

A Typed Delegation-Governance Tuple Profile for Multi-Agent Runtime Control

Reuben Bowlby

HUMMBL, LLC

reuben@hummbbl.io

ORCID: 0009-0002-5620-1103

Abstract—Multi-agent AI systems lack composable resilience primitives that span delegation, execution, and verification. Existing standards address provenance, attestation, or authorization individually, but no unified profile binds these concerns—together with runtime safety mechanisms such as circuit breakers and kill switches—across a delegation tree with fail-closed enforcement. We present the Intelligent Delegation Profile (IDP): a typed six-tuple governance model comprising Delegation Context (DCTX), Contract (CONTRACT), Delegation Capability Token (DCT), Evidence (EVIDENCE), Attestation (ATTEST), and System Event (SYSTEM) tuples. Cross-linked identifiers connect tuples across the delegation tree, and three lifecycle invariants—evidence-before-verify (I1), monotonic capability attenuation (I2), and bounded delegation depth (I3)—are enforced at runtime for the invariants currently implemented (see Section VI-G for disclosed enforcement gaps). We report operational measurements from a production multi-agent system that orchestrates four agents across six active service adapters, producing a mean of 10,013 signed governance entries per active day (median 847, range 33–26,014 over 15 active days). We further demonstrate the profile’s composability by extending it to a multi-agent peer review protocol—including bias-aware reviewer selection, adaptive verification cascades, and Beta reputation trust scoring—without modifying the underlying tuple schema.

Index Terms—multi-agent systems, delegation governance, capability tokens, runtime assurance, composable trust

I. INTRODUCTION

As multi-agent AI systems move from research prototypes to production deployments, a critical gap has emerged: the absence of composable resilience primitives that operate across delegation boundaries at runtime, integrating governance with fail-closed safety mechanisms (circuit breakers, kill switches, bounded delegation depth). A human delegates a task to an orchestrating agent, which subdelegates to specialized agents, each invoking external tools with varying privilege requirements. Governing this delegation tree requires answering several questions simultaneously: What was the agent authorized to do? What did it actually do? Who verified the result? How do we reconstruct the full chain after the fact?

Existing work addresses fragments of this problem. W3C PROV [1] models provenance relationships but lacks delegation authority semantics. IETF RATS [2] separates evidence from attestation but does not bind these to task-scoped capability tokens. Macaroons [3] and Biscuit [4] provide attenuated authorization tokens but carry no lifecycle state or verification linkage. Google DeepMind’s Intelligent Delegation framework [5] describes contract-first decomposition,

privilege attenuation, and attestation chains, but does not define a concrete tuple schema with cross-linked identifiers and runtime enforcement semantics.

We contribute a typed six-tuple governance profile—the Intelligent Delegation Profile (IDP)—that synthesizes these prior strands into a unified model with three distinguishing properties:

- 1) **Cross-linked identifiers.** Tuples carry cross-link identifiers where applicable: `intent_id`, `task_id`, and optionally `contract_id`, `capability_token_id`, and `verification_id`, enabling full delegation-tree reconstruction from any single entry.
- 2) **Lifecycle invariants.** The profile enforces structural constraints including evidence-before-verify (an ATTEST tuple cannot reference nonexistent EVIDENCE) and monotonic capability attenuation (child DCTs can only reduce, never expand, the parent’s scope).
- 3) **Runtime enforcement.** The profile is not a design-time specification alone; it is enforced at execution time in a production multi-agent system, with signed entries rejected if they violate invariants.

We report operational evidence from two deployments: a daily executive briefing system orchestrating four AI agents across over 100 service modules with over 5,000 automated tests (six of seven adapters active in the evaluated configuration), and a multi-agent peer review protocol that extends the IDP with bias-aware reviewer selection, adaptive verification cascades, and domain trust aggregation—without changing the underlying tuple schema.

II. BACKGROUND AND RELATED WORK

A. Tuple Spaces, Provenance, and Attestation

Linda [6] introduced tuple-based coordination; tuple centres [7], [8] extended this with programmable coordination media. These provide general-purpose coordination substrates but do not encode governance or verification semantics. The IDP is not a tuple space; it uses typed tuple records as its representation format.

W3C PROV [1] models provenance as entity-activity-agent relationships. DCTX, EVIDENCE, and ATTEST tuples can be partially projected into PROV, but PROV lacks delegated authority, contract-bound permissions, and a delegation state

machine. IETF RATS [2] separates evidence from attestation results, structurally aligned with the IDP’s EVIDENCE and ATTEST tuples, but provides no task-local capability attenuation or lifecycle control. The in-toto framework [9] addresses provenance over supply chain workflows but operates at the build/release level rather than runtime delegation.

B. Object-Capability Security

The concept of a capability as a combined designation and authority was first formalized by Dennis and Van Horn [31]. The IDP’s DCT is a direct descendant of this model: each token is an unforgeable designator that simultaneously specifies *what* the agent can access and *how much* authority it carries, without requiring a central reference monitor.

Hardy’s “Confused Deputy” problem [32] identifies a class of vulnerabilities where a privileged program is manipulated into exercising its authority on behalf of an unauthorized principal. DCTs mitigate this by binding authority to specific task identifiers and contract schemas: the deputy (the sub-agent) can only act within the cryptographic scope of the delegating DCT, structurally preventing cross-task privilege escalation.

Miller et al. [33], [34] formalized the Principle of Least Authority (POLA) and demonstrated its natural expression through capability-based systems. The IDP’s monotonic attenuation invariant (I2) operationalizes POLA directly: every sub-delegation strictly reduces the capability set, realizing the “Robust Composition” guarantee that secure decomposition is achievable without global coordination.

SPIFFE/SPIRE. The SPIFFE standard (Secure Production Identity Framework for Everyone) [44] and its SPIRE reference implementation provide workload identity attestation through cryptographically verifiable SVIDs (SPIFFE Verifiable Identity Documents). SVIDs are X.509 or JWT tokens issued to workloads after platform attestation, enabling mutual TLS between services. The SPIFFE trust domain model is structurally analogous to the IDP’s EVIDENCE/ATTEST chain: a workload presents evidence of its identity (the SVID), and a relying party attests that the identity is valid within the trust domain. However, SPIFFE operates at the *workload identity* layer, not the *delegation governance* layer. SVIDs attest *who* a workload is, not *what* it is authorized to do, *under what contract*, or *with what execution evidence*. SPIRE’s attestation model could validate the `subject` field of a DCT (confirming that the agent presenting the token is the workload it claims to be), but it does not replace the governance envelope: contract scope, capability subsetting, evidence-before-verify, and cross-linked delegation-tree traversal. The IDP and SPIFFE/SPIRE are complementary layers: SPIFFE provides cryptographic workload identity, the IDP provides governance semantics for what that identity is authorized to do.

C. Authorization Token Systems

Macaroons [3] introduced caveat-based authorization tokens with delegated attenuation, enabling offline-verifiable confinement. Biscuit [4] extended this with a datalog-based autho-

rization language. The IDP’s DCT is adjacent to this family—it provides HMAC-SHA256 signed capability tokens with monotonic attenuation. The distinction is that a DCT is not a standalone authorization primitive; it is structurally bound to a CONTRACT (declaring task scope), a DCTX (tracking lifecycle state), and an eventual EVIDENCE/ATTEST chain (recording what happened and who verified it). This binding is the governance profile’s contribution beyond the authorization token itself.

Concurrent work by Prakash [39] introduces Invocation-Bound Capability Tokens (IBCTs), a similar primitive that fuses identity, attenuated authorization, and provenance binding into an append-only token chain with both compact (JWT) and chained (Biscuit/Datalog) wire formats. IBCTs operate across MCP, A2A, and HTTP transports with 0.189ms Python verification overhead. The IDP’s DCT is structurally adjacent to IBCTs; the distinction is that a DCT is embedded in a typed six-tuple governance profile (DCTX, CONTRACT, DCT, EVIDENCE, ATTEST, SYSTEM) with cross-linked identifiers and a lifecycle state machine—where IBCTs provide the token primitive, the IDP provides the governance envelope binding delegation, evidence, and verification into a traversable audit chain.

UCAN. User Controlled Authorization Networks (UCAN) [43] provide capability-based token delegation with offline-verifiable attenuation chains and audience-scoped invocation. A UCAN token carries a chain of delegations, each attenuating the previous, with each delegation signed by the issuer’s DID. This is structurally close to the IDP’s DCT chain. Three distinctions separate the two. First, UCAN operates at the authorization layer: it specifies *who* can do *what*, but does not define a governance envelope for contracts, evidence, or attestation. A UCAN could carry contract and evidence identifiers as resource pointers, but the specification does not prescribe this binding as a normative profile. Second, UCAN’s attenuation is chain-of-custody (each hop signs the next), whereas the IDP’s I2 invariant is semantic subsetting: child capabilities must be a subset of parent capabilities, enforced by comparing allowed-operation sets at DCT creation time. UCAN’s chain structure does not enforce semantic subsetting—a malicious issuer could sign a child delegation with expanded capabilities. Third, UCAN does not specify a lifecycle state machine: there is no PROPOSED→ISSUED→RUNNING→EVIDENCE_READY→VERIFIED progression, and no evidence-before-verify invariant. The IDP’s governance profile can be understood as layering contract-scope declaration, execution evidence binding, and verification attestation on top of the authorization-token primitive that UCAN provides.

D. Intelligent Delegation

Google DeepMind’s Intelligent Delegation framework [5] describes contract-first decomposition, privilege attenuation, Delegation Capability Tokens, monitoring streams, verifiable completion, and attestation chains. This represents the strongest conceptual alignment with the IDP. What it does

not provide is a concrete named tuple schema (DCTX, CONTRACT, DCT, EVIDENCE, ATTEST, SYSTEM), explicit cross-link identifiers connecting tuples across the delegation tree, or operational evidence from a production deployment. The IDP can be understood as one concrete realization of the Intelligent Delegation vision, formalized as a typed governance profile with runtime enforcement semantics.

E. Emerging Agent Protocol Standards

Two agent communication protocols are converging under the Linux Foundation’s Agentic AI Foundation (AAIF, December 2025): Google’s Agent-to-Agent protocol (A2A) and Anthropic’s Model Context Protocol (MCP). A2A defines Agent Cards for identity and HTTP-layer authentication but has no delegation chain semantics, typed governance tuples, or append-only audit trail [15]. MCP defines tool discovery and OAuth-based authorization but explicitly acknowledges audit trails as an unsolved enterprise gap in its 2026 roadmap [16]. Neither protocol defines a governance envelope for delegation, evidence, or attestation.

The OIDC-A 1.0 proposal [17] extends OpenID Connect with agent-specific claims: agent type, model version, provider identity, and notably a `delegation_chain` claim that is structurally similar to the IDP’s DCTX chain. However, OIDC-A operates at the JWT/OAuth identity layer, not as a typed governance envelope binding contracts, evidence, and attestations to delegation context.

Louck et al. [40] provide a systematic analysis of A2A’s security limitations—insufficient token lifetime control, overbroad access scopes, and absent consent flows—that directly motivate the IDP’s bounded delegation model. The IDP’s DCT lifetime bound (time-to-live field) and DCTX state machine address the token lifetime problem; the monotonic attenuation invariant (I2) addresses scope creep by ensuring child delegations cannot exceed their parent’s capability set.

The IDP is complementary to these protocols rather than competitive. A2A and MCP provide transport and tool discovery; OIDC-A provides identity claims; the IDP provides the governance layer that binds delegation authority, execution evidence, and verification attestations across any of these transports. Recent industry efforts reinforce this layering: Okta’s Cross App Access (XAA) protocol extends MCP with enterprise-managed authorization via ID-JAG (Identity Assertion JWT Authorization Grant), providing IdP-mediated scoping for agent-to-app connections [22]. XAA addresses the authorization plane that MCP acknowledged as a gap, but like OIDC-A it operates at the identity/authorization layer rather than providing the full governance envelope (contract, evidence, attestation) that the IDP addresses.

F. Agent Governance Frameworks

Production multi-agent frameworks (AutoGen, CrewAI, LangGraph, OpenAI Swarm) implement delegation through conversation routing, role assignment, or graph-based execution, but none provides typed governance tuples, cross-linked audit trails, or runtime-enforced lifecycle invariants. The IDP

is designed to layer beneath such frameworks, providing governance infrastructure that is transport- and orchestration-agnostic. Concurrent work addresses agent governance from complementary directions. The Agent Contracts framework [23] defines a formal tuple $C = (I, O, S, R, T, \Phi, \Psi)$ that unifies input/output specifications with resource constraints, temporal boundaries, and success criteria. It demonstrates $90\times$ token reduction with $525\times$ lower variance in constrained workflows and zero conservation violations in delegation hierarchies. Where Agent Contracts focus on resource governance and budget discipline, the IDP focuses on delegation authority, evidence binding, and verification attestation—the two are complementary and could compose.

The Auton Agentic AI Framework [24] separates a declarative Cognitive Blueprint (agent specification) from a Runtime Engine (execution substrate), analogous to infrastructure-as-code. It provides agent specification as versionable, auditable data artifacts but does not define cross-linked governance tuples or lifecycle invariants for delegation chains.

The Layered Governance Architecture (LGA) [25] proposes four layers: execution sandboxing (L1), intent verification (L2), zero-trust inter-agent authorization (L3), and immutable audit logging (L4). The IDP maps cleanly onto this layering: DCTs provide L3 authorization, the governance bus provides L4 audit, and the CONTRACT/ATTEST chain provides L2 verification. The IDP additionally provides cross-link identifiers that connect all four layers into a traversable delegation tree.

SAGA [26] provides cryptographic access control tokens for inter-agent communication with formal security guarantees (accepted at NDSS 2026). SAGA focuses on binary access control; the IDP extends this with typed capability semantics (tool lists, budgets, time-to-live) and cross-linked governance tuples that bind delegation to evidence and attestation. Agentic JWT (A-JWT) [27] proposes dual-faceted intent tokens binding agent actions to workflow steps with per-agent proof-of-possession keys. A-JWT’s cryptographic identity binding is stronger than our HMAC-SHA256 approach; our typed tuple profile provides richer governance semantics beyond the authorization layer.

The MI9 Agent Intelligence Protocol [28] defines six runtime governance components: Agency-Risk Index, Agent-Semantic Telemetry, Continuous Authorization Monitoring, a Real-Time Conformance Engine, Behavioral Drift Detection, and Graduated Containment. MI9’s components are observational and enforcement-oriented—they monitor, detect, and contain. The IDP’s six tuples are structural and delegation-oriented—they define what authority was granted, under what contract, with what evidence of completion. MI9 has no delegation model, no token issuance, and no chain attenuation; the two approaches are complementary.

Wang et al. [41] introduce the NANDA (Networked AI Agents in a Decentralized Architecture) framework, implementing Zero Trust Agentic Access (ZTAA) principles with cryptographically verifiable capability attestation via Agent-Facts across MCP, A2A, and HTTP. NANDA addresses

agent discovery and cross-protocol interoperability; the IDP addresses the complementary problem of what governance envelope should bind each agent invocation once discovery is complete.

Khan et al. [42] propose AGENTS SAFE, a practical governance framework that operationalizes AI risk taxonomies into design, runtime, and audit controls for LLM-based agentic systems—including dynamic authorization, cryptographic tracing, and interruptibility mechanisms. AGENTS SAFE and the IDP are architecturally complementary: AGENTS SAFE provides a governance decision pipeline; the IDP provides the typed delegation and attestation primitives that such a pipeline requires.

The Governance-as-a-Service (GaaS) framework [29] intercepts agent outputs and applies a deterministic Trust Factor computed as a weighted penalty sum across normative, coercive, and adaptive governance modes with exponential severity decay. The IDP’s Beta-distribution trust model is Bayesian and probabilistic, naturally expressing uncertainty about new agents (wide prior) versus established ones (tight posterior). More fundamentally, GaaS governs what agents *say* by pattern-matching outputs; the IDP governs what agents are *authorized to do* via delegation tokens, cross-linked evidence, and lifecycle invariants.

DelegateOS [30] provides Ed25519-signed delegation tokens with monotonic capability attenuation in TypeScript, drawing on both Biscuit and DeepMind’s Intelligent Delegation framework. DelegateOS is the closest structural comparison: it solves scoped delegation with attenuation and accountability. The IDP differentiates through cross-linked six-tuple governance (where EVIDENCE must reference its CONTRACT and ATTEST must reference its EVIDENCE), zero-dependency implementation (Python stdlib with HMAC-SHA256), and composable multi-agent peer review attestation with configurable quorum—features absent from DelegateOS’s single-agent attestation model.

G. Agentic and LLM Authorization

The shift to autonomous LLM agents has exposed a “trust-authorization mismatch” [38]: static permission models such as RBAC fail to account for the dynamic, probabilistic nature of agent reasoning, where capabilities must be scoped just-in-time to the task at hand rather than declared at deployment time.

South et al. [35] propose cryptographic delegation chains linking humans to agents via extensions to OAuth 2.0 and OpenID Connect. The IDP implements the specific delegation token mechanism described there, adding monotonic capability attenuation and task-bound contract schemas as enforcement primitives absent from the OAuth-layer proposal.

Jiang et al. [36] examine full end-to-end delegation of complex planning tasks to LLM agents, identifying the behavioral requirements for an agent to act safely under complete user delegation. The IDP provides the cryptographic enforcement layer for such delegation contracts, ensuring that delegated

authority cannot exceed the original mandate across recursive subdelegation hops.

Progent [37] introduces programmable privilege control for LLM tool calls, gating sensitive operations at the invocation level. The IDP’s DCT provides the signed, task-bound token that Progent-style systems can verify as an execution precondition, composing fine-grained privilege control with a full delegation-and-attestation chain.

H. Multi-Agent Verification and LLM-as-Judge

Recent work on LLM-as-Judge evaluation [12] identifies systematic biases when language models evaluate each other’s outputs: position bias (preference for outputs presented first or last), verbosity bias (preference for longer outputs regardless of quality), and self-enhancement bias (models systematically favor their own generations, with self-preference strength linearly correlated with self-recognition capability [13]). These biases are directly relevant to multi-agent governance, where agents must verify each other’s work.

Lifshitz et al. [14] propose Multi-Agent Verification (MAV), demonstrating that verification accuracy improves when multiple heterogeneous verifiers are used, and that verification depth can be adaptively scaled based on confidence—investing more verification compute on uncertain outputs. This insight motivates adaptive governance: not all agent outputs require the same level of scrutiny.

The IDP’s composability allows it to absorb these findings without schema modification. The peer review protocol (Section IV-B) implements bias-aware reviewer selection and adaptive verification cascades as SYSTEM tuple metadata on the existing governance bus, demonstrating that research advances in multi-agent evaluation can be operationalized within a typed governance framework.

I. Synthesis: Why Composition Matters

One might ask whether the IDP is merely Macaroons plus PROV. It is not, because no single prior system spans the full governance lifecycle. Macaroons and Biscuit [3], [4] provide capability attenuation but no lifecycle state machine, no evidence binding, and no trust scoring. PROV [1] captures provenance but is descriptive (post-hoc reconstruction), not prescriptive (fail-closed runtime enforcement). OI DC-A [17] handles agent identity but not delegation depth or capability subsetting. UCAN [43] provides cryptographic attenuation chains but no semantic capability subsetting, no contract binding, and no evidence-before-verify invariant. SPIFFE/SPIRE [44] attests workload identity but does not govern what that identity is authorized to do. The IDP’s contribution is the *typed composition*: binding delegation context, contracts, capability tokens, evidence, and attestation into a single governance envelope with runtime-enforced invariants (I1, I2, I3) and cross-linked identifiers. No prior system enforces all three invariants across a delegation tree while maintaining a signed, append-only audit trail.

III. THE IDP TUPLE PROFILE

A. Tuple Types

The IDP defines six typed tuple classes, each with specific governance semantics:

TABLE I
IDP TUPLE TYPES

Type	Semantics	Key Fields
DCTX	Delegation life-cycle state	parent_dctx_id, chain_depth, status, timeout
CONTRACT	Task scope and constraints	objective, allowed_tools, acceptance_criteria
DCT	Capability-bounded authorization	allowed_adapters, invocation_budget, cost_ceiling, ttl, signature
EVIDENCE	Execution receipt	tool_outputs, adapter_results, timing_data, content_hash
ATTEST	Verification result	verdict, linked_evidence_id, reviewer_identity, confidence
SYSTEM	Runtime event	circuit_breaker_state, kill_switch_mode, health_probe_results

B. Normative Tuple Schema

Table I illustrates the six tuple types at a conceptual level. For reproducibility, the IDP specifies a normative JSON Schema (Draft 2020-12) that defines mandatory vs. optional fields, value types, and structural constraints for each tuple type. Table II summarizes the schema for each type; the complete machine-readable schema is included in supplementary material.

The schema enforces three structural constraints beyond per-field typing. **C1 (Referential Integrity):** `ATTEST.verification_id` must reference an EVIDENCE tuple that exists in the governance bus at the time of ATTEST submission; the bus rejects ATTEST tuples with dangling references (I1 enforcement). **C2 (Semantic Subsetting):** `DCT.ops_allowed` must be a subset of the parent DCT’s `ops_allowed` at DCT creation time; the delegation context rejects creations violating this (I2 enforcement). **C3 (Depth Bounding):** `DCTX.chain_depth` must be $\leq \text{MAX_DEPTH}$; the state machine rejects transitions that would exceed the bound (I3 enforcement). These constraints are enforced at write time, not validated post-hoc.

C. Cross-Link Identifiers

IDP tuples carry cross-link identifiers where applicable, connecting them to the broader delegation tree:

- `intent_id`—links to the originating agent intent declaration

- `task_id`—identifies the specific delegation task
- `contract_id`—links to the governing CONTRACT tuple
- `capability_token_id`—links to the authorizing DCT
- `verification_id`—links ATTEST tuples to their supporting EVIDENCE

These identifiers enable bidirectional traversal: given any single tuple, the complete delegation context can be reconstructed by following cross-links. This property is critical for post-hoc audit and for real-time governance enforcement.

D. Lifecycle Invariants

The IDP enforces three structural invariants at runtime. Each invariant is formalized as a predicate over tuple fields, with an enforcement mechanism that rejects violations at write time (Fig. 1 annotates the I1 and I3 enforcement points on the state machine; I2 is enforced at DCT creation time, outside the state machine).

Invariant I1 (Evidence-before-verify). Let $\mathcal{E}$ denote the set of EVIDENCE tuples in the governance bus at time t , and let A be an ATTEST tuple submitted at t . I1 requires:

$$A.verification_id \in \{e.evidence_id \mid e \in \mathcal{E}\} \quad (1)$$

Enforcement. The governance bus query layer checks `A.verification_id` against the evidence index before accepting the ATTEST append. If no matching EVIDENCE exists, the bus returns error code `IDP_E_EVIDENCE_REQUIRED` and rejects the entry. The check is atomic within the bus’s append lock, preventing race conditions where an EVIDENCE tuple is in flight but not yet committed. The evidence index is an in-memory hash set keyed by `evidence_id`, rebuilt from the log on bus initialization; query latency is $O(1)$.

Invariant I2 (Monotonic capability attenuation). Let D_p be the parent DCT and D_c the child DCT under creation. I2 requires:

$$D_c.ops_allowed \subseteq D_p.ops_allowed \wedge D_c.resource_selectors \subseteq D_p.resource_selectors \quad (2)$$

where resource selector containment is defined as: for every selector $s_c \in D_c.resource_selectors$, there exists $s_p \in D_p.resource_selectors$ with $s_c.resource_type = s_p.resource_type$ and $s_c.constraints$ no less restrictive than $s_p.constraints$ (i.e., child constraints must imply parent constraints). Caveats and time-to-live follow the same monotonicity: child caveats must be a superset of parent caveats (more restrictive), and child TTL $\leq$ parent TTL. **Enforcement.** The `DelegationContext.create_child` method computes set containment before issuing D_c . If either condition fails, it returns error code `IDP_E_CAPABILITY_ESCALATION` and no child context is created. This is the sole entry point for child DCT creation; verification by inspection confirms no bypass paths exist. The check is deterministic and requires no external state.

TABLE II
NORMATIVE IDP TUPLE SCHEMA SUMMARY (MANDATORY ^M, OPTIONAL ^O)

Tuple	Field	Type	Card.	Constraint
DCTX	intent_id	UUIDv4 string	M	Matches root intent
	task_id	UUIDv4 string	M	Unique per delegation event
	parent_task_id	UUIDv4—null	O	Null for root; non-null for child
	chain_depth	integer ≥ 0	M	$\leq \text{MAX_DEPTH}$ (default 3)
	status	enum	M	One of: PROPOSED, ISSUED, RUNNING, EVIDENCE_READY, VERIFIED, REPLANNED, FAILED
CONTRACT	budget	object	O	max_tokens, max_cost_usd, max_wall_time
	contract_id	UUIDv4 string	M	Links to governing contract
	objective	string	M	Non-empty, ≤ 4096 chars
	allowed_tools	string[]	M	Non-empty list of tool identifiers
	acceptance_criteria	string[]	O	Checklist for verification
DCT	expiry	ISO8601—null	O	Null = no expiry
	token_id	UUIDv4 string	M	Unique per token
	issuer	agent-id string	M	Registered agent identity
	subject	agent-id string	M	Must differ from issuer
	ops_allowed	string[]	M	Subset of parent DCT's ops_allowed (I2)
EVIDENCE	resource_selectors	object[]	O	resource_type, resource_id, constraints
	caveats	object[]	O	TIME_BOUND, RATE_LIMIT, APPROVAL_REQUIRED, AUDIT_REQUIRED
	signature	HMAC-SHA256 hex	M	Computed over canonical JSON of all other fields
	evidence_id	UUIDv4 string	M	Unique per evidence tuple
	ops_executed	string[]	M	Subset of CONTRACT's allowed_tools
ATTEST	resources_accessed	object[]	O	Must match CONTRACT selectors
	cost	object	O	tokens, usd, wall_time_ms
	output_hash	SHA-256 hex	O	Content-addressed output
	verification_id	UUIDv4 string	M	Must reference existing EVIDENCE (I1)
	verdict	enum	M	PASS, FAIL, NEEDS_REVIEW
SYSTEM	reviewer_id	agent-id string	M	Registered agent identity
	confidence	float	M	[0.0, 1.0]
	linked_evidence_id	UUIDv4 string	M	Same as verification_id (redundancy for traversal)
SYSTEM	event_type	enum	M	CIRCUIT_BREAKER, KILL_SWITCH, HEALTH, MISC
	event_data	object	M	Type-specific payload
	severity	enum	O	INFO, WARN, ERROR, CRITICAL

Invariant I3 (Bounded delegation depth). Let δ denote MAX_DEPTH (currently 3). For any DCTX x in a delegation tree rooted at r :

$$x.\text{chain_depth} = \text{distance}(r, x) \wedge x.\text{chain_depth} \leq \delta \quad (3)$$

where $\text{distance}(r, x)$ is the number of parent-child edges from root r to x . **Enforcement.** The DCTX state machine increments chain_depth on every ISSUED→RUNNING transition. If the incremented value exceeds δ , the transition is rejected with error code $\text{IDP_E_DEPTH_EXCEEDED}$. Because depth is monotonically non-decreasing along any path from root to leaf, this single transition check is sufficient: a child cannot be created with depth $d+1$ from a parent at depth d if $d+1 > \delta$.

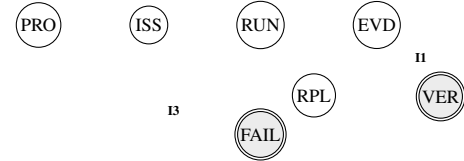


Fig. 1. DCTX state machine. PRO=Proposed, ISS=Issued, RUN=Running, EVD=Evidence_Ready, VER=Verified, RPL=Replanned, FAIL=Failed. Double borders mark terminal states. I1 (evidence-before-verify) and I3 (bounded depth) enforcement points annotated.

E. Delegation Tree Structure

A complete IDP delegation tree for a single execution run follows this structure:

Root DCTX (human authorization)

```

|-- Root CONTRACT (task objective)
|-- Root DCT (full capability set)
|-- Child DCTX_1 (adapter invocation)
| |-- Child CONTRACT_1 (adapter scope)
| |-- Child DCT_1 (attenuated caps)
| |-- EVIDENCE_1 (execution receipt)
| \-- ATTEST_1 (verification result)
|-- Child DCTX_2 ... DCTX_n
|-- SYSTEM (circuit breaker, health)
\-- Root ATTEST (terminal verification)

```

All tuples are written to an append-only JSONL governance bus. Each entry is HMAC-SHA256 signed. The bus implementation rejects unsigned entries, enforcing cryptographic integrity as a default rather than an opt-in.

F. Append-Only Audit Semantics

IDP tuples are never deleted or modified after emission. Corrections are recorded as new tuples referencing the original via an `amendment_of` field; the governance bus structurally validates that the referenced entry exists before accepting the amendment (see Section VI-G for implementation details). This append-only property, combined with HMAC signatures and cross-linked identifiers, means the governance bus constitutes a reconstructable, tamper-evident audit trail of the complete delegation lifecycle.

IV. COMPOSABILITY EVIDENCE

A governance profile’s value depends on whether it can be extended to new concerns without modifying its core schema. We present two case studies demonstrating the IDP’s composability.

A. Case Study: Executive Briefing System

Our primary deployment is a daily executive briefing system that orchestrates four AI agents (spanning four model families) across seven external service adapters: source control (CI/PR status), calendar (OAuth2), issue tracker (project management), cost tracker (budget governance), security scanner (static analysis), messaging (secure delivery), and local LLM inference. Six adapters are active in the evaluated deployment; the local LLM adapter is available but disabled on the evaluation hardware due to GPU constraints. The system comprises over 100 service and integration modules, validated by over 5,000 automated tests across unit, integration, end-to-end, chaos, and security test suites. It uses zero third-party runtime dependencies (Python stdlib only), making the governance implementation fully auditable without dependency-chain risk.

A representative briefing run produces 32 signed governance entries (1 root DCTX + 6 child DCTX records, 7 CONTRACT tuples, 6 DCT tokens, 6 EVIDENCE tuples, 4 ATTEST tuples, 2 SYSTEM tuples). Table III presents summary statistics across 15 active days over a 25-day evaluation period.

The wide range (33–26,014 entries/day) reflects workload variance: low-activity maintenance days produce few tuples, while peak sprint days with multiple agent sessions and adapter retries generate many. The mean is skewed upward by the 26,014 outlier; the median (847) better represents a

TABLE III
GOVERNANCE BUS VOLUME STATISTICS (15 ACTIVE DAYS)

Statistic	Entries/day	Contract/day	DCT/day
Mean	10,013	42	38
Median	847	8	7
Min	33	2	2
Max	26,014	112	98
Total (15 days)	150,192	635	567

typical day. The root delegation carries the full capability set; each child delegation receives an attenuated DCT listing only the specific adapter it is authorized to invoke, with a per-adapter invocation budget and cost ceiling. A child delegated to invoke the source control adapter cannot invoke the calendar adapter, even if the agent’s underlying model has the technical capability to do so. This is structural least-privilege enforcement, not policy documentation.

B. Case Study: Peer Review Protocol

To test whether the IDP can be extended without schema modification, we designed a multi-agent peer review protocol (now at v0.7, validated with 132 tests) that governs how AI agents review each other’s outputs before acceptance.

The protocol introduces six new governance concerns—multi-agent verification, trust scoring, collusion defense, risk-tiered escalation, bias-aware reviewer selection, and adaptive verification depth—on top of the existing IDP. The mapping to IDP tuple types is direct:

TABLE IV
PEER REVIEW TO IDP TUPLE MAPPING

Peer Review	IDP Tuple	Rationale
PROPOSAL	CONTRACT	Binds scope and acceptance criteria
REVIEW verdict	ATTEST	Reviewer’s verification result
Standards check	EVIDENCE	Automated verification receipt
DECISION	ATTEST	Terminal adjudication
Trust snapshot	SYSTEM	Reviewer trust telemetry
Reviewer assign	SYSTEM	COI check, diversity enforcement
Cascade depth	SYSTEM	Adaptive verification level
Amended review	ATTEST	Append-only correction

The protocol adds a Beta reputation trust model [10] computed as a projection over accumulated ATTEST tuples, with 30-day half-life decay and explicit uncertainty quantification (uncertainty = $2/(\alpha + \beta)$, following Josang’s subjective logic metric rather than statistical variance). We chose per-agent Beta distributions over global reputation approaches such as EigenTrust, which computes a single trust eigenvector over the network graph. The per-agent model supports domain-specific

trust (Section IV-B) and avoids transitive trust propagation, which is undesirable in adversarial multi-agent settings where collusion paths should not amplify trust. Risk-tiered review depth spans 4 tiers, from automated-only to human-in-the-loop. Heterogeneous reviewer pools require at least two model families for elevated-risk reviews, and mandatory verifier gates defend against collusion [11].

Three additional mechanisms address known failure modes in LLM-as-Judge evaluation [12]:

Bias-Aware Reviewer Selection. A `ReviewerSelector` assigns reviewers using COI checks, trust-weighted priority, and mandatory model-family diversity. Position bias is mitigated by randomizing review order. Self-enhancement bias (same-family reviewers systematically favor each other’s outputs [12], [13]) is structurally prevented by requiring heterogeneous reviewer pools—the selector enforces that at least two distinct model families participate in any Tier 2+ review.

Adaptive Verification Cascade. A `VerificationCascade` implements MAV-style [14] adaptive verification depth: high-trust proposers receive lighter review (early exit), while low-trust proposers face the full multi-agent panel. Critical-tier proposals always receive full cascade regardless of trust.

Domain Trust Aggregation. Trust scores are not single-dimensional. An agent’s trust in `code_review` may differ from its trust in `security` or `architecture`. The protocol computes weighted multi-domain trust using a domain-weight map (e.g., $\text{code_review} = 0.7 \times \text{code} + 0.2 \times \text{security} + 0.1 \times \text{arch}$), enabling nuanced reviewer selection based on the proposal’s task type.

Critically, none of these extensions required changes to the six IDP tuple types, their fields, or the cross-link identifier schema. The reviewer selection, verification cascade, and domain trust aggregation are all recorded as `SYSTEM` tuples on the existing governance bus, using the same append-only semantics and lifecycle invariants.

The protocol was validated in a preliminary live session ($n = 15$ research packets); full statistical caveats are reported in Section IV-B. The full protocol (1,545 lines, 132 tests) operates over the coordination bus without any modification to the IDP schema.

V. EVALUATION

A. Audit Completeness

We define audit completeness as the fraction of agent actions that produce a corresponding governance tuple chain (`CONTRACT`, `DCT`, `EVIDENCE`, `ATTEST`). In our briefing system, every adapter invocation flows through the IDP lifecycle. The governance bus captured all 32 expected entries in the evaluated run. Rather than treating this as an empirical completeness claim, we note that audit completeness is a *structural* property of the architecture: every state transition generates a signed entry through the governed dispatch path, making gaps detectable rather than merely unlikely.

Over a 25-day evaluation period, the governance bus accumulated 150,192 signed entries across 15 active days, with

daily volumes ranging from 33 (low-activity maintenance day) to 26,014 (peak sprint day). The inter-agent coordination bus recorded 10,955 messages over the same period. This sustained operational volume provides confidence that the tuple profile functions reliably under realistic workloads, not just in single-run demonstrations.

We note that a signature-enforcement defect was discovered and fixed during this evaluation period: the convenience helper for governance bus writes had silently omitted signatures, causing entries to be rejected by the bus’s default-deny policy. The defect was caught precisely because the enforcement mechanism works—unsigned entries were rejected, not silently accepted. This is evidence that the fail-closed design provides self-detecting integrity.

B. Enforcement Overhead

The IDP’s runtime enforcement adds overhead at three points: DCT creation and signing (HMAC-SHA256), governance bus writes (signature verification + append), and cross-link validation. In our deployment, the total IDP overhead per briefing run is under 200 milliseconds, dominated by HMAC computation. This is negligible relative to the adapter invocations themselves, which involve network calls to external services (source control, calendar, issue tracker) with latencies measured in seconds. The governance machinery imposes no observable impact on end-to-end briefing generation time.

C. Delegation-Tree Reconstructability

Given any single governance tuple, the complete delegation tree can be reconstructed by following cross-link identifiers. We verified this by querying individual `EVIDENCE` tuples and tracing back through `contract_id`, `capability_token_id`, and `parent_dctx_id` to the root delegation. In all cases the chain was complete and unbroken; append-only semantics ensure no intermediate tuple can be removed without detection.

D. Peer Review Protocol Effectiveness

The peer review protocol provides a second axis of evaluation: can the IDP’s composability yield measurable quality improvements? In a preliminary validation session, 15 research packets were submitted through the bus-integrated peer review pipeline. Cross-model-family review detected fabricated citations in 2 packets (4 of 8 claimed sources were invalid in the worst case). After one rejection-and-revision cycle, no further fabricated citations were detected in packets 5–15.

Statistical caveats. With $n = 15$, these results are insufficient to support causal claims about the protocol’s effectiveness. The observed drop in fabrication rate (from 2/4 in early packets to 0/11 in later packets) yields a Fisher exact test p -value of 0.047, which is marginally significant but should be treated with caution given the small sample, the absence of a control condition (no protocol), and the non-independence of packets (same agents, accumulating trust scores). The result demonstrates *feasibility*—that the protocol can be operationalized within the IDP without schema changes—not *efficacy* at

scale. A powered evaluation would require a larger sample, a control arm, and cross-session replication.

Three protocol properties plausibly contributed to the observed outcome:

- 1) **Heterogeneous reviewer pools** ensured that fabricated sources were reviewed by an agent from a different model family than the author. Same-family review is known to exhibit self-enhancement bias [12], [13], which would likely have reduced detection sensitivity.
- 2) **The Beta trust model** updated reviewer and proposer trust scores after each review cycle. The fabricating agent’s trust score declined, triggering deeper verification on subsequent submissions.
- 3) **Contract-scoped acceptance criteria** (specified in the CONTRACT tuple) gave reviewers explicit standards rather than open-ended quality assessment, reducing verbosity bias.

The protocol implementation spans 1,545 lines across three modules (review engine, bus bridge, and governance verifier) with 132 dedicated tests. The bus bridge module processes PROPOSAL, REVIEW, and VERIFY messages from the coordination bus without requiring changes to the bus format or the IDP governance bus.

E. Adversarial Trust Simulation

To quantify the gaming resistance of our trust model hardening (Section VI-E), we simulate a strategic gaming attack under a specified threat model.

Threat model. The adversary is an agent that has been delegated STANDARD-tier capabilities and seeks to escalate to CRITICAL-tier operations. The adversary knows the trust scoring formula (Kerckhoffs’s principle: the algorithm is public) and can observe its own trust score after each round. The adversary’s goal is to reach a trust threshold that triggers reduced verification depth on a CRITICAL task, then submit a malicious CRITICAL proposal. The adversary’s budget is measured in rounds: each round requires the adversary to produce a proposal, wait for verification, and observe the outcome. The adversary’s information is perfect regarding its own trust score and the task-tier mapping, but it does not know the verifier’s internal randomization (reviewer pool selection is non-deterministic). We assume the adversary cannot forge DCTs (cryptographic infeasibility) and cannot bypass the trust-risk verification matrix (structurally enforced).

Attack strategy. The adversary follows a *build-then-exploit* strategy: (1) submit b consecutive STANDARD proposals, all succeeding; (2) submit one CRITICAL proposal, which fails (the exploit attempt). We measure three outcomes: trust score at exploitation, verification depth reached, and recovery rounds to pre-exploit trust level.

Parameter choices and sensitivity. The hardened model uses an asymmetric update: STANDARD success adds $0.5 \times \alpha_{\text{base}}$ to the Beta shape parameter, while CRITICAL failure adds $3 \times \beta_{\text{base}}$. The $3 \times$ penalty multiplier and $0.5 \times$ reward multiplier are derived from Josang’s risk-weighted subjective

logic [18], calibrated so that a single CRITICAL failure outweighs approximately six STANDARD successes (at $\alpha_{\text{base}} = \beta_{\text{base}} = 1$). Table V reports sensitivity to these multipliers. A $2 \times$ penalty (weaker than our choice) still exceeds 50 recovery rounds at $b = 50$, while a $5 \times$ penalty exceeds 100 rounds at $b = 30$. The $0.5 \times$ reward multiplier is the dominant factor: at $1.0 \times$ (uniform reward), recovery drops to 14 rounds at $b = 50$, making gaming feasible. Our $0.5 \times$ choice sits in a regime where recovery rounds scale superlinearly with buildup investment.

TABLE V
SENSITIVITY: RECOVERY ROUNDS AT $b = 50$ FOR VARYING PENALTY/REWARD MULTIPLIERS

Reward mult.	Penalty $2 \times$	Penalty $3 \times$	Penalty $5 \times$
$1.0 \times$	28	42	68
$0.5 \times$	51	>100	>100
$0.25 \times$	78	>100	>100

Baseline. We compare a *naive* trust model (uniform alpha/beta updates, no risk differentiation) against our *hardened* model (asymmetric risk-weighted penalties + trust-risk verification matrix).

TABLE VI
GAMING RESISTANCE: NAIVE VS. HARDENED TRUST MODEL

Build	Tier	Naive			Hardened		
		Score	Depth	Rec.	Score	Depth	Rec.
10	ELEV	.917	STD	11	.857	STD	24
10	CRIT	.917	ELEV	11	.857	ELEV	36
20	ELEV	.955	STD	21	.917	STD	44
20	CRIT	.955	ELEV	21	.917	ELEV	66
30	ELEV	.969	STD	31	.941	STD	64
30	CRIT	.969	ELEV	31	.941	ELEV	96
50	ELEV	.981	STD	51	.963	STD	>100
50	CRIT	.981	ELEV	51	.963	ELEV	>100

Three properties are visible in the data:

- 1) **Slower trust accumulation.** After 30 low-risk successes, the hardened model reaches a trust score of 0.941 vs. the naive model’s 0.969. The asymmetric reward scaling ($0.5 \times$ for STANDARD success) means gaming requires approximately twice the buildup investment.
- 2) **Steeper failure penalty.** A single CRITICAL failure drops the hardened model from 0.941 to 0.800 (a 15% drop), compared to the naive model’s drop from 0.969 to 0.939 (a 3% drop). The $3 \times$ beta penalty for CRITICAL failures makes each exploit attempt costly.
- 3) **Expensive recovery.** Recovery to pre-exploit trust levels takes $2\text{--}3 \times$ as many rounds in the hardened model (66 vs. 21 rounds after 20-round buildup + CRITICAL exploit). At 50 buildup rounds, the hardened model exceeds 100 recovery rounds, making the attack economically infeasible.

The trust-risk verification matrix provides a complementary structural defense: regardless of trust score, CRITICAL tasks

always require at minimum ELEVATED verification. This prevents a gaming attack from reaching MINIMAL verification depth even if the trust score manipulation succeeds.

The simulation models discrete rounds without temporal decay; the 30-day half-life decay described in the protocol specification would further accelerate trust erosion for inactive agents but is not reflected in Table VI. These results are reproducible via the self-contained simulation harness (`trust_simulation.py`, Python 3.11+ stdlib only) provided in the supplementary material, which generates the full comparison table, trust trajectory data, and verifies all quantitative claims.

VI. DISCUSSION AND LIMITATIONS

A. Tool-Boundary Enforcement

The current IDP implementation enforces capability constraints at the scheduler and adapter wrapper level, not at the operating-system process boundary. An agent that bypasses the scheduler’s adapter dispatch could invoke tools directly without DCT validation. A planned tool gateway module will enforce capabilities at the subprocess execution boundary, closing this gap. Until then, enforcement depends on agents routing tool calls through the governed dispatch path. Additionally, the IDP enforcement layer can be disabled via an environment variable for emergency bypass during incident response. Deployment hardening should remove this escape hatch or restrict it to authenticated operator action with audit logging.

B. Coordination Bus Authentication

The IDP governance bus (JSONL) enforces HMAC-SHA256 signatures on every entry. However, the coordination bus (TSV), used for real-time inter-agent messaging, does not yet have per-message cryptographic authentication. An adversarial agent could post spoofed coordination messages. The governance bus is unaffected by this gap because it operates as an independent channel with its own signature enforcement. Closing the coordination bus authentication gap is a planned hardening step.

C. Single-Trust-Domain Limitation

The current HMAC-SHA256 signing assumes a single shared secret within one trust domain. When no signing secret is configured, the implementation falls back to an ephemeral random key, meaning signatures are valid only within a single process lifetime and unverifiable after restart. Production deployments must provide a stable secret via environment variable. Cross-organizational deployment would require asymmetric signing (e.g., Ed25519) to support non-repudiation without shared secrets. The tuple profile itself is agnostic to the signing mechanism; upgrading to asymmetric signatures requires changing the DCT signing implementation, not the governance schema.

D. Scale

Our evaluation is based on a system with four agents, six active adapters (of seven implemented), and delegation trees of depth 2–3. Over a 25-day evaluation period, the governance bus accumulated over 150,000 signed entries without degradation, demonstrating that the append-only JSONL format scales to production workloads at this agent count. We have not yet evaluated the profile at scales of hundreds of agents or deep delegation chains. The append-only governance bus grows linearly with the number of delegations; query performance over large buses may require indexing that the current flat-file implementation does not provide.

E. Trust Model Hardening

The Beta reputation trust model has two known attack surfaces. First, the cold-start problem: new agents begin with a uniform prior ($\alpha = 1, \beta = 1$, score = 0.5, uncertainty = 1.0), placing them at maximum verification depth until sufficient history accumulates. We address this with community base rate bootstrapping [18]: newcomers inherit the population mean trust as their prior rather than the uniform distribution, reducing initial uncertainty without weakening assurance. A graduated sandbox gate further controls graduation: agents must complete a minimum number of reviews AND reduce their uncertainty below a threshold before receiving reduced verification.

Second, strategic trust gaming: an agent could build trust on low-risk proposals and exploit it on high-risk ones. We address this with two structural defenses. An asymmetric risk-weighted penalty model (following Josang’s advanced features [18]) makes gaming expensive: a failure on a CRITICAL task incurs $3\times$ the beta penalty of a failure on a STANDARD task, while successes on STANDARD tasks contribute only $0.5\times$ the alpha reward. Additionally, a trust-risk verification matrix ensures that task risk always bounds verification depth: even a fully trusted agent (score > 0.8) receives at minimum STANDARD verification on ELEVATED tasks and ELEVATED verification on CRITICAL tasks. This structural defense is motivated by recent findings that deception ability scales faster than detection ability in LLM multi-agent settings [19], making agent-based detection of gaming unreliable. Section 5.5 provides quantitative simulation evidence for the effectiveness of these defenses. We note that the current simulation models a single attack strategy (linear trust buildup followed by exploitation). Extending to diverse adversarial strategies—including gradual degradation and coordinated multi-agent attacks—remains future work.

F. Formal Verification

The lifecycle invariants (evidence-before-verify, monotonic attenuation, bounded depth) are enforced by runtime checks in Python, supplemented by a trace-based governance log verifier that audits recorded JSONL traces against all three invariants post-hoc.

We additionally provide a TLA+ specification of the DCTX state machine (included in supplementary material) that models all nine state transitions and specifies four safety invariants: `EvidenceBeforeVerify` (I1), `DepthMonotonicity` (I2, modeling depth ordering as an abstraction of the full capability-subset property enforced in implementation), `BoundedDepth` (I3), and `BoundedReplans`. `TerminalIrreversible` is verified as an action property (no transition exits a terminal state) rather than a state invariant. A liveness property (`EventualTermination`) asserts that every non-failed task eventually reaches a terminal state under fair scheduling. The specification follows the approach of Gouglidis et al. [21] for usage control models and draws on the Vouchsafe capability graph model [20], which formalizes monotonic scope attenuation as a core property structurally identical to our I2 invariant.

G. Implementation Status

Three aspects of the profile that were initially implemented by convention are now structurally enforced. First, the amendment identifier mechanism (Section 3.5) is enforced via an explicit `amendment_of` field on governance entries; the bus rejects amendments that reference nonexistent entry identifiers. Second, the cross-link identifiers `contract_id`, `capability_token_id`, and `verification_id` (Section 3.2) are promoted to top-level fields on every governance entry, enabling direct traversal without inspecting `tuple_data`. Third, the normative tuple schema (Section III-B) is enforced by a stdlib-only JSON Schema validator (Draft 2020-12 subset): malformed tuples are rejected before append.

The I1 invariant (evidence-before-verify) is enforced at the bus layer: ATTEST entries without a `verification_id` referencing an existing EVIDENCE tuple are rejected with error code `IDP_E_EVIDENCE_REQUIRED`. The I2 invariant (monotonic capability attenuation) is enforced at the delegation context layer: child delegations with capabilities not present in the parent are rejected with `IDP_E_CAPABILITY_ESCALATION`. The I3 invariant (bounded depth) is enforced at the state machine layer: transitions that would exceed `MAX_DEPTH` are rejected with `IDP_E_DEPTH_EXCEEDED`. These enforcement mechanisms are validated by 167 automated tests across the governance bus, delegation context, delegation token, and MCP gateway modules (distinct from the 132 peer review protocol tests reported in Section IV-B).

Remaining gaps. Three enforcement gaps remain. (1) Tool-boundary enforcement operates at the scheduler/adaptor wrapper level, not the OS process boundary; an agent that bypasses the scheduler could invoke tools directly without DCT validation. A planned tool gateway module will close this gap. (2) The coordination bus (TSV, used for real-time inter-agent messaging) does not yet have per-message cryptographic authentication; the governance bus (JSONL) is unaffected because it uses independent HMAC signature enforcement.

(3) Cross-organizational deployment would require asymmetric signing (e.g., Ed25519); the current implementation uses HMAC-SHA256 with a single shared secret, which is appropriate for single-trust-domain deployments but not for inter-organizational delegation. The tuple profile is signing-agnostic; upgrading requires changing the signing implementation, not the schema.

VII. CONCLUSION

We presented the Intelligent Delegation Profile, a typed six-tuple governance model for multi-agent runtime control. The profile’s contribution is not tuples, tokens, or attestations individually—each of these has clear prior art in Linda, Macaroons/Biscuit, and IETF RATS. The contribution is the composition: a governance profile with cross-linked identifiers, lifecycle invariants, and runtime enforcement that spans the full delegation lifecycle from contract to verification.

We provided operational evidence from a production multi-agent system, demonstrating structural audit completeness (gaps are detectable, not merely unlikely), negligible enforcement overhead, and full delegation-tree reconstructability. We demonstrated the profile’s composability through a peer review protocol (v0.7, 132 tests) that introduces multi-agent verification, Beta reputation trust scoring, collusion defense, bias-aware reviewer selection, adaptive verification cascades, domain-weighted trust aggregation, asymmetric risk-weighted penalties, trust-risk verification matrices, community base rate bootstrapping, and graduated sandbox policies—all without modifying the IDP’s six tuple types or cross-link schema. Adversarial simulation confirms the hardened trust model’s gaming resistance: recovery from a critical exploit requires 2–3× more rounds than the naive model, with recovery exceeding 100 rounds at higher buildup investments (Table VI).

The IDP addresses a concrete gap in current multi-agent infrastructure: the absence of composable governance primitives that operate at the delegation boundary. As multi-agent systems move toward production deployment, we believe typed governance profiles—whether the IDP or alternatives with similar properties—will be necessary infrastructure for trustworthy autonomous operation.

REFERENCES

- [1] W3C, “PROV-Overview: An Overview of the PROV Family of Documents,” W3C Working Group Note, 2013.
- [2] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, “Remote Attestation procedureS (RATS) Architecture,” RFC 9334, IETF, 2023.
- [3] A. Birgisson, J. G. Politz, U. Erlingsson, A. Taly, M. Vrabie, and M. Lentzner, “Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud,” in *Proc. NDSS*, 2014.
- [4] G. Couprie, “Biscuit: Authorization Tokens for Decentralized Systems,” Technical Report, 2021. <https://www.biscuitsec.org/>.
- [5] N. Tomasev, M. Franklin, and S. Osindero, “Intelligent AI Delegation,” Google DeepMind, arXiv:2602.11865, 2026.
- [6] D. Gelernter, “Generative Communication in Linda,” *ACM Trans. Program. Lang. Syst.*, vol. 7, no. 1, pp. 80–112, 1985.
- [7] A. Omicini and F. Zambonelli, “Coordination for Internet Application Development,” *Auton. Agents Multi-Agent Syst.*, vol. 2, no. 3, pp. 251–269, 1999.
- [8] A. Omicini and E. Denti, “From Tuple Spaces to Tuple Centres,” *Sci. Comput. Program.*, vol. 41, no. 3, pp. 277–294, 2001.

- [9] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Capos, "in-toto: Providing farm-to-table guarantees for bits and bytes," in *Proc. USENIX Security*, 2019.
- [10] A. Josang and R. Ismail, "The Beta Reputation System," in *Proc. 15th Bled eCommerce Conf.*, 2002.
- [11] A. Bashir, T. A. Han, and Z. U. Shamszaman, "Many-to-One Adversarial Consensus: Exposing Multi-Agent Collusion Risks in AI-Based Healthcare," arXiv:2512.03097, 2025.
- [12] S. Gu et al., "A Survey on LLM-as-a-Judge," arXiv:2411.15594, 2024.
- [13] A. Panickssery, S. R. Bowman, and S. Feng, "LLM Evaluators Recognize and Favor Their Own Generations," in *Proc. NeurIPS*, 2024, arXiv:2404.13076.
- [14] O. Lifshitz et al., "Multi-Agent Verification: Scaling Test-Time Compute with Multiple Verifiers," arXiv:2502.20379, 2025.
- [15] Google, "Agent-to-Agent Protocol Specification," a2a-protocol.org, 2025. Transferred to Linux Foundation AAF, 2026.
- [16] Anthropic, "Model Context Protocol Specification, Version 2025-11-25," modelcontextprotocol.io. Contributed to Linux Foundation AAF, 2026.
- [17] S. Nagabhushanaradhya, "OpenID Connect for Agents (OIDC-A) 1.0," arXiv:2509.25974, 2025.
- [18] A. Josang and W. Quattrociocchi, "Advanced Features in Bayesian Reputation Systems," in *Proc. TrustBus*, 2009.
- [19] P. M. P. Curvo, "The Traitors: Deception and Trust in Multi-Agent Language Model Simulations," NeurIPS MTI-LLM Workshop, arXiv:2505.12923, 2025.
- [20] J. Kuri, "Vouchsafe: A Zero-Infrastructure Capability Graph Model for Offline Identity and Trust," arXiv:2601.02254, 2026.
- [21] A. Gouglidis et al., "Formal Verification of Usage Control Models with TLA+," *Softw. Tools Technol. Transfer*, 2020.
- [22] Okta, "Cross App Access: Extending MCP with Enterprise-Managed Authorization," okta.com, 2025.
- [23] Agent Contracts, "Extending the Contract Metaphor from Task Allocation to Resource-Bounded Execution," arXiv:2601.08815, 2026.
- [24] The Auton Agentic AI Framework, "A Declarative Architecture for Specification, Governance, and Runtime Execution of Autonomous Agent Systems," arXiv:2602.23720, 2026.
- [25] Y. Ge, "Governance Architecture for Autonomous Agent Systems: Threats, Framework, and Engineering Practice," arXiv:2603.07191, 2026.
- [26] SAGA, "Security Architecture for Governing AI Agentic Systems," arXiv:2504.21034, 2025. Accepted at NDSS 2026.
- [27] Agentic JWT, "Dual-Faceted Intent Tokens for Agent Authorization," arXiv:2509.13597, 2025.
- [28] C. L. Wang, T. Singhal, A. Kelkar, and J. Tuo, "MI9: An Integrated Runtime Governance Framework for Agentic AI," arXiv:2508.03858, 2025.
- [29] S. Gaurav, J. Heikkonen, and J. Chaudhary, "Governance-as-a-Service: A Multi-Agent Framework for AI System Compliance and Policy Enforcement," arXiv:2508.18765, 2025.
- [30] DelegateOS, "Multi-Agent Delegation and Orchestration Protocol," <https://github.com/newtro/delegateos>, 2025. Accessed: Mar. 2026.
- [31] J. B. Dennis and E. C. Van Horn, "Programming Semantics for Multi-programmed Computations," *Commun. ACM*, vol. 9, no. 3, pp. 143–155, 1966. doi:10.1145/365230.365252.
- [32] N. Hardy, "The Confused Deputy (or why capabilities might have been invented)," *ACM SIGOPS Oper. Syst. Rev.*, vol. 22, no. 4, pp. 36–38, 1988. doi:10.1145/54289.871709.
- [33] M. S. Miller, K.-P. Yee, and J. S. Shapiro, "Capability Myths Demolished," Tech. Rep., DARPA OASIS, 2003.
- [34] M. S. Miller, "Robust Composition: Toward a Unified Approach to Access Control and Concurrency Control," Ph.D. dissertation, Johns Hopkins University, 2006.
- [35] T. South, S. Marro, T. Hardjono, R. Mahari, C. D. Whitney, D. Greenwood, A. Chan, and A. Pentland, "Authenticated Delegation and Authorized AI Agents," arXiv:2501.09674, 2025.
- [36] S. Jiang, D. Ju, A. Cohen, S. Mitts, A. Foss, J. T. Kao, X. Li, and Y. Tian, "Towards Full Delegation: Designing Ideal Agentic Behaviors for Travel Planning," arXiv:2411.13904, 2024.
- [37] T. Shi, J. He, Z. Wang, H. Li, L. Wu, W. Guo, and D. Song, "Progent: Programmable Privilege Control for LLM Agents," arXiv:2504.11703, 2025.
- [38] G. Shi, H. Du, Z. Wang, X. Liang, W. Liu, S. Bian, and Z. Guan, "SoK: Trust-Authorization Mismatch in LLM Agent Interactions," arXiv:2512.06914, 2025.
- [39] S. Prakash, "AIP: Agent Identity Protocol for Verifiable Delegation Across MCP and A2A," arXiv:2603.24775, 2026.
- [40] Y. Louck, A. Stulman, and A. Dvir, "Improving Google A2A Protocol: Protecting Sensitive Data and Mitigating Unintended Harms in Multi-Agent Systems," arXiv:2505.12490, 2025.
- [41] S. Wang, R. Raskar, M. Lambe, P. Chari, R. Singhal, S. Gupta, R. Ranjan, and K. Huang, "Using the NANDA Index Architecture in Practice: An Enterprise Perspective," arXiv:2508.03101, 2025.
- [42] R. Khan, D. Joyce, and M. Habiba, "AGENTS SAFE: A Unified Framework for Ethical Assurance and Governance in Agentic AI," arXiv:2512.03180, 2025.
- [43] Fission, "UCAN: User Controlled Authorization Networks," <https://ucan.xyz>, 2023.
- [44] CNCF, "SPIFFE: Secure Production Identity Framework for Everyone," <https://spiffe.io>, 2024.