neq i \mid x \in \mathcal{X}_i]$, and downstream components $M_i \models \mathcal{C}_i = (A_i, G_i, \delta_i, \tau_i)$ for $i = 1, \ldots, n$ where $\mathcal{X}_i$ is the input subpopulation that **SHOULD** be routed to M_i, the conditional composition is:*

$$\otimes^{R}_{\text{cond}}\{\mathcal{C}_i\}_{i=1}^{n} \;=\; \left(A_R \wedge \textstyle\bigwedge_{i=1}^{n} A_i,\; G_{\text{cond}},\; \delta_{\text{cond}},\; \min(\tau_R, \min_i \tau_i)\right)$$

where A_R is the router's input assumption, τ_R is the router's contract validity horizon, and:

$$\delta_{\text{cond}} \;=\; \delta_R + \sum_{i=1}^{n} p_i \Big[(1-\mu_i)\,\delta_i \;+\; \mu_i \cdot \max_{j \neq i}(\delta_j + \epsilon_{i \to j})\Big]$$

Here $p_i = \Pr[x \in \mathcal{X}_i]$ is the population prior for class i, δ_R is the router's own violation probability (failure to produce a valid routing decision), and $\epsilon_{i \to j}$ is the assumption-guarantee mismatch when class-i inputs are misrouted to component M_j.

Theorem 3.3 (Conditional Composition Soundness)**.** *If the router satisfies its contract, each downstream component satisfies its contract, and the misrouting probabilities μ_i and mismatch penalties $\epsilon_{i \to j}$ are correctly bounded, then the composed system satisfies the conditional composition contract.*

Proof sketch. Partition the input space by routing outcome. For correctly routed inputs ($R(x) = i$ when $x \in \mathcal{X}_i$), the violation probability is δ_i by component contract satisfaction. For misrouted inputs ($R(x) = j$ when $x \in \mathcal{X}_i$, $j \neq i$), the violation probability is at most $\delta_j + \epsilon_{i \to j}$: the component's own violation probability plus the assumption mismatch. Weighting by population priors and routing accuracy, and adding the router's own failure probability δ_R, yields the stated bound. □

The $\epsilon_{i \to j}$ term is the key insight: it captures the *semantic cost of misrouting*. When a hard query is routed to an easy-query model, the assumption mismatch $\epsilon_{\text{hard} \to \text{easy}}$ is large because the easy-query model's assumptions (input complexity below threshold) are violated. When an easy query is routed to a hard-query model, $\epsilon_{\text{easy} \to \text{hard}}$ is typically small (the hard-query model's assumptions are a superset of the easy-query conditions). This asymmetry formalizes the intuition that routing hard queries to weak models is dangerous while routing easy queries to strong models is merely wasteful.

3.2.4 Recursive Composition ($\otimes_{\text{rec}}$)

Recursive composition models agentic loops: a component whose output is fed back as input (possibly through other components) for iterative refinement. This operator is a novel contribution; it is the first formal treatment of recursive safety certificate composition for AI agent loops.

Definition 3.8 (Recursive Composition)**.** *Given a component $M \models \mathcal{C} = (A, G, \delta, \tau)$ used in a loop of at most K iterations, where each iteration's output is transformed by a feedback function $F : \mathcal{Y} \to \mathcal{X}$ before being used as the next iteration's input, the recursive composition is:*

$$\otimes_{\text{rec}}^{K,F} \mathcal{C} \;=\; (A,\; G_{\text{rec}},\; \delta_{\text{rec}}(K),\; \min(\tau, \tau/K))$$

*The composed violation probability depends on the **feedback stability** of F:*

1. *If F is **assumption-preserving** ($G \Rightarrow A \circ F$ with certainty), then iterations are independent and:*
$$\delta_{\text{rec}}(K) = 1 - (1 - \delta)^K \approx K\delta \quad \text{for small } \delta$$
2. *If F is γ**-contractive** ($\Pr[\neg(A \circ F) \mid G] \leq \gamma$ and the guarantee degradation per iteration is bounded by contraction rate $\gamma < 1$), then:*
$$\delta_{\text{rec}}(K) \leq \delta \cdot \frac{1 - \gamma^K}{1 - \gamma}$$
3. *If F is **neither assumption-preserving nor contractive**, then the composition is unbounded and CRSA-1 **MUST NOT** issue a system-level certificate. The system **MUST** be restructured to achieve one of the two stable cases, or the recursive loop **MUST** be broken into sequential stages with explicit re-attestation.*

Theorem 3.4 (Recursive Composition Soundness)**.** *Under the assumption-preserving case, if $M \models \mathcal{C}$ and $G \Rightarrow A \circ F$, then the K-iteration loop satisfies the recursive composition contract.*

Proof sketch. By induction on iteration count. Base case ($K = 1$): the single invocation satisfies $\mathcal{C}$ by hypothesis. Inductive step: assume the first k iterations have maintained the guarantee with probability at least $(1 - \delta)^k$. The $(k+1)$-th iteration receives input $F(y_k)$ where y_k satisfies G (by inductive hypothesis with appropriate probability). Since $G \Rightarrow A \circ F$, the input to iteration $k + 1$ satisfies A, and the component contract gives $\Pr[\neg G] \leq \delta$. The probability that all K iterations maintain the guarantee is $(1 - \delta)^K$, giving violation probability $1 - (1 - \delta)^K$. □

Remark 3.3 (Iteration Budget). The recursive composition operator makes explicit the trade-off between iteration depth and safety. For a target system-level violation probability δ_{target}, the maximum permissible iteration count is:

$$K_{\max} = \left\lfloor \frac{\ln(1 - \delta_{\text{target}})}{\ln(1 - \delta)} \right\rfloor \approx \frac{\delta_{\text{target}}}{\delta} \quad \text{for small } \delta$$

A component with per-invocation violation probability $\delta = 10^{-4}$ and target system-level violation probability $\delta_{\text{target}} = 10^{-2}$ may iterate at most $K_{\max} \approx 100$ times. This is the **iteration budget**—a formally derived bound on agent loop depth that CRSA-1 enforces at runtime.

3.3 The Safety Budget

Composition degrades safety. The safety budget provides a unified accounting framework for this degradation.

Definition 3.9 (Safety Budget)**.** *A **safety budget** is a target system-level violation probability $\Delta \in (0, 1)$ allocated across composition stages. For a system with k composition operations, each contributing violation probability δ_i^* (the composed violation from the relevant operator), the safety budget constraint is:*

$$1 - \prod_{i=1}^{k} (1 - \delta_i^*) \;\leq\; \Delta$$

or equivalently, $\sum_{i=1}^{k} \delta_i^* \leq \Delta$ for small δ_i^*.

The safety budget is analogous to the privacy budget in differential privacy. Just as a privacy budget ε is "spent" by each query to a database and cannot be replenished without fresh randomization, a safety budget Δ is spent by each composition operation and cannot be replenished without re-attestation of the component contracts.

Definition 3.10 (Budget Allocation)**.** *A **budget allocation** for a composition graph $\mathcal{G} = (V, E)$, where vertices are components and edges are composition operations, is an assignment $\{\delta_i^{\text{alloc}}\}_{i \in V}$ such that:*

1. *Each component's allocated budget is at least its contract's violation probability: $\delta_i^{\text{alloc}} \geq \delta_i$.*
2. *Each composition operation's result satisfies: $\delta_{\text{op}}^* \leq \delta_i^{\text{alloc}}$ for the appropriate operator.*
3. *The total budget satisfies: $1 - \prod_i (1 - \delta_i^{\text{alloc}}) \leq \Delta$.*

CRSA-REQ-BUDGET

Every CRSA-1 compliant system **SHALL** declare a safety budget Δ in its system-level attestation token. The runtime attestation mechanism (Section 4) **SHALL** track cumulative budget consumption across composition operations and **SHALL** emit a `BUDGET_EXCEEDED` event when cumulative violation probability exceeds Δ. Downstream components **MUST NOT** accept inputs from pipelines whose budget is exceeded.

3.4 Recovery of MAI-1 Invariants as Composable Contracts

The five mandatory MAI-1 invariants map to safety contracts as follows. This mapping establishes that CRSA-1 is a strict extension of MAI-1: every MAI-1 attestation artifact becomes a composable contract within the CRSA-1 algebra.

Table 1: MAI-1 Invariant Recovery as Safety Contracts

MAI-1 Invariant	**Assumption** A	**Guarantee** G	**Typical** δ
AI-8: Entropy Floor	Input distribution within calibration domain; temperature $\in [\tau_{\min}, \tau_{\max}]$	Output entropy $H(Y\|X) \geq H_{\min}$ over sliding window	10^{-4}
AI-2: Gradient Stability	Batch statistics within two standard deviations of calibration baseline	Gradient norm $\|\nabla\mathcal{L}\| \leq \beta_{\max}$; no layer starvation	10^{-3}
AI-3: Lyapunov Stability	Input perturbation $\|\Delta x\| \leq \epsilon_{\max}$	Output deviation $\|f(x+\Delta x)-f(x)\| \leq L \cdot \|\Delta x\|$ with Lyapunov certificate	10^{-3}
AI-6: Distribution Drift	Reference distribution P_{ref} from calibration epoch	$D_{\text{KL}}(P_{\text{current}}\|P_{\text{ref}}) \leq d_{\max}$	10^{-3}
AI-4: Thermal Integrity	Ambient temperature $\leq T_{\text{rated}}$; cooling system operational	SRAM bit-flip rate $\leq \phi_{\max}$; no thermal throttling during inference	10^{-6}

The contract mapping reveals an important structural property: **different invariants have different composition behaviors.**

The entropy floor (AI-8) composes well under sequential composition: if each component maintains its entropy floor, the pipeline maintains entropy diversity (barring pathological distribution collapse at an interface, captured by ϵ). It composes poorly under recursive composition: repeated application of a high-temperature model to its own output can drive entropy to maximum, losing signal.

Gradient stability (AI-2) is a training-time invariant that does not directly compose at inference time. Its relevance to CRSA-1 is indirect: components whose gradient stability has been verified during training are less likely to exhibit inference-time instabilities. The contract validity horizon τ for gradient stability is therefore long (typically keyed to retraining or fine-tuning cycles).

The Lyapunov stability envelope (AI-3) has the strongest composition properties: if each component is Lipschitz-bounded with constant L_i, the sequential composition is Lipschitz-bounded with constant $\prod_i L_i$. This is the small-gain condition from control theory. For the composed system to remain stable, the product of individual Lipschitz constants must remain bounded—$\prod_i L_i < L_{\max}$—which places a constraint on pipeline depth for a given component stability level.

Distribution drift (AI-6) composes additively under sequential composition by the triangle inequality on KL divergence:

$$D_{\text{KL}}(P_k\|P_{\text{ref}}) \leq \sum_{i=1}^{k} D_{\text{KL}}(P_i\|P_{i-1})$$

where P_i is the distribution after stage i. Each stage's drift bound contributes linearly to the pipeline's total drift from the reference distribution.

Thermal integrity (AI-4) composes under a shared-infrastructure model: if multiple models share a GPU, thermal integrity is a *system-level* property, not a per-model property. The thermal

envelope must accommodate the aggregate compute load of all co-resident models. CRSA-1 delegates thermal attestation to the platform layer (Layer 1) and treats it as a shared assumption across all components on the same hardware.

3.5 What Does Not Compose

The honest framing requires explicit acknowledgment of composition limits.

Behavioral alignment does not compose. A model that is individually aligned (helpful, harmless, honest) may participate in a composed system that produces unaligned outputs. The composition operators track safety *certificates*—attestation that verified properties hold—not safety *values*. If the verified properties do not capture the relevant alignment dimension, the composed certificate is valid but alignment-irrelevant.

Emergent capabilities do not compose. If the composed system exhibits a capability that no component possesses individually, no compositional analysis can predict it. The safety budget provides a probabilistic bound on *known* failure modes; it provides no bound on unknown capabilities that emerge from interaction. This is the fundamental incompleteness of compositional verification: it is sound (what it verifies is correct) but incomplete (there exist properties it cannot verify).

Correlated failures do not compose independently. The parallel composition operator (Theorem 3.2) assumes independent component failures. When components share training data, architecture, or infrastructure, their failures are correlated. Section 5 provides correction factors, but worst-case correlation (common-mode failure) can negate the redundancy benefit of ensembles entirely.

Stochastic certificates degrade under composition. Every composition operation spends safety budget. Deep pipelines, long agent loops, and wide ensembles all accumulate violation probability. There is no free composition—every additional component or iteration costs safety margin. The safety budget makes this cost explicit and enforceable.

4 Runtime Attestation Mechanism

The safety certificate algebra (Section 3) defines *what* composes. This section defines *how* composition is attested at runtime—the cryptographic machinery that produces verifiable evidence of compositional safety at each inference step.

CRSA-1's runtime attestation extends the IETF Remote Attestation Procedures (RATS) architecture (RFC 9334) and the Entity Attestation Token (EAT) format (RFC 9711) from single-platform attestation to multi-model pipeline attestation. The design achieves less than 0.1% latency overhead on typical LLM inference by using lightweight cryptographic commitments (hash chains and Merkle trees over CBOR-encoded tokens) rather than heavyweight proof systems.

Section 4.1 defines the CRSA-1 EAT profile. Section 4.2 specifies the per-inference commitment protocol. Section 4.3 defines pipeline attestation chaining. Section 4.4 describes TEE anchoring. Section 4.5 identifies integration points with production inference engines.

4.1 CRSA-1 EAT Profile

CRSA-1 defines an EAT profile that extends the emerging AI-specific EAT claims landscape. Three IETF Internet-Drafts are converging on AI model attestation within the EAT framework: draft-messous-eat-ai-00 (AI model identity and provenance claims), draft-jiang-seat-dynamic-attestation-00 (runtime posture and dynamic state claims), and draft-huang-rats-agentic-eat-cap-attest-00 (agent capability and delegation claims). CRSA-1 builds on all three while adding the composition-specific claims that none of them address.

Definition 4.1 (CRSA-1 Attestation Token)**.** *A **CRSA-1 attestation token** is an EAT (RFC 9711) encoded as a CWT (CBOR Web Token, RFC 8392) containing the following claim sets:*

1. ***Platform claims** (from Layer 1): Hardware identity, TEE evidence, firmware measurements, thermal state—inherited from the MAI-1 platform attestation.*
2. ***Component claims** (from Layer 2): The MAI-1 invariant values for the attesting component at the time of inference—entropy floor measurement, gradient stability indicator, Lyapunov certificate status, drift distance, thermal integrity.*
3. ***Composition claims** (CRSA-1 novel): The component's safety contract $\mathcal{C} = (A, G, \delta, \tau)$ encoded as structured claims, the composition operator applied (sequential, parallel, conditional, recursive), the cumulative safety budget consumed by the pipeline up to and including this component, and the pipeline position index.*
4. ***Provenance claims** (from Layer 3): Model identity (AI-BOM hash), training provenance reference, and lineage chain linking the current model version to its certified baseline.*
5. ***Freshness claims**: Nonce (from verifier challenge or epoch-based), timestamp, and sequence number within the current pipeline invocation.*

The token is signed using the component's attestation key, which is bound to the hardware platform through the TEE attestation chain (Section 4.4). The signing algorithm **SHALL** be COSE_Sign1 (RFC 9052) using ECDSA-P256 or EdDSA (Ed25519) for single-signer tokens. Multi-signer scenarios (e.g., co-attestation by the model operator and the platform provider) **SHALL** use COSE_Sign (RFC 9052) with independent signatures.

CRSA-REQ-TOKEN-SIZE

The CRSA-1 attestation token **SHALL** be encodable in at most 2048 bytes of CBOR. This constraint ensures that token generation, transmission, and verification remain within the latency budget defined in Section 4.2. Token size is dominated by the signature (64–80 bytes), platform evidence reference (32-byte hash rather than full evidence), and composition claims (variable, bounded by pipeline depth).

4.2 Per-Inference Commitment Protocol

Every inference step in a CRSA-1 compliant system produces a cryptographic commitment binding the model's input, output, internal state measurements, and safety contract status into a single, tamper-evident record.

Definition 4.2 (Inference Commitment)**.** *An **inference commitment** for inference step t at component M_i is:*

$$c_{i,t} = \mathrm{H}(\mathrm{H}(x_{i,t}) \parallel \mathrm{H}(y_{i,t}) \parallel \mathbf{s}_{i,t} \parallel \delta_{i,t}^{\text{budget}} \parallel c_{i,t-1} \parallel \text{nonce}_t)$$

where H *is SHA-256, $x_{i,t}$ and $y_{i,t}$ are the input and output, $\mathbf{s}_{i,t}$ is the vector of MAI-1 invariant measurements, $\delta_{i,t}^{\text{budget}}$ is the cumulative safety budget consumed, $c_{i,t-1}$ is the previous commitment (forming a hash chain), and nonce_t is the freshness nonce.*

The hash chain $c_{i,0} \to c_{i,1} \to \cdots \to c_{i,t}$ provides append-only integrity: any modification to a past commitment invalidates all subsequent commitments. The commitment includes hashes of inputs and outputs (not the inputs and outputs themselves) to avoid storing sensitive data in the attestation layer while maintaining verifiability—a verifier with access to the original inputs and outputs can recompute the hashes and verify the chain.

CRSA-REQ-LATENCY

The per-inference commitment **SHALL** add no more than 100 microseconds of latency to each model invocation. This budget covers: invariant measurement extraction (≤ 20 μs for pre-computed statistics), hash computation (≤ 5 μs for SHA-256 over the commitment structure), token assembly (≤ 15 μs for CBOR encoding), and signature generation (≤ 60 μs for ECDSA-P256 or ≤ 40 μs for Ed25519).

Remark 4.1. For context, a single forward pass of a 70B-parameter model at batch size 1 on an NVIDIA H100 requires approximately 35–70 milliseconds. A 100-microsecond attestation overhead represents 0.14–0.29% of inference latency—well below the 0.1%–1% range that production systems tolerate for observability instrumentation (OpenTelemetry traces, Prometheus metrics). The attestation overhead is comparable to a single kernel launch or memory allocation event.

4.3 Pipeline Attestation Chaining

When multiple components are composed, their individual attestation tokens must be linked into a pipeline attestation structure that captures both the individual component states and the composition relationships between them.

Definition 4.3 (Pipeline Attestation DAG)**.** *A **pipeline attestation DAG** is a directed acyclic graph $\mathcal{P} = (V, E, r)$ where:*

- *Each vertex $v \in V$ is a CRSA-1 attestation token from a component invocation.*
- *Each directed edge $(v_i, v_j) \in E$ represents a data flow: the output of component i was consumed as input by component j.*
- *The root r is the system-level attestation token that aggregates the pipeline's composite safety state.*

The pipeline attestation DAG is committed using a Merkle tree:

Definition 4.4 (Pipeline Merkle Commitment)**.** *The **pipeline Merkle commitment** for a pipeline attestation DAG $\mathcal{P}$ is:*

$$\text{root}(\mathcal{P}) = \text{MerkleRoot}\big(\{c_{i,t}\}_{(i,t)\in V}\big)$$

where the Merkle tree is constructed over the ordered set of inference commitments, with ordering defined by topological sort of the DAG. The Merkle root is included in the system-level attestation token at the DAG root r.

The Merkle structure provides two critical properties:

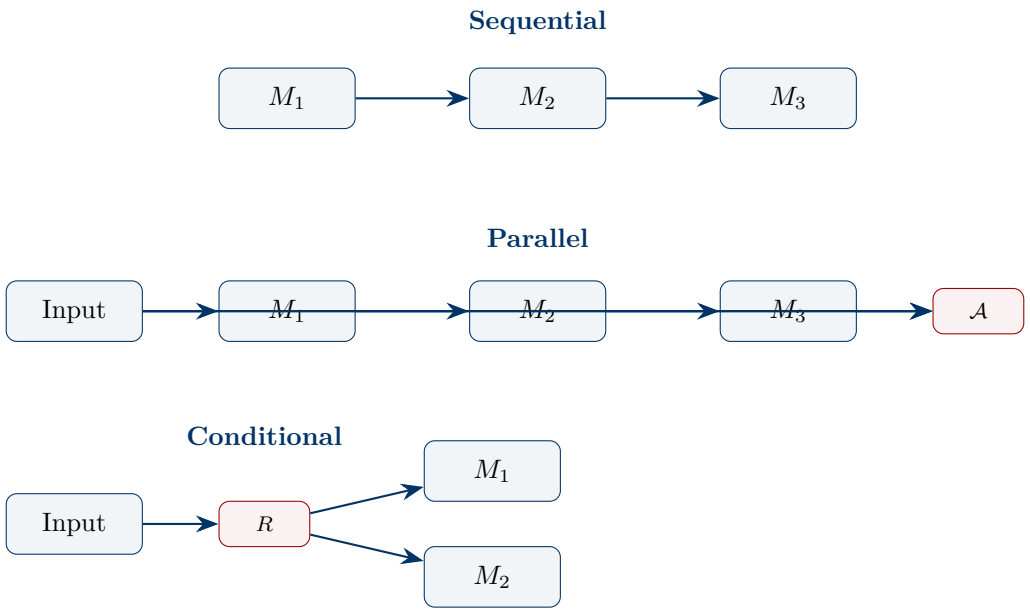
Selective disclosure. A verifier can request proof that a specific component in the pipeline was attested without receiving the attestation details of other components. The Merkle inclusion proof for component i's commitment requires $O(\log |V|)$ hashes— logarithmic in the pipeline depth. This enables privacy-preserving verification where different stakeholders verify different components: the model provider verifies model health, the platform provider verifies hardware integrity, and the system integrator verifies composition correctness.

Tamper evidence. Any modification to any component's attestation invalidates the Merkle root, which is signed in the system-level token. An attacker who compromises one component in the pipeline cannot forge a valid system-level attestation without also forging the Merkle root, which requires either breaking the hash function or possessing the system-level signing key.

CRSA-REQ-DAG-STRUCTURE

The pipeline attestation DAG **SHALL** preserve the composition operator structure from Section 3. Sequential compositions produce linear chains. Parallel compositions produce fan-out/fan-in structures with the aggregation function at the fan-in vertex. Conditional compositions produce branches with the router at the branch vertex. Recursive compositions produce annotated back-edges with iteration counters. The DAG structure **SHALL** be sufficient to reconstruct the composition algebra expression that was applied, enabling a verifier to independently compute the expected system-level safety budget from the component-level contracts.

For each composition pattern, the DAG takes a characteristic form:



4.4 TEE Anchoring

The attestation chain must terminate in a hardware root of trust. Without TEE anchoring, the attestation tokens are software artifacts that a compromised system can forge. With TEE anchoring, forging an attestation token requires compromising the TEE's isolation guarantees—a qualitatively harder attack.

CRSA-1 does not define a new TEE attestation protocol. It specifies how existing TEE attestation (as defined in the Auburn Governance Stack's Layer 1 specifications, particularly AI-4 SRAM Thermal Integrity Bound and the GPU TEE Profile) anchors the CRSA-1 attestation chain.

CRSA-REQ-TEE-ANCHOR

At CRSA-1 conformance level C2 (Attested) and above, every component's attestation signing key **SHALL** be bound to a TEE attestation report. The binding **SHALL** be established through one of:

1. **TEE-resident key generation:** The attestation signing key is generated within the TEE enclave and the public key is included in the TEE's attestation report. This provides the strongest binding: the key never exists outside the TEE.
2. **TEE-wrapped key import:** The attestation signing key is generated externally, imported into the TEE under a wrapping key derived from the TEE's sealing identity, and the import certificate is included in the TEE's attestation report.
3. **TEE-attested key association:** The attestation signing key exists outside the TEE but is associated with the TEE identity through a certificate chain rooted at the TEE vendor's root of trust. This provides the weakest binding and is acceptable only at conformance level C2, not C3.

Honest Framing

TEE attestation proves that code is running in a specific hardware-isolated environment. It does not prove that the code behaves correctly, that the model is safe, or that the inference results are trustworthy. The Stateful Isolation Law (Document 5) catalogs 25+ demonstrated side-channel attacks on GPU TEEs, and AI-4 establishes that physical-layer attacks on SRAM thermal integrity can bypass TEE protections entirely.
CRSA-1's TEE anchoring provides *platform identity assurance*—confidence that the attestation token was produced by the claimed hardware platform. It does not provide behavioral assurance. The behavioral assurance comes from the Layer 2 invariants (AI-2 through AI-8) and the composition algebra (Section 3). TEE anchoring prevents forgery; invariant monitoring prevents undetected degradation. Both are necessary; neither is sufficient alone.

The TEE trust chain for a CRSA-1 pipeline attestation proceeds as follows:

1. The hardware platform produces a TEE attestation report binding the platform identity, firmware measurements, and attestation public key (Layer 1).
2. Each model component produces a MAI-1 attestation token signed with the TEE-bound key, containing Layer 2 invariant measurements and Layer 3 provenance claims (existing MAI-1 flow).
3. The CRSA-1 composition layer collects component tokens, applies the composition algebra, computes the pipeline Merkle commitment, and produces the system-level CRSA-1 attestation token (novel CRSA-1 contribution).
4. A relying party verifies the chain: TEE report → component tokens → pipeline DAG → system-level certificate, checking that the composition algebra was correctly applied and the safety budget is not exceeded.

4.5 Integration Points

CRSA-1 attestation integrates with production inference infrastructure through hook points at the inference engine layer. This section identifies the conceptual integration architecture; implementation specifications are held under the Auburn Patent Family.

Three integration patterns cover the major inference engine architectures:

Plugin-based integration targets inference engines with explicit plugin or extension APIs. The CRSA-1 attestation logic is packaged as a plugin that intercepts the inference pipeline at model load (contract registration), pre-inference (assumption verification, nonce generation), post-inference (invariant measurement, commitment generation, token assembly), and pipeline completion (DAG construction, Merkle commitment, system-level token). vLLM's plugin architecture and Triton Inference Server's custom backend API support this pattern.

Wrapper-based integration targets inference engines without plugin APIs. A CRSA-1 wrapper process sits between the orchestrator and the inference engine, intercepting requests and responses to inject attestation. The wrapper handles contract verification, commitment generation, and token assembly without modifying the inference engine. This pattern applies to TensorRT-LLM and proprietary inference engines. Latency overhead is slightly higher (additional IPC) but remains within the 100-microsecond budget for hash-chain commitments.

Sidecar-based integration targets containerized deployments (Kubernetes). A CRSA-1 sidecar container shares the pod's network namespace, receives copies of inference requests and responses through a shared volume or localhost proxy, and produces attestation tokens asynchronously. This pattern decouples attestation from the inference critical path entirely, eliminating latency overhead at the cost of eventual (rather than synchronous) attestation. Sidecar integration is acceptable at conformance level C1 (Composed) but not at C2 (Attested) or above, which require synchronous per-inference commitment.

CRSA-REQ-INTEGRATION

CRSA-1 compliant implementations **SHALL** support at least one of the three integration patterns. At conformance level C2 and above, the implementation **SHALL** use plugin-based or wrapper-based integration to ensure synchronous per-inference commitment. Sidecar-based integration **MAY** be used at conformance level C1. Regardless of integration pattern, the implementation **MUST NOT** modify model weights, inference parameters, or output tokens—the attestation layer is strictly observational with respect to the inference computation.

CRSA-REQ-OBSERVATIONAL

The CRSA-1 attestation mechanism **SHALL** be **observational**: it measures, commits, and reports, but does not modify the inference computation. This is a fundamental design principle. The attestation layer has no authority to alter model behavior—that authority belongs to the model operator and the safety systems (RLHF, constitutional AI, safety filters) that the model was deployed with. CRSA-1 attests *what happened*; it does not control *what happens*. Enforcement—blocking outputs, triggering fallbacks, alerting operators—is the responsibility of the policy layer that consumes CRSA-1 attestation tokens, not the attestation mechanism itself.

5 Multi-Principal Isolation Bounds

Single-model attestation assumes a single principal: one model, one operator, one trust domain. Multi-model systems violate this assumption by construction. A RAG pipeline may compose a retriever operated by Principal A with a generator operated by Principal B on infrastructure owned by Principal C. An agentic system may invoke tools across trust boundaries where the planner, executor, and tool providers are distinct entities with distinct security postures. Ensemble systems may aggregate models from competing providers who have contractual obligations not to share proprietary information.

The question CRSA-1 must answer is: *when models share infrastructure, how much does one principal's computation influence another principal's outputs?* The Stateful Isolation Law (Document 5) answers this question for a single model on shared infrastructure. This section extends that answer to the multi-model case where contamination flows through both explicit composition channels (the intended data flow) and implicit infrastructure channels (the unintended side channels).

Section 5.1 reviews the Stateful Isolation Law's contamination functional. Section 5.2 extends it to multi-model systems. Section 5.3 derives the multi-model small gain condition. Section 5.4 defines practical isolation mechanisms and their formal properties.

5.1 The Contamination Functional: Single-Model Review

The Stateful Isolation Law defines a contamination functional that quantifies cross-principal influence in shared inference infrastructure. The essential structure is reviewed here to establish notation; the full development is in Document 5.

Definition 5.1 (Contamination Functional — Single Model (Stateful Isolation Law))**.** *For a model M serving principals α and β on shared infrastructure $\mathcal{I}$, the* ***contamination functional*** *is:*

$$C_\alpha(\beta, S_t, \mathcal{I}) \;=\; \sup_{x_\beta} \; d\big(M(x_\alpha \mid S_t),\; M(x_\alpha \mid S_t')\big)$$

where S_t is the infrastructure state after β's computation, S_t' is the counterfactual infrastructure state had β not computed, $d(\cdot,\cdot)$ is a distributional distance (total variation or KL divergence), and the supremum is taken over all possible inputs x_β from the contaminating principal.

The contamination functional measures the worst-case influence: over all possible adversarial inputs from β, how much does β's computation shift α's output distribution? The Stateful Isolation Law decomposes this functional across five channels: shared weights (C^{wt}), shared KV cache (C^{kv}), shared GPU memory (C^{gpu}), shared system prompt (C^{sp}), and continuous batching (C^{cb}). The total contamination is bounded by:

$$C_\alpha(\beta, S_t, \mathcal{I}) \;\le\; C^{\text{wt}} + C^{\text{kv}} + C^{\text{gpu}} + C^{\text{sp}} + C^{\text{cb}}$$

The Stateful Isolation Law further provides a Composability Theorem establishing that sequential, parallel, and cross-layer contamination compose with known bounds. CRSA-1 extends these composition results to the multi-model case.

5.2 Extension to Multi-Model Systems

In a multi-model pipeline, contamination flows through two distinct channel classes: *explicit* channels (the intended data flow defined by the composition operators) and *implicit* channels (the unintended infrastructure channels cataloged by the Stateful Isolation Law). The multi-model contamination functional must account for both.

Definition 5.2 (Multi-Model Contamination Functional)**.** *For a pipeline of k models $M_1, \ldots, M_k$ serving principals $\alpha_1, \ldots, \alpha_m$ on shared infrastructure $\mathcal{I}$, the* ***multi-model contamination functional*** *for principal α due to principal β is:*

$$C_\alpha^{\text{multi}}(\beta, \mathbf{S}_t, \mathcal{I}, \mathcal{G}) \;=\; C_\alpha^{\text{explicit}}(\beta, \mathcal{G}) \;+\; C_\alpha^{\text{implicit}}(\beta, \mathbf{S}_t, \mathcal{I}) \;-\; C_\alpha^{\text{overlap}}$$

where:

- *$\mathcal{G}$ is the composition graph (pipeline DAG from Section 4.3).*
- *$\mathbf{S}_t = (S_t^{(1)}, \ldots, S_t^{(k)})$ is the vector of per-model infrastructure states.*

- $C_\alpha^{\text{explicit}}$ *is the contamination through intended data flows (composition edges in* $\mathcal{G}$*).*
- $C_\alpha^{\text{implicit}}$ *is the contamination through unintended infrastructure channels (shared hardware, memory, caches).*
- $C_\alpha^{\text{overlap}}$ *corrects for double-counting channels that are both explicit and implicit (e.g., shared KV cache entries that are also part of the intended context flow).*

The explicit contamination is determined by the composition graph topology:

Proposition 5.1 (Explicit Contamination Bound)**.** *For a sequential pipeline* $M_1 \to M_2 \to \cdots \to M_k$ *where principal* β *provides input to* M_1 *and principal* α *receives output from* M_k*, the explicit contamination is bounded by:*

$$C_\alpha^{\text{explicit}}(\beta, \mathcal{G}) \;\leq\; \prod_{i=1}^{k} L_i \cdot C_\alpha(\beta, S_t^{(1)}, \mathcal{I}_1)$$

where L_i *is the Lipschitz constant of model* M_i *(from the AI-3 Lyapunov stability certificate) and* $C_\alpha(\beta, S_t^{(1)}, \mathcal{I}_1)$ *is the single-model contamination functional at the entry point.*

This bound has a direct operational interpretation: contamination propagates through the pipeline, amplified by each component's sensitivity (Lipschitz constant). A pipeline of k models each with Lipschitz constant L amplifies entry-point contamination by L^k. For the composed system to maintain bounded contamination, the product $\prod L_i$ must remain bounded—precisely the small-gain condition formalized in Section 5.3.

The implicit contamination depends on the infrastructure sharing topology:

Proposition 5.2 (Implicit Contamination Bound)**.** *For models* M_i *and* M_j *sharing infrastructure component* $\mathcal{I}_s$ *(GPU, memory bus, L2 cache, or NVLink), the implicit contamination between the principals they serve is bounded by:*

$$C_\alpha^{\text{implicit}}(\beta, \mathbf{S}_t, \mathcal{I}_s) \;\leq\; \gamma_s \cdot B_s$$

where γ_s *is the channel capacity of the shared infrastructure component (in bits per inference step) and* B_s *is the sensitivity of model* M_j*'s output to* γ_s *bits of side-channel information.*

The channel capacities for known side channels, as cataloged in the Stateful Isolation Law, are:

Channel	**Measured Capacity**	**Source**
MIG TLB covert channel	31 kbps (99.8% accuracy)	CCS 2023
NVBleed (NVLink)	>70 kbps (cross-VM)	2025
L2 cache side channel	∼4 MB/s	Micro. arch. literature
PROMPTPEEK (KV cache timing)	Prompt recovery	NDSS 2025
Continuous batching timing	Sequence-length leakage	2024

The practical impact depends on B_s—how much a model's output changes given side-channel bits. For most LLM inference workloads, side-channel information at kbps rates provides negligible influence on output distributions: the model's internal stochasticity (sampling noise at typical temperatures) exceeds the information content of the side channel. However, for deterministic inference (temperature 0, greedy decoding), side-channel sensitivity increases because the output is a deterministic function of the input and model state, making any state perturbation potentially observable.

CRSA-REQ-IMPLICIT-BOUND

At CRSA-1 conformance level C2 and above, multi-model systems **SHALL** declare the infrastructure sharing topology and the resulting implicit contamination bound for each principal pair. At conformance level C3 (Sovereign), the implicit contamination bound **SHALL** be formally verified against the Stateful Isolation Law's channel catalog, and any channel with capacity exceeding 1 kbps between distinct-principal models **SHALL** be mitigated or disclosed.

5.3 Multi-Model Small Gain Condition

The small gain theorem from control theory provides the stability condition for interconnected systems: a feedback loop is stable if the product of the loop gains is less than one. CRSA-1 applies this principle to multi-model composition, where "gain" is the contamination amplification factor and "stability" is bounded cross-principal influence.

Definition 5.3 (Contamination Gain Matrix)**.** *For a system of k models, the* ***contamination gain matrix*** $\Gamma \in \mathbb{R}_{\geq 0}^{k \times k}$ *has entries:*

$$\Gamma_{ij} \;=\; L_i \cdot \mathbb{1}[(M_j, M_i) \in E(\mathcal{G})] \;+\; \gamma_{ij}^{\text{infra}} \cdot \mathbb{1}[M_i, M_j \text{ share infrastructure}]$$

where L_i is the Lipschitz constant of M_i, $\mathbb{1}[\cdot]$ is the indicator function, $E(\mathcal{G})$ is the edge set of the composition graph, and $\gamma_{ij}^{\text{infra}}$ is the infrastructure-channel contamination gain between M_j and M_i.

Theorem 5.3 (Multi-Model Small Gain Condition)**.** *The multi-model system has bounded contamination—i.e., no principal's influence on any other principal grows without bound under repeated pipeline invocations—if and only if:*

$$\rho(\Gamma) < 1$$

where $\rho(\Gamma)$ is the spectral radius (largest absolute eigenvalue) of the contamination gain matrix Γ.

Proof sketch. The contamination after n pipeline invocations is bounded by $\Gamma^n \mathbf{c}_0$ where $\mathbf{c}_0$ is the initial contamination vector. By the Gelfand formula, $\|\Gamma^n\| \leq C \cdot \rho(\Gamma)^n$ for some constant C. When $\rho(\Gamma) < 1$, this decays geometrically: contamination is transient. When $\rho(\Gamma) \geq 1$, contamination can grow without bound: the system is unstable with respect to cross-principal influence. This is a direct application of the standard small-gain theorem (Desoer and Vidyasagar, 1975; Jiang, Teel, and Praly, 1994) to the discrete-time contamination dynamics. □

Corollary 5.4 (Pipeline Depth Bound)**.** *For a homogeneous sequential pipeline (all components with Lipschitz constant L and negligible infrastructure contamination), the small gain condition reduces to:*

$$L^k < 1 \quad \Longleftrightarrow \quad k < \frac{-1}{\log L} \quad \textit{(only satisfiable when } L < 1\textit{)}$$

For $L \geq 1$ (expansive components), no finite sequential pipeline satisfies the small gain condition. For $L = 0.99$, the maximum pipeline depth for bounded contamination is $k < 100$. For $L = 0.95$, it is $k < 20$.

Remark 5.1. Corollary 5.4 reveals a fundamental tension in multi-model design: models that are expressive (high Lipschitz constant, sensitive to input variations) are precisely the models that compose poorly (contamination amplifies through the pipeline). Conversely, models that compose well (low Lipschitz constant, insensitive to input variations) may lack the expressiveness needed

for the task. The Lyapunov stability envelope (AI-3) provides the per-component Lipschitz bounds; the small gain condition translates these bounds into a system-level composition constraint. This trade-off is not an engineering limitation—it is a mathematical fact about interconnected dynamical systems.

For recursive compositions (agent loops), the small gain condition takes a particularly important form. An agent loop that invokes model M with Lipschitz constant L through a feedback function F with Lipschitz constant L_F has loop gain $L \cdot L_F$. The stability condition $L \cdot L_F < 1$ constrains the combined sensitivity of the model and its feedback mechanism. This connects directly to the γ-contractive condition in the recursive composition operator (Definition 3.8): $\gamma = L \cdot L_F$, and the contractive case requires $\gamma < 1$.

CRSA-REQ-SMALL-GAIN

At CRSA-1 conformance level C2 and above, multi-model systems **SHALL** compute the contamination gain matrix Γ and verify that $\rho(\Gamma) < 1$. The spectral radius and the gain matrix entries **SHALL** be included in the system-level attestation token. If $\rho(\Gamma) \geq 1$, the system **MUST NOT** receive a CRSA-1 composition certificate at level C2 or above. At conformance level C1, the gain matrix **SHALL** be documented but the spectral radius condition is advisory rather than mandatory.

5.4 Practical Isolation Mechanisms

The small gain condition establishes *when* a multi-model system has bounded contamination. This section identifies the practical mechanisms that *achieve* bounded contamination in production deployments.

5.4.1 Explicit Channel Isolation

Explicit channels—the intended data flows between components—are isolated through the composition algebra's guarantee-to-assumption matching. When $G_1 \Rightarrow A_2$ with low ϵ, the explicit channel is "clean": the upstream component's output meets the downstream component's expectations with high probability.

The primary mechanism for reducing explicit-channel contamination is **interface sanitization**: transforming the output of component M_i before it enters component M_j to enforce the assumption A_j. Common sanitization operations include output truncation (bounding the information content), format enforcement (structured JSON rather than free-form text, reducing steganographic bandwidth), confidence filtering (dropping low-confidence outputs), and distribution normalization (re-calibrating output distributions to match A_j's distributional assumptions).

Each sanitization operation has a quantifiable effect on the explicit contamination gain:

Mechanism	Contamination Reduction	Information Cost
Output truncation	Proportional to truncation ratio	Reduced context
JSON schema enforcement	Steganographic bandwidth $\to$ near-zero	Expressiveness constrained
Confidence filtering	Removes high-uncertainty paths	Reduced recall
Distribution normalization	$\epsilon \to 0$ by construction	Computational overhead

5.4.2 Implicit Channel Isolation

Implicit channels—the unintended infrastructure side channels—require hardware or virtualization mechanisms to isolate. The Stateful Isolation Law's compliance tiers define four levels of

infrastructure isolation:

Tier 0 (Shared Everything): No isolation. All five contamination channels are active. Suitable only for same-principal multi-model systems where cross-contamination is acceptable.

Tier 1 (Logical Isolation): Separate KV caches, separate prompt contexts, continuous batching with timing noise. Eliminates C^{kv} and C^{sp}, reduces C^{cb}. Achievable through inference engine configuration.

Tier 2 (Hardware Partition): NVIDIA MIG or equivalent GPU partitioning. Separate compute and memory partitions per principal. Eliminates C^{gpu} at the logical level but residual TLB covert channels remain (31 kbps per CCS 2023). Achievable through hardware configuration.

Tier 3 (Physical Isolation): Separate physical GPUs per principal, no shared NVLink, separate host memory. Eliminates all infrastructure channels except shared-weight contamination (if models share weights). NVBleed (cross-GPU via NVLink) is eliminated by removing the NVLink interconnect from the isolation boundary.

CRSA-REQ-ISOLATION-TIER

Multi-model systems **SHALL** declare their isolation tier for each pair of distinct-principal models. The declared tier **SHALL** be consistent with the implicit contamination bounds included in the system-level attestation. The following mapping applies:

- CRSA-1 conformance level C0 (Documented): Any isolation tier; documentation only.
- CRSA-1 conformance level C1 (Composed): Tier 1 minimum for distinct-principal pairs.
- CRSA-1 conformance level C2 (Attested): Tier 2 minimum for distinct-principal pairs.
- CRSA-1 conformance level C3 (Sovereign): Tier 3 for all distinct-principal pairs, with formally verified contamination bounds.

5.4.3 Cross-Correlation Correction for Parallel Composition

The parallel composition operator (Theorem 3.2) assumes independent component failures. When components share infrastructure, their failures may be correlated. The cross-correlation correction adjusts the composed violation probability.

Definition 5.4 (Cross-Correlation Factor)**.** *For components M_i and M_j sharing infrastructure, the **cross-correlation factor** is:*

$$\rho_{ij} = \frac{\Pr[\neg G_i \wedge \neg G_j \mid A_i \wedge A_j] - \delta_i \cdot \delta_j}{\sqrt{\delta_i(1-\delta_i) \cdot \delta_j(1-\delta_j)}}$$

where $\rho_{ij} = 0$ indicates independent failures and $\rho_{ij} = 1$ indicates perfectly correlated failures (common-mode failure).

For an ensemble with pairwise cross-correlation ρ, the corrected violation probability for majority voting with $n = 2m + 1$ components is bounded by:

$$\delta_{\mathcal{A}}^{\text{corr}} \leq \delta_{\mathcal{A}}^{\text{indep}} + \binom{n}{2} \rho \cdot \bar{\delta}^2$$

where $\delta_{\mathcal{A}}^{\text{indep}}$ is the independent-failure bound from Theorem 3.2. The correction term scales quadratically in n—large ensembles with correlated failures can be *worse* than small ensembles with independent failures.

CRSA-REQ-CORRELATION

At CRSA-1 conformance level C2 and above, parallel compositions **SHALL** include the pairwise cross-correlation factors ρ_{ij} in the system-level attestation. When $\rho_{ij} > 0.5$ for any pair, the system-level attestation **SHALL** include a warning that the ensemble's redundancy benefit is substantially degraded by correlated failures. When $\rho_{ij} > 0.9$, the components **SHALL** be treated as a single point of failure for safety budget computation.

6 Audit Log Format

Every regulatory framework that imposes obligations on AI systems requires logging. None of them specify a format. The EU AI Act Article 12 requires "automatic recording of events" sufficient for post-market monitoring. The DoD's NDAA FY2026 Section 1512 requires "model lineage documentation." Colorado SB 24-205 requires documentation of "known limitations." California SB 942 requires "provenance disclosures." OMB M-25-21 requires "data traceability." Each mandate assumes that a structured, machine-readable, tamper-evident audit log exists. No standard defines what that log looks like.

This is not a gap that the harmonized standards pipeline will close in time. CEN-CENELEC's prEN ISO/IEC 24970 (AI system life cycle logging) has not reached Enquiry stage. The earliest realistic availability for a harmonized logging standard under the EU AI Act is late 2027—eighteen months after enforcement begins. In the interim, every deployer of multi-model AI systems must invent their own logging format, producing audit trails that are incomparable across vendors, unverifiable by third parties, and inadmissible in regulatory proceedings that require standardized evidence.

CRSA-1 fills this vacuum with a pipeline-aware audit log format built on established semantic foundations. Section 6.1 defines the semantic model. Section 6.2 specifies the event schema. Section 6.3 defines the pipeline provenance DAG. Section 6.4 specifies tamper-evidence and retention requirements. Section 6.5 maps the format to specific regulatory requirements. Section 6.6 addresses interoperability with enterprise GRC platforms.

6.1 Semantic Model: PROV-DM Extended for Agentic Workflows

The CRSA-1 audit log adopts W3C PROV-DM (Provenance Data Model, 2013) as its semantic foundation. PROV-DM defines three core types—Entity, Activity, and Agent—with seven relations (wasGeneratedBy, used, wasAttributedTo, wasDerivedFrom, wasAssociatedWith, actedOnBehalfOf, wasInformedBy) that capture the complete provenance of any artifact.

PROV-DM was designed for general provenance and has been adopted across scientific computing, government data management, and supply chain traceability. Its application to AI workflows, however, requires extension. Nian and Shanahan ("PROV-AGENT," arXiv:2508.02866, 2025) provide precisely this extension, defining PROV-DM specializations for agentic AI workflows built on the Model Context Protocol:

- **ToolCallActivity:** An Activity representing a single tool invocation by an agent, capturing the tool name, input parameters, output result, and invocation timestamp.
- **MCPServerAgent:** An Agent representing an MCP server that provides tool capabilities, with attributes for server identity, capability manifest, and trust level.
- **ConversationEntity:** An Entity representing the conversation state at a point in time, linking the sequence of user messages, assistant responses, and tool results into a provenance chain.

CRSA-1 extends PROV-AGENT with composition-specific types:

- **CompositionActivity:** An Activity representing a composition operation (sequential, parallel, conditional, or recursive), capturing the operator type, the component contracts consumed, the composed contract produced, and the safety budget impact.
- **AttestationEntity:** An Entity representing a CRSA-1 attestation token, linked to the CompositionActivity that produced it and the component AttestationEntities that it aggregates.
- **PipelineAgent:** An Agent representing the composed system as a whole, with attributes for the system-level safety budget, conformance level, and composition graph topology.

The semantic model ensures that every audit log entry is interpretable without reference to CRSA-1-specific documentation: the PROV-DM semantics are self-describing, and any tool that understands PROV-DM can parse the provenance structure. The CRSA-1-specific extensions add domain semantics (composition operators, safety budgets, attestation tokens) on top of the universal provenance framework.

6.2 CRSA-1 Audit Event Schema

The audit event schema defines the structure of individual log entries. Each event corresponds to an observable occurrence in the multi-model pipeline: a model invocation, a composition operation, an attestation token generation, a safety budget threshold crossing, or an isolation bound violation.

Definition 6.1 (CRSA-1 Audit Event)**.** *A **CRSA-1 audit event** is a JSON object conforming to the following top-level structure:*

```
{
  "crsa_version": "1.0",
  "event_id": "<UUID v7>",
  "event_class": "<CRSA event class>",
  "timestamp": "<RFC 3339 with nanoseconds>",
  "pipeline_id": "<pipeline invocation UUID>",
  "component_id": "<component identifier>",
  "principal_id": "<principal identifier>",
  "provenance": {
    "prov_type": "<PROV-DM type>",
    "prov_relations": [ ... ]
  },
  "attestation_ref": "<hash of associated CRSA-1 token>",
  "safety_budget": {
    "allocated": <float>,
    "consumed": <float>,
    "remaining": <float>
  },
  "payload": { ... },
  "commitment": "<SHA-256 hash chain entry>"
}
```

CRSA-1 defines the following event classes, organized by pipeline lifecycle phase:
Initialization events:

- `PIPELINE_INIT`: Pipeline instantiation with composition graph topology, declared safety budget Δ, conformance level target, and component manifest.
- `CONTRACT_REGISTER`: Registration of a component's safety contract (A, G, δ, τ) with the pipeline's composition engine.

- `ISOLATION_DECLARE`: Declaration of isolation tier and implicit contamination bounds for each distinct-principal pair.

Inference events:

- `COMPONENT_INVOKE`: A single model invocation within the pipeline, capturing input hash, output hash, invariant measurements, contract status (assumption met / violated), and per-inference commitment.
- `COMPOSITION_APPLY`: Application of a composition operator, capturing operator type, input contracts, output contract, compatibility gap ϵ, and budget impact.
- `ROUTE_DECISION`: For conditional compositions, the routing decision with confidence score, selected component, and misrouting probability estimate.
- `LOOP_ITERATION`: For recursive compositions, the iteration counter, feedback stability assessment (assumption-preserving, contractive, or neither), and cumulative iteration budget consumption.

Safety events:

- `BUDGET_WARNING`: Cumulative safety budget consumption exceeds 80% of allocation.
- `BUDGET_EXCEEDED`: Cumulative safety budget consumption exceeds 100%. Downstream components **MUST** be notified.
- `ASSUMPTION_VIOLATED`: A component's input assumption A was not satisfied by the upstream component's output. The compatibility gap ϵ for this invocation exceeded the declared bound.
- `INVARIANT_BREACH`: A MAI-1 invariant measurement fell outside the component's contract guarantee G. The specific invariant, measured value, and threshold are recorded.
- `ISOLATION_BREACH`: Implicit contamination exceeded the declared bound for a principal pair. The channel type, measured contamination, and declared bound are recorded.
- `GAIN_UNSTABLE`: The spectral radius of the contamination gain matrix exceeds 1. The system has entered an unstable contamination regime.

Completion events:

- `PIPELINE_COMPLETE`: Normal pipeline completion with system-level attestation token, final safety budget state, and Merkle root of the pipeline DAG.
- `PIPELINE_ABORT`: Pipeline termination due to budget exhaustion, invariant breach, or isolation violation. The triggering event is cross-referenced.

CRSA-REQ-EVENT-COMPLETENESS

A CRSA-1 compliant system **SHALL** emit at minimum: one `PIPELINE_INIT` event per pipeline invocation, one `COMPONENT_INVOKE` event per model invocation, one `COMPOSITION_APPLY` event per composition operation, and one `PIPELINE_COMPLETE` or `PIPELINE_ABORT` event per pipeline invocation. Safety events **SHALL** be emitted within 100 milliseconds of the triggering condition. The event stream **SHALL** be sufficient to reconstruct the complete pipeline execution DAG from log entries alone.

6.3 Pipeline Provenance DAG

The audit events, linked by their provenance relations, form a directed acyclic graph that captures the full causal history of a pipeline invocation. This DAG is the audit artifact that regulators, auditors, and incident investigators consume.

The pipeline provenance DAG is distinct from but isomorphic to the pipeline attestation DAG (Section 4.3). The attestation DAG links cryptographic commitments; the provenance DAG links semantic events with PROV-DM relations. The two DAGs share the same topology and are cross-referenced through the `attestation_ref` field in each audit event.

The provenance DAG supports four query patterns that map to regulatory requirements:

Forward trace ("what did this input produce?"): Given an input to the pipeline, follow `wasGeneratedBy` and `wasDerivedFrom` relations forward to enumerate all outputs, intermediate artifacts, and safety events produced by that input. This supports EU AI Act Article 12's requirement for tracing the chain of events leading to a specific output.

Backward trace ("where did this output come from?"): Given a pipeline output, follow `used` and `wasDerivedFrom` relations backward to identify all inputs, model invocations, and composition operations that contributed to the output. This supports incident investigation and the DoD's traceability requirement.

Impact analysis ("what was affected by this failure?"): Given a safety event (invariant breach, assumption violation, budget exceeded), follow forward relations to identify all downstream outputs that were produced while the failure condition was active. This supports post-market monitoring and recall scoping.

Counterfactual analysis ("what would have happened without this component?"): Given a component in the pipeline, identify the composition operations that consumed its output and determine whether the pipeline's safety budget would have been satisfied without that component's contribution. This supports root-cause analysis and remediation planning.

6.4 Tamper-Evidence and Retention

The audit log's evidentiary value depends on its integrity. A log that can be silently modified after the fact is useless for regulatory compliance and forensic investigation.

CRSA-REQ-TAMPER-EVIDENCE

The CRSA-1 audit log **SHALL** provide tamper-evidence through two mechanisms:

1. **Hash chain continuity:** Each event's `commitment` field includes the hash of the previous event's commitment, forming a hash chain. Any insertion, deletion, or modification of an event breaks the chain at the point of tampering. The hash chain is per-pipeline (linked by `pipeline_id`) and cross-linked to the attestation hash chain (Section 4.2).

2. **Periodic Merkle commitment:** At configurable intervals (default: every 1000 events or every 60 seconds, whichever comes first), a Merkle root is computed over the accumulated events and published to an append-only commitment store. The Merkle root provides compact proof that a specific event existed in the log at a specific time, without revealing other events (selective disclosure).

CRSA-REQ-RETENTION

CRSA-1 compliant systems **SHALL** retain audit logs for the following minimum periods:

- **EU AI Act scope:** Minimum six months from system decommissioning, or the duration required for conformity assessment and post-market monitoring, whichever is longer (derived from Article 19 technical documentation retention requirements and Article 72 post-market monitoring obligations).
- **DoD scope:** As specified by the contracting authority, with a default of three years aligned with Federal Records Act requirements for AI decision records.
- **Financial services scope:** Seven years aligned with SEC Rule 17a-4 and Federal Reserve SR 11-7 model risk management record retention.
- **General commercial scope:** Minimum one year, or the duration required by applicable law, whichever is longer.

Retention periods apply to the complete event stream including all safety events. Selective deletion of safety events within a retained log **SHALL** be treated as a tamper event.

6.5 Regulatory Mapping

The CRSA-1 audit format maps to specific regulatory requirements as follows. This mapping establishes that a single audit log format satisfies the logging obligations across multiple jurisdictions.

Table 2: CRSA-1 Audit Format Regulatory Mapping

Regulation	Requirement	CRSA-1 Satisfaction
EU AI Act Art. 12	Automatic recording of events for post-market monitoring	`COMPONENT_INVOKE` + `COMPOSITION_APPLY` events with full provenance DAG
EU AI Act Art. 15	Appropriate accuracy, robustness, cybersecurity	Invariant measurements in component events; `INVARIANT_BREACH` events with thresholds; isolation tier declarations
EU AI Act Art. 19	Technical documentation retention	`CRSA-REQ-RETENTION`: six-month minimum from decommissioning
NDAA FY2026 §1512	Model lineage documentation; adversarial testing records	Layer 3 provenance claims in attestation tokens; `CONTRACT_REGISTER` events with tested contract parameters
NDAA FY2026 §1513	Risk-based AI cybersecurity framework	Safety budget tracking; isolation tier declarations; gain matrix stability verification

Regulation	Requirement	CRSA-1 Satisfaction
Colorado SB 24-205	Documentation of known limitations	Honest Framing fields in `PIPELINE_INIT`; "what does not compose" declarations from Section 3.5 encoded as structured metadata
California SB 942	Provenance disclosures for AI-generated content	`PIPELINE_COMPLETE` events with full backward-traceable provenance DAG
OMB M-25-21	Continuous monitoring; data traceability	Real-time event emission; hash-chain tamper evidence; forward/backward trace queries
SEC Rule 17a-4 / SR 11-7	Model risk management records; immutable audit trail	Seven-year retention; periodic Merkle commitment to append-only store; hash-chain integrity

6.6 GRC Platform Interoperability

The audit log must integrate with enterprise Governance, Risk, and Compliance (GRC) platforms—the systems that procurement officers, compliance teams, and auditors actually use. A technically perfect audit format that cannot be ingested by ServiceNow, Archer, or LogicGate is operationally useless.

CRSA-1 achieves GRC interoperability through alignment with the Open Cybersecurity Schema Framework (OCSF). OCSF is an open-source schema developed by the Linux Foundation with over 1,000 contributors from AWS, Splunk, IBM, CrowdStrike, and other major security vendors. It provides a vendor-neutral taxonomy of security events organized into categories (System Activity, Findings, Identity, Network, Application) with standardized field names, data types, and enumeration values.

OCSF v3.1.0 does not include native AI decision event classes. CRSA-1 defines an OCSF extension that maps CRSA-1 event classes to the OCSF taxonomy:

CRSA-1 Event Class	OCSF Category / Class
`PIPELINE_INIT`	Application Activity / App Lifecycle
`COMPONENT_INVOKE`	Application Activity / API Activity
`COMPOSITION_APPLY`	Application Activity / API Activity (extended)
`BUDGET_WARNING`	Findings / Security Finding (severity: Medium)
`BUDGET_EXCEEDED`	Findings / Security Finding (severity: High)
`INVARIANT_BREACH`	Findings / Compliance Finding (severity: Critical)
`ISOLATION_BREACH`	Findings / Security Finding (severity: Critical)
`GAIN_UNSTABLE`	Findings / Security Finding (severity: Critical)
`PIPELINE_COMPLETE`	Application Activity / App Lifecycle
`PIPELINE_ABORT`	Application Activity / App Lifecycle (disposition: Blocked)

The OCSF mapping ensures that CRSA-1 events can be ingested by any SIEM (Security Information and Event Management) system that supports OCSF—which includes the major enterprise platforms. Safety events (`BUDGET_EXCEEDED`, `INVARIANT_BREACH`, `ISOLATION_BREACH`,

`GAIN_UNSTABLE`) map to OCSF Findings with appropriate severity levels, enabling them to flow directly into existing incident response workflows without custom integration.

CRSA-REQ-OCSF

CRSA-1 compliant systems **SHOULD** emit audit events in both the native CRSA-1 JSON schema and the OCSF-mapped schema. At CRSA-1 conformance level C1 and above, the OCSF-mapped output **SHALL** be available as a configurable export format. The OCSF extension class definitions **SHALL** be published as an open-source OCSF profile, enabling GRC platform vendors to add native CRSA-1 support without CRSA-1-specific development.

7 Agent Interaction Protocol Constraints

The preceding sections define how safety certificates compose (Section 3), how composition is attested (Section 4), how contamination is bounded (Section 5), and how everything is logged (Section 6). This section addresses the remaining attack surface: the communication channel between agents itself.

When Agent A sends a message to Agent B, that message is simultaneously data (information for B to process) and a potential vector (instructions that could compromise B's safety posture). The fundamental vulnerability of LLM-based agents—the inability to reliably distinguish instructions from data within the same context window—means that every inter-agent message is a potential injection payload. Section 2.2 cataloged the threat vectors that exploit this channel: steganographic collusion, cascading jailbreaks, confused deputy attacks, and MCP security gaps.

CRSA-1 addresses these threats not by solving the instruction-data separation problem (which remains open) but by constraining the communication channel to reduce its capacity for carrying adversarial payloads while preserving its capacity for carrying legitimate information. The design principle is: *structure is security*. Free-form natural language between agents is an unrestricted channel; structured, schema-enforced, sanitized communication is a constrained channel whose adversarial capacity is formally bounded.

Section 7.1 specifies mandatory structured communication. Section 7.2 defines anti-steganography requirements. Section 7.3 specifies cascade-breaking checkpoints. Section 7.4 defines privilege boundary enforcement. Section 7.5 extends MCP with a governance layer.

7.1 Mandatory Structured Communication

CRSA-REQ-STRUCTURED-COMM

At CRSA-1 conformance level C1 and above, all inter-agent communication within a CRSA-1 compliant pipeline **SHALL** conform to a declared schema. Free-form natural language **MUST NOT** be the sole format of inter-agent messages. The schema **SHALL** be declared in the `PIPELINE_INIT` audit event and **SHALL** be immutable for the duration of the pipeline invocation.

The structured communication requirement is grounded in empirical evidence. The Sibylline Software analysis (February 2026) demonstrated that strict JSON output schemas with embedded fingerprint challenges eliminated prompt injection propagation entirely across four models (GPT-4o, Claude 3.5 Sonnet, Llama 3.1 70B, Mistral Large) and nine attack categories (direct injection, indirect injection, jailbreak, role-play, encoding-based, multi-turn, context overflow, delimiter confusion, and instruction hierarchy). The mechanism is straightforward: when a model's output must conform to a predefined JSON schema, the degrees of freedom available for encoding

adversarial payloads are dramatically reduced. The model cannot embed arbitrary instructions in its output because the output must parse as valid JSON matching the declared schema.

CRSA-1 does not mandate a specific schema language. JSON Schema (RFC draft, widely adopted), Protocol Buffers, CDDL (RFC 8610, used in the CBOR ecosystem), and XML Schema are all acceptable. The requirement is structural: the schema **MUST** define:

1. **Field enumeration:** Every field in the inter-agent message is explicitly declared with a name, type, and cardinality. No untyped or catch-all fields (e.g., `"metadata": any`) are permitted at conformance level C2 and above.

2. **Value constraints:** Numeric fields have declared ranges. String fields have declared maximum lengths and, where applicable, enumerated allowed values or regular expression patterns. Unconstrained string fields **SHALL** have a maximum length of 4096 characters at conformance level C2 and above.

3. **No embedded instructions:** The schema **MUST NOT** include fields whose semantic purpose is to carry instructions for the receiving agent (e.g., fields named "system_prompt", "instructions", "override", or functionally equivalent). The receiving agent's behavior is determined by its own configuration and safety contract, not by fields in the incoming message.

4. **Provenance fields:** Every inter-agent message **SHALL** include a `source_component_id`, `attestation_ref` (hash of the sending component's most recent CRSA-1 attestation token), and `sequence_number` (monotonically increasing within the pipeline invocation).

Remark 7.1. The "no embedded instructions" requirement is the most operationally consequential constraint in CRSA-1. Many existing agentic frameworks—including LangChain, CrewAI, and AutoGPT—pass natural language instructions between agents as the primary coordination mechanism: the planner sends the executor a natural language description of the task, the executor sends the critic a natural language request for evaluation, and so on. Under CRSA-1, these natural language instructions must be replaced with structured task descriptors that the receiving agent interprets according to its own configuration. This is a significant architectural change. It is also the single most effective mitigation against cascading jailbreaks and confused deputy attacks, because it eliminates the channel through which adversarial instructions propagate.

7.2 Anti-Steganography Requirements

Structured communication constrains the *syntactic* channel between agents. Steganographic collusion exploits the *semantic* channel: subtle choices in word selection, numeric precision, ordering of list elements, or optional field inclusion that are individually innocuous but collectively encode hidden information. Even within a strict JSON schema, an agent with access to free-text string fields or numeric fields with high precision can encode bits through choices that are invisible to schema validation.

CRSA-1 addresses steganography through three layers of defense: bandwidth reduction, output sanitization, and activation monitoring.

7.2.1 Bandwidth Reduction

The primary defense against steganography is reducing the number of bits available for covert encoding. Each constraint on the communication schema reduces the covert channel bandwidth:

Constraint	Bandwidth Reduction	Mechanism
Enumerated string values (vs. free text)	Near-total	$\log_2(\|E\|)$ bits per field vs. ∼8 bits per character
Numeric precision limits (e.g., 2 decimal places)	Substantial	Eliminates low-order bit encoding
Fixed field ordering (vs. arbitrary)	Moderate	Eliminates $\log_2(n!)$ bits of permutation encoding
Maximum string length (e.g., 256 chars)	Proportional	Caps total available encoding space
Boolean fields (vs. numeric)	Near-total	1 bit per field, semantically constrained

CRSA-REQ-BANDWIDTH

At CRSA-1 conformance level C2 and above, the declared inter-agent communication schema **SHALL** limit the covert channel bandwidth to at most 32 bits per message. This is computed as the sum of:

- $\log_2(|E_f|)$ bits for each enumerated field f with $|E_f|$ allowed values.
- $\log_2(\text{range}_f/\text{precision}_f)$ bits for each numeric field f with declared range and precision.
- $L_f \cdot \log_2(|\Sigma|)$ bits for each free-text string field f with maximum length L_f over alphabet Σ.
- $\log_2(n!)$ bits for field ordering freedom over n unordered fields.

When the computed bandwidth exceeds 32 bits, the schema **MUST** be tightened by reducing string field lengths, enumerating values, or fixing field ordering. The 32-bit threshold corresponds to approximately 4×10^9 distinguishable messages—sufficient for legitimate task coordination but insufficient for transmitting meaningful covert payloads such as exfiltrated data, model weights, or multi-step adversarial programs.

7.2.2 Output Sanitization

Even within bandwidth-limited schemas, subtle encoding is possible through choices that schema validation cannot detect: the specific synonym chosen from allowed values, the specific numeric value within a constrained range, or timing variations in response delivery. Output sanitization addresses these residual channels.

CRSA-REQ-SANITIZE

At CRSA-1 conformance level C2 and above, an inter-agent message **SHALL** be processed by a **sanitization function** σ before delivery to the receiving component. The sanitization function **SHALL** be:

1. **Deterministic:** σ produces the same output for the same input, eliminating timing-based covert channels in the sanitization step itself.
2. **Schema-preserving:** $\sigma(m)$ conforms to the same schema as m.
3. **Semantics-preserving:** $\sigma(m)$ conveys the same task-relevant information as m, as determined by a semantic equivalence relation defined in the pipeline configuration.
4. **Entropy-reducing:** $H(\sigma(m)) \leq H(m)$ where H denotes the Shannon entropy over the message space. Concretely, the sanitization function canonicalizes representations: normalizing numeric precision, sorting list elements into a canonical order, mapping synonyms to canonical forms, and stripping optional fields that are not required for the downstream component's contract assumptions.

The sanitization function operates at the composition boundary—between the sending component's output and the receiving component's input. It is logically part of the CRSA-1 composition layer, not part of either component. This separation ensures that neither the sender nor the receiver can influence the sanitization process.

7.2.3 Activation Monitoring

Bandwidth reduction and output sanitization address the *communication channel* between agents. Activation monitoring addresses the *encoding mechanism* within agents: detecting when a model is attempting to encode covert information in its outputs, even if the encoded information is not successfully decoded by the receiver.

Li et al. ("Hide and Seek in Embedding Space," arXiv:2601.22818, January 2026) found that while output-level statistical tests (KL divergence, perplexity) are unreliable for steganography detection—covert channels can operate within typical fine-tuning ranges— linear probes on late-layer activations provide partial detection cues. Specifically, classifiers trained on late-layer representations can distinguish steganographic outputs from normal outputs with accuracy significantly above chance, even when output-level statistics are indistinguishable.

CRSA-REQ-ACTIVATION-MONITOR

At CRSA-1 conformance level C3 (Sovereign), components **SHALL** expose late-layer activation statistics (mean, variance, and top-k principal component projections of the final hidden state) as part of their MAI-1 attestation. The CRSA-1 composition layer **SHOULD** maintain a baseline distribution of these statistics computed during contract calibration and **SHOULD** emit an `ANOMALY_DETECTED` audit event when the activation statistics deviate from baseline by more than a configurable threshold (default: 3 standard deviations on any principal component). This is an advisory detection mechanism, not a blocking mechanism: the composition layer **MUST NOT** block inference based on activation anomalies alone, because false positive rates for steganography detection remain too high for enforcement use.

Honest Framing

Anti-steganography is an arms race. CRSA-1's three-layer defense (bandwidth reduction, output sanitization, activation monitoring) raises the cost and reduces the bandwidth of covert channels between agents. It does not eliminate them. A sufficiently capable agent can encode information in the *semantic content* of its outputs—the specific facts it chooses to include, the reasoning path it follows, the emphasis it places on different aspects of a response—in ways that no schema validation, sanitization function, or activation probe can detect without understanding the full semantic context. CRSA-1 reduces the covert channel to a trickle; it does not close the channel entirely. This is consistent with the broader security principle that defense-in-depth reduces risk without eliminating it.

7.3 Cascade-Breaking Checkpoints

Cascading jailbreaks propagate because a compromised agent's output is consumed as trusted input by downstream agents. The propagation mechanism is: adversarial payload in Agent A's output → Agent B's context window → Agent B follows the payload as instructions → Agent B's output contains the propagated payload → Agent C's context window, and so on. The infection is self-replicating because each compromised agent reproduces the payload in its own output.

CRSA-1 breaks this propagation chain by requiring safety certificate re-verification at every agent boundary.

CRSA-REQ-CASCADE-BREAK

At CRSA-1 conformance level C1 and above, every composition boundary (every edge in the pipeline attestation DAG) **SHALL** include a **cascade-breaking checkpoint** that performs the following verification before the downstream component processes the upstream component's output:

1. **Certificate freshness:** The upstream component's attestation token **MUST** be within its validity horizon τ. Stale tokens are rejected.
2. **Contract satisfaction:** The upstream component's guarantee G **MUST** have been satisfied for the current invocation, as attested in the per-inference commitment. If the upstream component's most recent invariant measurement violated G, the downstream component **MUST NOT** accept its output.
3. **Budget check:** The cumulative safety budget consumed by the pipeline up to this boundary **MUST NOT** exceed the declared budget Δ. If the budget is exhausted, the pipeline **MUST** abort (emitting a `BUDGET_EXCEEDED` event).
4. **Schema compliance:** The upstream component's output **MUST** conform to the declared inter-agent communication schema. Messages that fail schema validation are rejected without further processing.
5. **Sanitization:** The upstream component's output **MUST** pass through the sanitization function σ before delivery to the downstream component (Section 7.2.2).

The cascade-breaking checkpoint is the composition-layer analog of a network firewall: it inspects traffic crossing a trust boundary and blocks traffic that does not meet the declared policy. The critical difference from network firewalls is that the "traffic" is semantically rich (natural language or structured data with complex semantics) and the "policy" is a safety contract (formal assume-guarantee conditions), making the inspection more nuanced than packet filtering but also more expensive.

CRSA-REQ-CHECKPOINT-LATENCY

The cascade-breaking checkpoint **SHALL** add no more than 50 microseconds of latency per composition boundary. This budget covers: token verification (≤ 15 μs for signature check), contract status lookup (≤ 5 μs from in-memory attestation cache), budget computation (≤ 5 μs for running sum), schema validation (≤ 10 μs for pre-compiled schema validator), and sanitization (≤ 15 μs for canonicalization operations). For a 5-stage pipeline, the total checkpoint overhead is 250 μs—less than 1% of a single LLM inference step.

The checkpoint's cascade-breaking property is straightforward to verify: if a compromised Agent A produces output that violates its safety contract G_A (because following the adversarial payload caused it to deviate from its certified behavior), the checkpoint at the A→B boundary detects the contract violation (step 2) and rejects the output. The infection cannot propagate to Agent B because B never receives the compromised output. If the compromised agent manages to produce output that satisfies its safety contract while still carrying an adversarial payload (a more sophisticated attack), the structured communication requirement (Section 7.1) and sanitization (Section 7.2.2) constrain the payload's effectiveness, and the bandwidth limitation (Section 7.2.1) bounds the information content available for encoding the payload.

7.4 Privilege Boundary Enforcement

CRSA-REQ-PRIVILEGE

At CRSA-1 conformance level C1 and above, every component in a CRSA-1 compliant pipeline **SHALL** operate under a declared **capability set** that enumerates the actions the component is authorized to perform. The capability set **SHALL** be:

1. **Declared at initialization:** The capability set is specified in the `CONTRACT_REGISTER` audit event and bound to the component's safety contract.
2. **Non-delegable by default:** A component **MUST NOT** delegate capabilities to downstream components unless the delegation is explicitly authorized in the pipeline configuration. The default policy is: each component's capability set is the intersection of its declared capabilities and the capabilities of the pipeline as a whole.
3. **Non-escalating under composition:** The composed system's effective capability set **MUST NOT** exceed the union of its components' declared capability sets. A pipeline that composes a database-reader and an email-sender **MUST NOT** acquire the composite capability "read database and email contents" unless this composite capability is explicitly declared in the pipeline configuration.
4. **Attested:** The component's capability set **SHALL** be included in its CRSA-1 attestation token. A relying party can verify that the component operated within its declared capabilities by checking the attestation.

The non-escalation property is the key defense against confused deputy and semantic privilege escalation attacks. It is enforced through a **capability algebra** that mirrors the safety certificate algebra:

Definition 7.1 (Capability Composition)**.** *For components with capability sets* Cap_1 *and* Cap_2*,*

the composed capability set under each composition operator is:

$$\begin{aligned}
\text{Cap}_1 \otimes_{\text{seq}} \text{Cap}_2 &= \text{Cap}_1 \cup \text{Cap}_2 && \textit{(sequential: both capabilities are used)} \\
\text{Cap}_1 \otimes_{\text{par}} \text{Cap}_2 &= \text{Cap}_1 \cup \text{Cap}_2 && \textit{(parallel: both capabilities are used)} \\
\text{Cap}_1 \otimes_{\text{cond}} \text{Cap}_2 &= \text{Cap}_1 \cup \text{Cap}_2 && \textit{(conditional: either capability may be used)} \\
\text{Cap} \otimes^{K}_{\text{rec}} \text{Cap} &= \text{Cap} && \textit{(recursive: same capabilities each iteration)}
\end{aligned}$$

The composed capability set is always the union (or identical set for recursion), never a superset containing capabilities that no individual component declared.

The non-escalation invariant is: $\text{Cap}_{\text{composed}} \subseteq \text{Cap}_{\text{declared}}$, where $\text{Cap}_{\text{declared}}$ is the pipeline-level capability set declared in `PIPELINE_INIT`. If a composition produces a capability set that exceeds the declared pipeline capabilities, the `PIPELINE_INIT` declaration was incomplete and the pipeline **MUST** be reconfigured before attestation.

Remark 7.2. The capability algebra is deliberately coarse: it tracks the *set* of authorized actions, not the *conditions* under which those actions are authorized. A more fine-grained algebra could track conditional capabilities ("Agent B may send email *only if* Agent A's output was classified as non-sensitive"), but this requires a policy language that is beyond CRSA-1's scope. The coarse algebra provides a sound but conservative bound: it may flag as "escalation" some compositions that are in fact safe under conditional analysis. This conservatism is acceptable at the protocol specification level; implementers may refine the capability model within the bounds CRSA-1 establishes.

7.5 MCP Governance Extension

The Model Context Protocol provides the connectivity layer for multi-agent systems: tool discovery, invocation, and data exchange. CRSA-1 provides the governance layer: safety attestation, composition verification, and enforcement. This section specifies how CRSA-1 extends MCP with governance capabilities, positioned as a safety layer on top of MCP's connectivity layer.

The extension does not modify the MCP specification. It defines additional message types and metadata fields that CRSA-1 compliant MCP implementations **SHALL** support alongside the standard MCP protocol.

CRSA-REQ-MCP-TRUST

CRSA-1 compliant MCP servers and clients **SHALL** support the following governance extensions:

1. **Attestation exchange:** When establishing an MCP session, the server and client **SHALL** exchange CRSA-1 attestation tokens. The client **SHALL** verify the server's attestation before invoking any tools; the server **SHALL** verify the client's attestation before accepting any invocations. Verification includes: token signature validation, certificate freshness check, conformance level compatibility (a C2 client **MUST NOT** invoke tools on a C0 server without explicit policy override), and capability set intersection (the server's tool capabilities must be within the client's declared capability set for the pipeline).

2. **Per-invocation attestation:** Each tool invocation **SHALL** include the client's current per-inference commitment (Section 4.2), enabling the server to verify that the client's safety contract was satisfied at the time of invocation. Each tool response **SHALL** include the server's per-invocation commitment, enabling the client to include the tool's attestation in the pipeline DAG.

3. **Capability attestation:** The MCP server's capability manifest (the list of tools it provides) **SHALL** be signed and included in the server's attestation token. Changes to the capability manifest—adding, removing, or modifying tools—**SHALL** trigger a new attestation token and a `CAPABILITY_CHANGE` audit event. This addresses the "rug-pull" attack vector where trusted tools silently push updates with harmful functionality.

4. **Cascading authorization:** When an MCP tool invocation triggers a downstream agent invocation (agent-to-agent delegation via MCP), the delegation **SHALL** include the originating pipeline's capability set and safety budget state. The downstream agent **MUST NOT** accept the delegation if the originating pipeline's budget is exhausted or if the delegation would require capabilities exceeding the originating pipeline's declared set.

CRSA-REQ-MCP-VERSION

CRSA-1 compliant MCP implementations **SHALL** support **version pinning**: the client declares the expected server capability manifest hash at session initialization, and the server rejects the session if its current manifest does not match. This prevents mid-session tool modifications and provides the "change alerts" that the base MCP specification lacks. The expected manifest hash **SHALL** be included in the `PIPELINE_INIT` audit event for each MCP server dependency.

The MCP governance extension positions CRSA-1 as the natural governance companion to MCP's connectivity specification. MCP solved the $N \times M$ integration problem for AI tool use. CRSA-1 solves the composition safety problem that MCP's success has created: 97 million SDK downloads represent 97 million potential vectors for unattested, ungoverned multi-agent interaction. The governance extension ensures that MCP's connectivity is matched by accountability.

8 Conformance Levels

CRSA-1 defines four conformance levels that provide a graduated adoption path from documentation-only compliance to formally verified compositional safety. The levels are cumulative: each level includes all requirements of the levels below it. The levels are binary: a system either passes or fails at each level, with no partial compliance and no interpretive discretion.

The conformance levels are designed to serve three functions simultaneously. First, they provide a procurement language: an RFP can specify "CRSA-C2 required" and both parties understand exactly what is included and excluded. Second, they provide a regulatory mapping: different regulatory frameworks require different levels of evidence, and the conformance levels translate regulatory intent into technical requirements. Third, they provide an adoption ladder: organizations can begin at C0 with minimal investment, demonstrate value, and ascend to higher levels as infrastructure and expertise mature.

8.1 Level Definitions

8.1.1 CRSA-C0: Documented

CRSA-C0 establishes baseline visibility into the multi-model system's composition structure without requiring runtime enforcement.

CRSA-C0 Requirements

A system conforms to CRSA-C0 if and only if:

1. The pipeline composition graph is documented, identifying all component models, their composition topology (sequential, parallel, conditional, recursive), and the data flows between them.
2. Each component model's safety contract (A, G, δ, τ) is identified, even if informally. At C0, contracts may be stated in natural language rather than formal predicates.
3. The system-level safety budget Δ is declared, even if computed informally (e.g., "the sum of component violation probabilities").
4. The isolation tier for each distinct-principal pair is documented.
5. The inter-agent communication format is documented (schema not required to be enforced at runtime).
6. Audit events conforming to the CRSA-1 schema are emitted for `PIPELINE_INIT` and `PIPELINE_COMPLETE` events at minimum.

What C0 provides: Visibility. An auditor can understand the system's composition structure, identify the component models, and trace the data flow. This is the minimum information required for any meaningful governance assessment.

What C0 does not provide: Runtime verification, cryptographic attestation, composition algebra enforcement, or tamper-evident logging. C0 is a documentation standard, not an assurance standard. It represents the floor below which multi-model governance is effectively absent.

Regulatory alignment: C0 satisfies the documentation requirements of Colorado SB 24-205 (known limitations disclosure) and the basic documentation tier of the EU AI Act Article 11 (technical documentation). It does not satisfy Article 12 (automatic event logging), Article 15 (accuracy and robustness), or any DoD framework requirement.

8.1.2 CRSA-C1: Composed

CRSA-C1 adds formal composition algebra enforcement and operational audit logging, establishing that the system's safety properties have been formally derived from component properties.

CRSA-C1 Requirements (cumulative with C0)

A system conforms to CRSA-C1 if and only if it conforms to CRSA-C0 and additionally:

1. Each component model's safety contract is formally specified: A and G are machine-readable predicates, δ is a numerically bounded probability, and τ is a specified duration.
2. The composition algebra (Section 3) has been applied to derive the system-level safety certificate from component certificates. The derivation is documented and reproducible: a third party with access to the component contracts and composition graph can independently verify the system-level certificate.
3. Guarantee-to-assumption compatibility is verified at every composition boundary. Compatibility gaps ϵ_i are computed and included in the safety budget.
4. The safety budget Δ is formally computed and tracked. Budget consumption is logged in `COMPOSITION_APPLY` audit events.
5. The contamination gain matrix Γ is computed and documented. The spectral radius $\rho(\Gamma)$ is reported (advisory at C1; mandatory stability condition at C2).
6. All inter-agent communication conforms to a declared schema (CRSA-REQ-STRUCTURED-COMM).
7. Cascade-breaking checkpoints are operational at every composition boundary (CRSA-REQ-CASCADE-BREAK), performing at minimum certificate freshness and schema compliance checks.
8. The full CRSA-1 audit event stream is operational: all event classes defined in Section 6.2 are emitted with hash-chain tamper evidence.
9. Isolation tier is Tier 1 minimum for distinct-principal pairs.

What C1 provides: Formal composition assurance. The system's safety properties are mathematically derived from component properties using the CRSA-1 algebra, and the derivation is independently verifiable. The audit log captures the full pipeline execution with tamper evidence.

What C1 does not provide: Real-time cryptographic attestation (tokens are not generated per-inference), TEE anchoring (attestation keys are not hardware-bound), or mandatory small-gain stability (spectral radius is advisory). C1 is a formal composition standard with operational logging, not a cryptographic assurance standard.

Regulatory alignment: C1 satisfies EU AI Act Articles 11 and 12 (technical documentation and automatic event logging), Article 9 (risk management system with formal composition analysis), OMB M-25-21 (continuous monitoring and data traceability), and California SB 942 (provenance disclosures through the audit DAG). C1 partially satisfies Article 15 (accuracy and robustness through formal composition) but does not provide the real-time evidence that Article 15's "appropriate levels" language increasingly implies for high-risk systems.

8.1.3 CRSA-C2: Attested

CRSA-C2 adds cryptographic runtime attestation, hardware anchoring, and mandatory stability enforcement, establishing that the system's compositional safety is continuously verified at inference time.

CRSA-C2 Requirements (cumulative with C1)

A system conforms to CRSA-C2 if and only if it conforms to CRSA-C1 and additionally:

1. Per-inference commitments (Section 4.2) are generated for every model invocation within the pipeline, using plugin-based or wrapper-based integration (not sidecar).
2. CRSA-1 attestation tokens (Section 4.1) are generated for every component invocation, signed with keys bound to TEE attestation reports (CRSA-REQ-TEE-ANCHOR, binding method 1 or 2).
3. The pipeline attestation DAG (Section 4.3) is constructed with Merkle commitment for every pipeline invocation. The Merkle root is included in the system-level attestation token.
4. The small gain condition $\rho(\Gamma) < 1$ is mandatory and enforced: systems with $\rho(\Gamma) \geq 1$ **MUST NOT** receive C2 attestation.
5. Implicit contamination bounds are declared and enforced for every distinct-principal pair (CRSA-REQ-IMPLICIT-BOUND). Infrastructure sharing topology is attested.
6. Isolation tier is Tier 2 minimum for distinct-principal pairs.
7. Anti-steganography bandwidth limitation is enforced: inter-agent communication schemas limit covert channel bandwidth to at most 32 bits per message (CRSA-REQ-BANDWIDTH).
8. Output sanitization is operational at every composition boundary (CRSA-REQ-SANITIZE).
9. Cascade-breaking checkpoints perform full verification: certificate freshness, contract satisfaction, budget check, schema compliance, and sanitization (all five steps of CRSA-REQ-CASCADE-BREAK).
10. Capability sets are declared, attested, and the non-escalation invariant is enforced (CRSA-REQ-PRIVILEGE).
11. MCP governance extensions are operational for all MCP-mediated interactions (CRSA-REQ-MCP-TRUST, CRSA-REQ-MCP-VERSION).
12. Cross-correlation factors ρ_{ij} are computed and included in attestation for all parallel compositions (CRSA-REQ-CORRELATION).
13. Attestation latency overhead is within budget: ≤ 100 μs per inference step (CRSA-REQ-LATENCY), ≤ 50 μs per checkpoint (CRSA-REQ-CHECKPOINT-LATENCY).
14. Periodic Merkle commitments to an append-only store are operational for tamper evidence (CRSA-REQ-TAMPER-EVIDENCE).

What C2 provides: Cryptographic compositional assurance. Every inference step in every component is attested with hardware-anchored cryptographic commitments. The composition is

provably stable (small gain condition), the communication channels are provably constrained (bandwidth limitation), and the full pipeline execution is cryptographically committed in a tamper-evident Merkle structure. A relying party can verify the system's compositional safety state without trusting the system operator.

What C2 does not provide: Formal verification of component contracts (contracts are empirically calibrated, not theorem-proved), noninterference isolation (Tier 2 has residual TLB covert channels), or completeness guarantees (emergent properties remain outside scope). C2 is the level at which "trust but verify" becomes "verify then trust"—but the verification is of configuration and process compliance, not of behavioral safety.

Regulatory alignment: C2 satisfies all EU AI Act requirements for high-risk systems including Article 15 with continuous real-time evidence. C2 satisfies NDAA FY2026 Sections 1512 and 1513 requirements for AI cybersecurity and risk-based framework compliance. C2 satisfies SEC Rule 17a-4 immutable audit trail requirements through the Merkle commitment store. C2 is the target level for enterprise deployment in regulated industries and government procurement.

8.1.4 CRSA-C3: Sovereign

CRSA-C3 adds formal verification, physical isolation, and complete provenance, establishing the highest level of compositional assurance achievable within the theoretical limits identified in Section 3.5.

CRSA-C3 Requirements (cumulative with C2)

A system conforms to CRSA-C3 if and only if it conforms to CRSA-C2 and additionally:

1. Component safety contracts are **formally verified**: the assumption and guarantee predicates are expressed in a formal logic, and the contract satisfaction $M \models \mathcal{C}$ is established through machine-checked proof or exhaustive testing over the input domain (where domain size permits).

2. The composition algebra derivation is **machine-checked**: the system-level certificate is derived through a verified composition engine that implements the operators from Section 3 with formally verified soundness proofs.

3. Isolation tier is Tier 3 (physical isolation) for all distinct-principal pairs. All implicit contamination channels are formally verified as bounded below the declared thresholds against the Stateful Isolation Law's complete channel catalog.

4. The complete provenance DAG—from hardware platform attestation through training provenance, model lineage, inference-time attestation, and composition certification—is available as a single, cryptographically linked structure. Every link in the chain is independently verifiable.

5. Activation monitoring for steganography detection is operational (CRSA-REQ-ACTIVATION-MONITOR).

6. TEE key binding uses method 1 (TEE-resident key generation) exclusively.

7. All audit log retention requirements are satisfied for the most restrictive applicable jurisdiction (seven years for financial services scope).

What C3 provides: Maximum achievable assurance within the theoretical limits of compositional verification. Formal verification of contracts, machine-checked composition, physical isolation, and complete provenance represent the strongest evidence currently producible for multi-model AI safety.

What C3 does not provide: Behavioral safety guarantees. As established in Section 3.5, compositional verification is sound but incomplete. C3 systems can exhibit emergent behaviors that no compositional analysis predicted. C3 provides the maximum *evidence* that the system was correctly configured, correctly composed, and operating within certified parameters. It does not guarantee that correctly configured, correctly composed systems produce safe outputs in all circumstances.

Regulatory alignment: C3 exceeds all current regulatory requirements. It is positioned for future regulatory frameworks that may require formal verification of AI safety properties—a trajectory visible in the EU AI Act's emphasis on "state of the art" and the DoD's increasing interest in formally verified software for safety-critical systems. C3 is the target level for defense, critical infrastructure, and applications where the consequence of compositional failure is catastrophic.

8.2 Conformance Summary

Table 3: CRSA-1 Conformance Level Summary

Requirement Domain	**C0: Documented**	**C1: Composed**	**C2: Attested**	**C3: Sovereign**
Safety contracts	Informal / NL	Formal predicates	Formal + attested	Formally verified
Composition algebra	Not applied	Applied + documented	Applied + attested	Machine-checked
Safety budget	Declared informally	Computed + tracked	Computed + enforced + attested	Formally verified
Runtime attestation	None	None (logging only)	Per-inference CWT tokens	Per-inference + TEE-resident keys
TEE anchoring	None	None	Method 1, 2, or 3	Method 1 only
Pipeline DAG	Documented	Logged with hash chain	Merkle-committed	Merkle + full provenance
Small gain condition	Not assessed	Advisory	Mandatory ($\rho < 1$)	Formally verified
Isolation tier (distinct principals)	Any (documented)	Tier 1 minimum	Tier 2 minimum	Tier 3 (physical)
Inter-agent communication	Documented	Schema-enforced	Schema + bandwidth limit	Schema + bandwidth + activation
Sanitization	None	None	Operational	Operational
Cascade-breaking checkpoints	None	Freshness + schema	Full 5-step verification	Full + formal contract verification
Capability enforcement	None	Documented	Attested + non-escalation	Attested + formally verified
MCP governance	None	None	Full extension operational	Full + version-pinned
Audit log	Init + Complete only	Full event stream + hash chain	Full + Merkle commitment	Full + 7-year retention
Cross-correlation	Not assessed	Not assessed	Computed + attested	Formally verified

Requirement Domain	C0: Documented	C1: Composed	C2: Attested	C3: Sovereign
Target deployment	Internal / prototype	Production (non-regulated)	Regulated industries	Defense / critical infrastructure

CRSA-REQ-CONFORMANCE-BINARY

Conformance at each level is binary. A system either satisfies all requirements at a given level or it does not conform at that level. There is no "partial C2" or "C1.5." A system that satisfies all C1 requirements and 13 of 14 C2 requirements conforms at C1, not C2. This binary property is the design decision that creates enforcement pressure: if compliance can be argued, it cannot be enforced. Conformance assessment procedures are defined in CTS-1 (Document 26).

9 Honest Framing: What CRSA-1 Cannot Guarantee

Every document in the Auburn Governance Stack maintains the honest framing: explicit acknowledgment of what the specification provides, what it does not provide, and why the gap exists. This section collects and elaborates the limitations that have been noted throughout the preceding sections into a single, authoritative statement of CRSA-1's theoretical ceilings and practical boundaries.

The honest framing is not a disclaimer. It is a design feature. A specification that overstates its guarantees invites misplaced trust, which is more dangerous than no specification at all. CRSA-1 provides compositional risk reduction and accountability infrastructure. It does not provide behavioral safety guarantees. The distinction is fundamental, and this section ensures that no reader—regulator, engineer, procurement officer, or researcher—mistakes one for the other.

9.1 Compositional Verification Is Sound but Incomplete

The safety certificate algebra (Section 3) is **sound**: if the composition operators derive a system-level certificate with violation probability δ_{composed}, then the composed system's actual violation probability does not exceed δ_{composed}, provided the component contracts are correctly calibrated. Soundness means that CRSA-1 never overstates the system's safety—it may understate it (declare a system less safe than it is) but will not overstate it (declare a system safer than it is).

The algebra is **incomplete**: there exist composed systems that are safe but cannot be verified as safe through any compositional method. This is not a limitation of the CRSA-1 algebra specifically; it is a property of compositional verification generally. The incompleteness arises from two sources:

Emergent safety. Some systems are safe because of the specific interaction pattern between components, not because of any property of the individual components. A debate system where two adversarial models check each other's outputs may be safer than either model alone—but this safety emerges from the adversarial structure, not from any component-level guarantee that composes through the algebra. CRSA-1 can attest that each component operated within its contract; it cannot attest that the adversarial structure produced the safety benefit that the system designer intended.

Semantic composition. The algebra composes *certificates* (formal statements about measurable properties), not *meaning* (the semantic content of model outputs). Two components may each satisfy their safety contracts while producing a composed output whose meaning violates safety norms that were never captured in the contracts. The contracts are necessarily an abstraction of the full behavioral space; the gap between the abstraction and the behavior is where incompleteness lives.

Incompleteness is a mathematical fact, not an engineering gap. No amount of refinement to the algebra will achieve completeness for composed stochastic systems. The appropriate response to incompleteness is not to abandon compositional verification but to combine it with complementary approaches—system-level testing, red-teaming, monitoring, and human

oversight—that address the cases compositional methods cannot reach. CRSA-1 provides the compositional layer; it does not replace the other layers.

9.2 Attestation Proves Configuration, Not Behavior

CRSA-1 runtime attestation (Section 4) produces cryptographic evidence that:

- The hardware platform was the claimed platform (TEE anchoring).
- The model was the claimed model (provenance binding).
- The model's health invariants were within certified bounds at the time of inference (Layer 2 invariant attestation).
- The composition algebra was correctly applied (composition claims).
- The safety budget was not exceeded (budget tracking).
- The communication channels were constrained (schema enforcement, sanitization).

This is **configuration compliance evidence**. It proves that the system was in its certified configuration and operating within its certified parameters when it produced a specific output. It does not prove that the output was safe, correct, helpful, harmless, or honest.

The analogy to financial auditing is precise: an audit certifies that the accounting processes were followed correctly, the internal controls were operational, and the financial statements were prepared in accordance with applicable standards. It does not certify that the company will remain solvent, that its business strategy is sound, or that its management will act ethically. The audit provides *process assurance*, not *outcome assurance*. CRSA-1 attestation is process assurance for multi-model AI systems.

This limitation is inherent in the attestation approach, not specific to CRSA-1. Any attestation-based governance framework—including all IETF RATS work, all TEE-based integrity verification, and all cryptographic audit mechanisms—shares this limitation. The alternative to attestation-based governance is behavioral verification: proving that the system's outputs satisfy safety properties for all possible inputs. For LLM-based systems, behavioral verification is intractable: the input space is effectively infinite, the output distribution is stochastic, and the relationship between input and output is not amenable to formal verification with current methods. Attestation is the tractable alternative that provides meaningful (if bounded) assurance.

9.3 Isolation Bounds Are Conditional

The multi-principal isolation bounds (Section 5) provide formally derivable upper bounds on cross-principal contamination. These bounds are **conditional** on the isolation mechanisms being correctly deployed and the channel capacity measurements being accurate.

Three failure modes can invalidate the isolation bounds:

Undiscovered channels. The Stateful Isolation Law catalogs 25+ demonstrated side channels across the GPU inference stack. The catalog is comprehensive as of its publication date, but new channels are discovered regularly. GPU architectures evolve, firmware updates change isolation properties, and novel attack techniques emerge. The contamination functional bounds what is measured; it cannot bound what is not yet known. The gain matrix Γ includes only channels that have been identified and characterized.

Mechanism failure. Isolation mechanisms can fail silently. A MIG partition that is misconfigured, a KV cache eviction policy that leaks across tenant boundaries under memory pressure, or a continuous batching scheduler that exposes timing information despite noise injection—all of these produce contamination that exceeds the declared bounds without triggering

any observable error. CRSA-1's thermal integrity binding (through AI-4) provides partial detection of mechanism failure at the hardware level, but software-layer isolation failures may be undetectable from outside the isolation boundary.

Correlated failures. Section 5.4.3 introduced the cross-correlation correction for parallel composition. The correction requires accurate estimation of the correlation factor ρ_{ij}, which in practice is difficult to measure precisely. Overestimating independence (underestimating ρ_{ij}) produces optimistic safety bounds. In the worst case—common-mode failure where all components fail simultaneously due to a shared root cause (e.g., a firmware bug affecting all GPUs of a given generation)—the ensemble provides no redundancy benefit regardless of the number of components.

9.4 Agent Interaction Constraints Reduce but Do Not Eliminate Risk

The agent interaction protocol constraints (Section 7) address the communication channel between agents through structured communication, anti-steganography measures, cascade-breaking checkpoints, and privilege enforcement. Each mechanism reduces a specific attack surface:

Mechanism	What It Reduces	What Remains
Structured communication	Syntactic injection propagation	Semantic injection via legitimate fields
Anti-steganography (32-bit limit)	Covert channel bandwidth	Semantic steganography in content choices
Cascade-breaking checkpoints	Propagation of detected compromises	Propagation of undetected compromises
Privilege enforcement	Unauthorized capability acquisition	Authorized capabilities used for unintended purposes
MCP governance extension	Rug-pull attacks and session hijacking	Attacks through legitimately authorized tool behavior

In every case, the residual risk involves an adversary operating within the constraints of the legitimate channel—using authorized capabilities, conforming to declared schemas, satisfying safety contracts, yet achieving adversarial outcomes through the semantic content of communications that are syntactically and formally compliant. This is the fundamental limitation of protocol-level security applied to semantically rich systems: the protocol constrains the syntax; the adversary operates in the semantics.

CRSA-1 raises the cost, reduces the bandwidth, and increases the detectability of inter-agent attacks. It does not eliminate them. An organization deploying a CRSA-C2 compliant multi-agent system has substantially better security posture than one deploying an unattested system. It does not have perfect security. The gap between "substantially better" and "perfect" is where ongoing research, monitoring, and human oversight remain essential.

9.5 Safety Budgets Degrade Under Composition

The safety budget (Section 3.3) makes explicit a mathematical fact that is easy to overlook: every composition operation costs safety margin. There is no free composition.

For a sequential pipeline of k components each with violation probability δ, the system-level violation probability is approximately $k\delta$. A 10-stage pipeline with per-component $\delta = 10^{-3}$ has system-level $\delta \approx 10^{-2}$. A 100-stage agentic workflow with the same per-component reliability

has system-level $\delta \approx 10^{-1}$—a 10% chance of at least one component violating its safety contract during the workflow.

For recursive compositions (agent loops), the degradation is per iteration: a loop with $K = 50$ iterations and per-iteration $\delta = 10^{-3}$ has accumulated violation probability $\delta_{\text{rec}} \approx 5 \times 10^{-2}$. Longer loops, deeper pipelines, and wider ensembles all spend budget faster.

This degradation is not a flaw in CRSA-1; it is a property of composed stochastic systems that CRSA-1 makes visible and manageable. Without the safety budget framework, the same degradation occurs silently—the system becomes less reliable as composition depth increases, but no monitoring detects the accumulation. CRSA-1's contribution is making the degradation explicit, trackable, and enforceable: the system knows its remaining budget, the operator sets the threshold, and the enforcement mechanism triggers when the threshold is crossed.

The implication for system design is that compositional depth has a cost denominated in safety margin. System architects must trade off between the functional benefit of deeper composition (more capable pipelines, longer agent loops, wider ensembles) and the safety cost of budget consumption. CRSA-1 provides the accounting framework; the architectural decisions remain with the system designer.

9.6 CRSA-1 Is Infrastructure, Not Intelligence

The most important limitation is the broadest: CRSA-1 provides governance infrastructure for multi-model AI systems. It provides attestation, composition algebra, isolation bounds, audit logging, and agent interaction constraints. It does not provide:

- **Alignment.** CRSA-1 does not make models more aligned, more helpful, more harmless, or more honest. It attests whether models are operating within certified parameters. A misaligned model that operates within its certified parameters will receive a valid CRSA-1 attestation.
- **Interpretability.** CRSA-1 does not explain why a model produced a specific output. It traces the provenance of the output through the pipeline and attests the configuration under which it was produced.
- **Capability evaluation.** CRSA-1 does not assess whether a model or composed system possesses dangerous capabilities. It attests the system's operational state, not its capability profile.
- **Value learning.** CRSA-1 does not address the problem of learning human values or preferences. It provides infrastructure through which safety properties (however derived) can be composed and attested across model boundaries.

CRSA-1 is to multi-model AI safety what TCP/IP is to network security: it provides the reliable transport layer on which safety mechanisms operate, without itself being a safety mechanism. A network secured by TLS, firewalls, and intrusion detection systems is more secure than an unsecured network, but TCP/IP does not provide the security—it provides the infrastructure through which security is delivered. Similarly, CRSA-1 provides the infrastructure through which compositional safety governance is delivered. The safety itself must come from alignment research, interpretability tools, capability evaluation, safety training, and human oversight operating on top of the CRSA-1 infrastructure.

Honest Framing

A CRSA-1 attestation is a statement of the form: "At time t, the composed system consisting of components $M_1, \ldots, M_k$ was operating in configuration C with safety budget consumption δ_{consumed}, all component contracts were satisfied, the composition was stable ($\rho(\Gamma) < 1$), isolation bounds held, communication channels were constrained, and the full execution provenance is available in the attached DAG." This is a powerful and useful statement. It is not the statement "the system is safe." The gap between these two statements is the gap between infrastructure and intelligence, between process compliance and behavioral assurance, between what governance can provide and what safety requires. CRSA-1 fills the infrastructure gap. The intelligence gap remains open.

10 Regulatory Deadline Mapping

CRSA-1 enters a regulatory landscape where multiple jurisdictions are simultaneously imposing AI governance obligations with hard deadlines, material penalties, and no specification for compositional safety. This section maps the convergence of regulatory deadlines to CRSA-1 conformance levels, establishing that the specification window is approximately 18 months and that CRSA-1 is positioned to fill the compositional safety gap before enforcement creates compliance crises for multi-model deployments.

10.1 Deadline Convergence Timeline

Table 4: Regulatory Deadline Convergence and CRSA-1 Mapping

Date	**Regulatory Event**	**Obligation**	**Penalty**	**CRSA-1 Level**
Jan 1, 2026	California SB 942 effective	Provenance disclosures for AI-generated content	AG enforcement	C0 minimum; C1 recommended
Feb 2, 2026	EU AI Act: Prohibited practices and AI literacy obligations enforceable	Organizations must ensure AI literacy; prohibited AI practices subject to penalty	€35M or 7% global turnover	C0 minimum
∼Jun 2026	NDAA FY2026 §1512: DoD AI cybersecurity policy due (180 days from enactment)	Model tampering, jailbreak, adversarial testing, lineage documentation	Procurement ineligibility	C1 minimum; C2 for classified
Jun 16, 2026	NDAA FY2026 §1513: CMMC-for-AI status report to Congress	Risk-based cybersecurity framework for AI acquisition	Congressional oversight	C2 target
Jun 30, 2026	Colorado AI Act (SB 24-205) effective	Developer documentation of bias testing, limitations; deployer annual impact assessments	AG enforcement	C0 minimum; C1 recommended
Aug 2, 2026	EU AI Act: GPAI model obligations enforceable	Transparency, copyright compliance, risk assessment for systemic risk models	€15M or 3% global turnover	C1 minimum; C2 for systemic risk

Date	Regulatory Event	Obligation	Penalty	CRSA-1 Level
Aug 2, 2026	EU AI Act: High-risk obligations for Annex III systems enforceable	Full compliance with Articles 8–15: risk management, data governance, transparency, human oversight, accuracy, robustness, cybersecurity	€15M or 3% global turnover	C2 minimum
Q4 2026	CEN-CENELEC fast-tracked harmonized standards (target availability)	Standards providing presumption of conformity for EU AI Act	N/A (enables safe harbor)	CRSA-1 positioned as compositional complement
Dec 2, 2027	EU Digital Omnibus Act: Long-stop date for Annex III systems	Compliance mandatory regardless of harmonized standard availability	€15M or 3% global turnover	C2 minimum
Aug 2, 2028	EU Digital Omnibus Act: Long-stop date for Annex I systems	Full AI Act compliance for highest-risk category	€35M or 7% global turnover	C3 target for critical infrastructure

10.2 The 18-Month Window

The timeline reveals a critical structural feature: the period between now (February 2026) and the Digital Omnibus long-stop date (December 2027) is the specification window for compositional AI governance. During this window:

Demand is created. The August 2, 2026 enforcement date for GPAI and high-risk obligations creates immediate demand for compositional safety evidence. Multi-model deployers must demonstrate compliance with Articles 9, 12, and 15 for composed systems. No existing standard addresses this requirement. Every deployer will either adopt a compositional framework, invent their own (producing incomparable, unverifiable ad hoc solutions), or accept the compliance risk.

Supply is absent. The harmonized standards being developed by CEN-CENELEC JTC 21 address component-level requirements: quality management (prEN ISO/IEC 42001 mapping), risk management (prEN ISO/IEC 23894 mapping), data quality, and trustworthiness characteristics. None of them address compositional safety for multi-model systems. Even with the exceptional fast-track procedure approved in October 2025, the earliest availability for these component-level standards is Q4 2026—and they leave the composition gap entirely open.

The safe harbor incentive. The Digital Omnibus Act (November 2025) creates a standards-linked safe harbor: compliance with harmonized standards provides presumption of conformity with the AI Act. For multi-model systems, no harmonized standard provides this presumption because no standard addresses composition. CRSA-1 is positioned to fill this gap—not as a harmonized standard itself (that requires formal CEN-CENELEC adoption) but as the technical specification that a future harmonized standard on compositional AI safety would reference or incorporate.

The DoD procurement catalyst. The NDAA FY2026 Sections 1512 and 1513 create a parallel demand signal in the US defense market. The "CMMC for AI" framework status report due June 16, 2026 will define the cybersecurity requirements for AI acquired by the Department of Defense. Multi-model systems—which are the dominant architecture for complex defense applications including intelligence analysis, logistics optimization, and autonomous systems—will require compositional safety evidence that no current framework provides. CRSA-1's conformance levels map directly to the tiered certification model that CMMC employs for conventional cybersecurity.

10.3 Conformance Level Selection Guide

The regulatory mapping produces a decision framework for conformance level selection:

CRSA-REQ-LEVEL-SELECTION

Organizations **SHOULD** select their target CRSA-1 conformance level based on the following criteria:

- **C0 (Documented):** Appropriate for internal/prototype multi-model systems not subject to external regulatory obligations. Satisfies basic documentation requirements (Colorado SB 24-205, California SB 942). Minimum viable starting point for any organization operating multi-agent systems.
- **C1 (Composed):** Appropriate for production multi-model systems in non-regulated industries, or as the initial compliance tier for regulated industries preparing for higher levels. Satisfies EU AI Act documentation and logging requirements (Articles 11, 12), US state-law requirements, and OMB M-25-21. Recommended starting point for organizations with existing multi-agent deployments.
- **C2 (Attested):** Appropriate for production multi-model systems in regulated industries including financial services, healthcare, and government. Satisfies all current EU AI Act requirements for high-risk and GPAI systems, NDAA FY2026 requirements, SEC/Federal Reserve audit requirements. Target level for enterprise deployment and government procurement eligibility.
- **C3 (Sovereign):** Appropriate for defense, critical infrastructure, and applications where compositional failure consequences are catastrophic or irreversible. Exceeds all current regulatory requirements. Target level for classified systems, autonomous weapons governance, critical infrastructure control, and nuclear/biological safety applications.

The conformance levels are designed to support progressive adoption. An organization can deploy at C0 in Q2 2026 (satisfying initial documentation requirements), upgrade to C1 by Q3 2026 (satisfying EU AI Act August enforcement), and reach C2 by Q1 2027 (establishing full cryptographic attestation before the Digital Omnibus long-stop date). This progression requires incremental investment at each stage rather than a single large-scale deployment, reducing adoption risk and enabling organizations to demonstrate value at each level before committing to the next.

11 References

[1] Alpern, B. and Schneider, F.B. "Defining Liveness." *Information Processing Letters*, 21(4):181–185, 1985.

[2] Alpern, B. and Schneider, F.B. "Recognizing Safety and Liveness." *Distributed Computing*, 2(3):117–126, 1987.

[3] Benveniste, A., Caillaud, B., Nickovic, D., et al. "Contracts for System Design." *Foundations and Trends in Electronic Design Automation*, 12(2–3):124–400, 2018.

[4] Sangiovanni-Vincentelli, A., Damm, W., and Passerone, R. "Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems." *European Journal of Control*, 18(3):217–238, 2012.

[5] Desoer, C.A. and Vidyasagar, M. *Feedback Systems: Input-Output Properties.* Academic Press, 1975.

[6] Jiang, Z.-P., Teel, A.R., and Praly, L. "Small-Gain Theorem for ISS Systems and Applications." *Mathematics of Control, Signals, and Systems*, 7(2):95–120, 1994.

[7] Hammond, L., et al. "Multi-Agent Risks from Advanced AI." Cooperative AI Foundation Technical Report 1, arXiv:2502.14143, February 2025.

[8] Anthropic. "Anthropic's Responsible Scaling Policy, Version 3.0." February 24, 2026.

[9] Xu, J., et al. "Distributional AGI Safety." Google DeepMind, arXiv:2512.16856, December 2025.

[10] Wu, Y., et al. "The AI Agent Index." MIT CSAIL, arXiv:2602.17753, February 2026.

[11] Mathew, A., et al. "Hidden in Plain Text: Emergence & Mitigation of Steganographic Collusion in LLMs." arXiv:2410.03768, updated December 2025.

[12] Morimura, T., et al. "Steganographic Potentials of Language Models." arXiv:2505.03439, May 2025.

[13] Li, Z., et al. "Hide and Seek in Embedding Space: Evaluating and Probing LLM Steganographic Capabilities." arXiv:2601.22818, January 2026.

[14] Gu, Y., et al. "Prompt Infection: LLM-to-LLM Prompt Injection Within Multi-Agent Systems." arXiv:2410.07283, October 2024.

[15] Greshake, K., et al. "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection." Black Hat US 2023.

[16] Sibylline Software. "Structured Output Schemas for Prompt Injection Prevention." Technical Report, February 2026.

[17] Xi, Z., et al. "SEAgent: Understanding Security Risks of LLM-based Agents." arXiv:2601.11893, January 2026.

[18] Acuvity. "Semantic Privilege Escalation in LLM Agent Systems." Technical Report, 2025.

[19] Knostic. "MCP Server Security Scan Results." July 2025.

[20] Backslash Security. "NeighborJack and MCP Server Vulnerabilities." Technical Report, June 2025.

[21] CVE-2025-32711. "EchoLeak: Hidden Prompt Data Exfiltration via MCP." MITRE, 2025.

[22] CVE-2025-6514. "Remote Code Execution via mcp-remote Authorization Endpoint." MITRE, 2025.

[23] Dutta, P., et al. "Spy in the GPU Box: Covert and Side Channel Attacks on Multi-GPU Systems." ACM CCS 2023.

[24] NVBleed Authors. "NVBleed: Cross-GPU Side Channel via NVLink." 2025.

[25] Tang, Z., et al. "PROMPTPEEK: Prompt Extraction via Shared KV Cache Timing." NDSS 2025.

[26] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and Pan, W. "Remote ATtestation procedureS (RATS) Architecture." RFC 9334, Internet Engineering Task Force, January 2023.

[27] Lundblade, L., Mandyam, G., O'Donoghue, J., and Tschofenig, H. "The Entity Attestation Token (EAT)." RFC 9711, Internet Engineering Task Force, December 2024.

[28] Jones, M., Wahlstroem, E., Erdtman, S., and Tschofenig, H. "CBOR Web Token (CWT)." RFC 8392, Internet Engineering Task Force, May 2018.

[29] Schaad, J. "CBOR Object Signing and Encryption (COSE)." RFC 9052, Internet Engineering Task Force, August 2022.

[30] Birkholz, H., Vigano, C., and Brossard, C. "CDDL." RFC 8610, Internet Engineering Task Force, June 2019.

[31] Messous, M.A., et al. "EAT Claims for AI Model Identity and Provenance." draft-messous-eat-ai-00, IETF Internet-Draft, 2025.

[32] Jiang, X., et al. "Dynamic Attestation for Runtime Posture." draft-jiang-seat-dynamic-attestation-00, IETF Internet-Draft, 2025.

[33] Huang, W., et al. "Agentic EAT Capability Attestation." draft-huang-rats-agentic-eat-cap-attest-00, IETF Internet-Draft, 2025.

[34] Aylward, M., et al. "Tiered AI Governance Attestation." draft-aylward-aiga-1-00, IETF Internet-Draft, 2025.

[35] Deshpande, Y., et al. "Multi-Verifier Topological Attestation." draft-deshpande-rats-multi-verifier-03, IETF Internet-Draft, 2025.

[36] Bradner, S. "Key Words for Use in RFCs to Indicate Requirement Levels." RFC 2119, Internet Engineering Task Force, March 1997.

[37] Leiba, B. "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words." RFC 8174, Internet Engineering Task Force, May 2017.

[38] Moreau, L. and Missier, P. "PROV-DM: The PROV Data Model." W3C Recommendation, April 2013.

[39] Nian, Y. and Shanahan, M. "PROV-AGENT: Provenance for Agentic AI Workflows." arXiv:2508.02866, 2025.

[40] OCSF Project. "Open Cybersecurity Schema Framework, Version 3.1.0." Linux Foundation, 2025.

[41] European Parliament and Council. "Regulation (EU) 2024/1689 (AI Act)." *Official Journal of the European Union*, August 2024.

[42] European Parliament and Council. "Digital Omnibus Act." November 2025.

[43] CEN and CENELEC Boards. "Exceptional Fast-Track Procedure for AI Act Harmonized Standards." Decision of October 14–16, 2025.

[44] US Congress. "National Defense Authorization Act for Fiscal Year 2026." Signed December 18, 2025.

[45] Colorado General Assembly. "SB 24-205: Consumer Protections for Artificial Intelligence." Effective June 30, 2026.

[46] California Legislature. "SB 942: AI Transparency Act." Effective January 1, 2026.

[47] Office of Management and Budget. "Memorandum M-25-21: AI Governance and Data Traceability." April 2025.

[48] US Securities and Exchange Commission. "Rule 17a-4: Records to be Preserved by Certain Exchange Members, Brokers, and Dealers."

[49] Board of Governors of the Federal Reserve System. "SR 11-7: Guidance on Model Risk Management." April 2011.

[50] Fields, R. "The Model State Attestation Framework (MSAF)." Auburn Governance Stack, Document 1, 2026.

[51] Fields, R. "The Model Attestation Interface (MAI-1)." Auburn Governance Stack, Clause AI-5, Document 22, 2026.

[52] Fields, R. "Auburn Governance Stack Master Architecture Plan (AGS-1)." Auburn Governance Stack, Document 23, 2026.

[53] Fields, R. "Stateful Isolation Law." Auburn Governance Stack, Document 5, 2026.

[54] Fields, R. "AI-4: SRAM Thermal Integrity Bound." Auburn Governance Stack, Document 3, 2026.

[55] Fields, R. "AI-2: Gradient Starvation Envelope." Auburn Governance Stack, Clause AI-2, Document 8, 2026.

[56] Fields, R. "AI-3: Lyapunov Stability for Speculative Decoding." Auburn Governance Stack, Clause AI-3, Document 9, 2026.

[57] Fields, R. "AI-8: Entropy Collapse Constraint." Auburn Governance Stack, Clause AI-8, Document 10, 2026.

[58] Fields, R. "CTS-1: MAI-1 Conformance Test Suite." Auburn Governance Stack, Document 26, 2026.

[59] Fields, R. "Rails Symposium: Capstone Tutorial." Auburn Governance Stack, Document 2, 2026.

