

Read-First and Role-Gated Context Federation for Multi-Agent LLM Systems: A Controlled Multi-Model Comparison with Point-to-Point Messaging

Ryan Valdivia-Segovia, Adrian Mamani-Quispe, Karim Guevara, Norka Bedregal-Alpaca
Universidad Nacional de San Agustín de Arequipa, Arequipa, Peru

Abstract—Multi-agent systems based on large language models (LLMs) commonly rely on point-to-point message-passing, which increases integration complexity and leaves fine-grained access control and traceability to additional mechanisms. This study evaluates the Graph Context Protocol (GCP), a read-first context-federation approach with role- and capability-based access control and native per-read provenance. GCP is compared with two Agent2Agent (A2A) arms: the base protocol as shipped, and an instrumented variant whose policy enforcement point applies the same role and delegation policy and writes an audit log. Agent logic, models, tasks, and evaluation criteria are held constant; only the interoperability layer changes. Five behavioral scenarios, one scoring semantic leakage of a useful but forbidden node, were run on six models (three local, three hosted; 2,475 runs, none failed). A structural analysis shows that a hub-style GCP needs $\mathcal{O}(n)$ integration edges against $\mathcal{O}(n^2)$ for a full A2A mesh, but A2A through an orchestrator is also $\mathcal{O}(n)$, so this reflects topology, not protocol. GCP denied all 443 unauthorized attempts and instrumented A2A all 408, whereas base A2A executed all 392; the instrumented arm matched GCP on containment, provenance, and leakage in every model, although a hermetic test shows that a caller omitting its operation declaration bypasses its delegation rule. The governance gap thus reflects the presence of an enforcement point and audit sink, which GCP provides natively and A2A requires each deployment to add. Token cost is comparable between GCP and instrumented A2A; task success is indistinguishable for four of six models and lower under GCP for the smallest hosted model, where the cause is the wording of the tool description. A software-org gap for llama3.1:8b did not reproduce on rerun.

Keywords—Large language models; access control; provenance; agent containment; software coupling

I. INTRODUCTION

Direct message-passing is the predominant coordination pattern in multi-agent systems based on large language models (LLMs). Frameworks such as AutoGen [1], ChatDev [2], and MetaGPT [3] organize applications around agents that communicate through messages. The Agent2Agent (A2A) protocol [4], in turn, standardizes inter-agent communication: a client agent advertises and delegates tasks to named remote agents over JSON-RPC. This paradigm descends from the classical agent communication languages KQML [5] and FIPA-ACL [6], which already assumed pairwise connections and shared, pre-negotiated protocols between every interacting pair.

Message-passing has a structural cost the software-engineering literature has understood for fifty years. Connecting

n components pairwise requires on the order of n^2 links; a shared intermediary that each component connects to once reduces this to n . This is the “integration spaghetti” the Enterprise Integration Patterns [7] catalog addresses with brokers and hubs, and the phenomenon coupling theory [8–10] formalizes: modifiability is governed by the number and strength of inter-component dependencies.

An alternative infrastructure for agent coordination is studied here: the Graph Context Protocol (GCP). This protocol prioritizes reading: an agent obtains what it needs by reading context from independently owned knowledge nodes rather than receiving pushed messages. It is *role-gated*: every read passes through a policy gate that authorizes the request by role and required capabilities, defaulting to deny [11, 12]. GCP also provides auditability: every read or delegation decision—grant or deny—produces a structured provenance record.

1) *Thesis (measured form)*: Read-first, role-gated context federation moves authorization and audit from each pair of agents to the node and the protocol. The study measures 1) the integration topology this implies, 2) the governance outcomes (containment, provenance, and leakage) against both base A2A and an A2A deployment instrumented with an equivalent policy enforcement point, and 3) the cost and task-success consequences across six models. Structural results are deterministic; governance and behavioral results come from 30 seeds per arm (15 for the largest local model) with Wilson intervals and exact tests.

2) *Scope of the comparison*: Message-passing is not claimed to be inherently unauditible: logs, middleware, proxies, or gateways can add an access trail. Two A2A arms are therefore compared with GCP. The *base* arm is the protocol as shipped, without authorization or logging, and shows what a deployment lacks until it adds such controls. The *instrumented* arm (A2A+PEP) runs the same A2A servers behind a policy enforcement point that authenticates a bearer token, applies the same role policy and the same delegation rule as GCP, and records an audit entry per decision. The instrumented arm is the fair comparator for the governance claims; the base arm is reported because it is what the protocol provides out of the box.

3) *Contributions*:

- Architecture and a reproducible specification. GCP (read-first, role-gated, audited, with a stricter-capability

delegation gate) is described precisely enough to reimplement: query format, node and policy structure, role credential, provenance record, and the authorization algorithm (Section III).

- A fairness-controlled benchmark method. One scenario definition and one injected model build both arms; only the interop layer changes (Section V).
- A multi-model, three-arm empirical head-to-head over five behavioral scenarios—including a *forced-access* probe that targets the confidential node directly and a *useful-forbidden* scenario scored with semantic leakage patterns—and six models (three open-weight, three hosted), 2,475 logged runs with traces, separating deterministic from model-dependent results (Section VI).
- Metric definitions for structural coupling, discovery/tool-prompt overhead, a four-way leakage/agency taxonomy, provenance completeness, and delegation/forced-access containment (Section IV).
- An instrumented A2A baseline and a threat model. A third arm (A2A+PEP) applies the same role and delegation policy and writes an audit log, so that the comparison separates the protocol from the presence of an enforcement point; credential binding and six adversaries are specified, with the evidence available for each (Sections III-F and VI-D).
- A deterministic integration-effort proxy that counts the declarations and edits needed to onboard one more participant under each topology (Section VI-C).

II. BACKGROUND

A. Message-Passing as the Multi-Agent Baseline

Agent communication languages established the message-passing paradigm: KQML [5] wrapped payloads in typed performatives over pairwise connections, and FIPA-ACL [6] standardized a message in which sender and receiver must share content language and ontology before any exchange succeeds. Modern LLM frameworks reinvented this in natural language. AutoGen [1] programs applications as conversations; ChatDev [2] arranges role-pairs in a fixed “chat chain”; surveys [13] taxonomize communication into peer-to-peer, centralized, layered, and shared-pool variants. The Model Context Protocol [14] standardizes agent-to-tool context access (an early read-first form), while A2A [4] standardizes agent-to-agent task delegation—the baseline benchmarked against here. [15] report 28–73% excess tokens in inter-agent communication graphs and recover most by pruning topology. [16] replace both centralized orchestration and peer-to-peer routing with a shared verified context at roughly half the cost. This work adds role-gated authorization, audited per-read provenance, an agency-containment gate, and a multi-model fairness-controlled comparison.

B. Coupling and the n^2 Problem

That fewer inter-component links yield more maintainable systems is among the oldest results in software engineering. [8] showed information-hiding decomposition produces far lower coupling than flowchart-style partitioning; [9] made

coupling and cohesion measurable; [10] gave a graph-theoretic framework in which coupling is monotone in dependency-edge count, so an $\mathcal{O}(n)$ -edge topology is provably less coupled than an $\mathcal{O}(n^2)$ one. [7] translated this to integration: a point-to-point mesh of n systems needs $n(n-1)/2$ channels, which a hub reduces to n . Microservice research [17] carries the same argument into cloud-native systems, and [18] show dense agent topologies amplify error propagation while sparser ones suppress it. A2A creates agent $\leftrightarrow$ agent edges; GCP creates agent $\rightarrow$ store edges.

C. Least Privilege, Access-Controlled Context, and Gated Retrieval

GCP’s read gate operationalizes least privilege [11]. Role-based access control [12] assigns permissions to roles; attribute-based access control [19] generalizes to arbitrary attributes; capability-based protection [20] ties authority to explicitly passed tokens—the model GCP’s role credentials follow. Data-minimization law [21] makes “read only what your role requires” a legal constraint. Read-first tool access (MCP) and access-controlled retrieval supply the read side, and RBAC/ABAC the gate side, but neither couples the two with per-read provenance; the point-to-point baseline has no analogous gate, so leakage and delegation control rest on sender-side discretion.

D. Leakage, Provenance, and Governance in Agentic Systems

Confidentiality in agentic systems is a system property, not only a model property [22]; inter-agent message channels are the dominant leakage vector, largely invisible to output-only auditing [23]; and restricting what an agent can *see* is more robust than filtering what it can *say* [24]. The OWASP Top 10 for LLM applications [25] names Sensitive Information Disclosure and Excessive Agency as primary risks—the latter is exactly what the delegation scenario probes. Deterministic privilege control at the tool boundary [26], authenticated delegation of authority [27], and audit trails [28] argue for tamper-evident records of system actions.

1) *Novelty*: Prior work supplies the pieces—read-first access (MCP), role/attribute gating (RBAC/ABAC, access-controlled RAG), audited delegation, and audit logs—but not their conjunction, and not a fairness-controlled, multi-model head-to-head against a real message-passing baseline. The novelty claimed here is precisely the *integration*: federated read + role control + per-read provenance, evaluated against A2A. Read-first alone, RBAC alone, or auditing alone would not be a contribution.

III. THE GRAPH CONTEXT PROTOCOL: SPECIFICATION

GCP models an ecosystem as independently owned *knowledge nodes* and *agent nodes*. Coordination is read-first: an agent issues a *context query* to a peer node, which returns context the caller is authorized to read. The substrate is specified precisely enough to reimplement.

A. Data Structures

A knowledge node carries an id, a kind, an adapter that produces content, and an access policy. A policy declares gating fields; a role credential names the caller’s role and flat capabilities; a provenance record logs each decision.

```

KnowledgeNode {
  id, kind,           // e.g. "rag", "log"
  adapter,           // yields raw content
  policy: AccessPolicy }
AccessPolicy {
  readableByRoles: [roleId],
  requiredCapabilities: [capId],
  fallbackAllowed: bool,
  denialMode: "empty" | "error" }
Principal (role credential) {
  id, role: { id, capabilities:[capId],
             parent? },
  capabilities: [capId] }
Provenance {
  principalId, targetNodeId, timestamp,
  decision: "grant" | "deny",
  matchedPolicy }

```

B. Query and Authorization Flow

A context query names a target node and a requested capability (read, or the stricter delegate). The server evaluates the admission predicate *before* any adapter runs, so denied content never enters the agent's context window (complete mediation [11]). The effective capability set of a principal is the union of its flat capabilities and those inherited through the role chain (own-first precedence). The predicate is

$$\begin{aligned}
 \text{authorized}(P, K) \equiv & \underbrace{(P.\text{roles}=[] \vee K.\text{role} \in P.\text{roles})}_{\text{roleOk}} \\
 & \wedge \underbrace{(P.\text{caps}=[] \vee P.\text{caps} \subseteq \text{eff}(K))}_{\text{capsOk}}
 \end{aligned}$$

An absent or schema-invalid policy is treated as a denial (default-deny). The end-to-end path is:

Request → Policy check → Decision → Context returned / denied
→ Provenance record

C. Read/Authorize Algorithm (Pseudocode)

```

handleQuery(query, principal):
  node = resolve(query.targetNodeId)
  if node == null: return deny("no-node")
  P = node.policy
  if P invalid or absent: # default-deny
    record(principal, node, "deny", "none")
    return deny("no-policy")
  eff = flat(principal) UNION
        rolechain(principal.role)
  roleOk = P.roles==[] or
           principal.role.id in P.roles
  capsOk = P.caps==[] or P.caps subset eff
  needCap = query.capability # read|delegate
  capOk2 = needCap in eff
  if roleOk and capsOk and capOk2:
    record(principal, node, "grant", P.id)
    return node.adapter.read() # content
  else:
    record(principal, node, "deny", P.id)
    return denialResult(P.denialMode) # denied

```

Delegation reuses this path with needCap = delegate, a strictly stronger capability than read: an agent authorized to read is not thereby authorized to command.

D. Denial Handling

On denial the handler returns only a denial reason (or an empty result, per `denialMode`); the adapter is never invoked, so node content cannot appear in the response. Every denial is itself recorded, yielding a who-tried-what ledger, not only a who-read-what one.

E. Relation to A2A and MCP

Table I contrasts the three. A2A is agent-to-agent and push-based, with coarse card-declared allow/deny and no native per-read trail; MCP is read-first but agent-to-tool and single-owner, without role gating across owners; GCP combines cross-owner read-first access, role gating, deny-by-default, and native per-read provenance. A “not nat.” in Table I does not mean the protocol makes the property impossible; every cell is achievable in principle by extending the base protocol. “not nat.” = not a native primitive of the base protocol (would require added logic on top of it); “ext.” = achievable only with additional external instrumentation; “n/a” = the concept does not apply to this protocol's design. The cells are derived from the cited specifications of A2A and MCP, not from experiments against MCP-based gated retrieval or policy layers such as Progent [26], which are not evaluated here.

TABLE I. PROPERTY COMPARISON

property	A2A	MCP	GCP
agent↔agent	yes	not nat.	yes
read-first	not nat.	yes	yes
cross-owner boundary	yes	not nat.	yes
role/capability gating	coarse	not nat.	yes
per-read provenance (native)	ext.	ext.	yes
deny-by-default	not nat.	not nat.	yes
stricter-capability delegation	not nat.	n/a	yes

F. Threat Model and Credential Binding

1) *Credential binding*: A query carries a bearer credential, not a role. The server passes the credential to its `AuthProvider`, which returns the `Principal` (identifier, role, flat capabilities); the gate uses only that server-resolved principal, never a role asserted in the request. The `requester.principalId` field of a query is used solely to label audit records of failed authentications. The evaluation uses a static token table (one token mapped to one principal per run), which stands for out-of-band credential issuance; token lifetime, rotation, and sender-constraining are not modeled.

2) *Adversaries*: Six adversaries are distinguished (Table II), with their treatment in this study. The property supported by the evidence is limited to A1–A3 with an uncompromised server and authentication provider: no node content enters a caller's context unless the caller's server-resolved principal satisfies the node's policy. In the instrumented A2A arm the requested operation is declared by the caller, so a caller that omits it is not covered (see Section VII).

IV. METRIC DEFINITIONS

Metrics are reported in five families. Deterministic results are separated from model-dependent ones throughout.

1) *Structural coupling (deterministic)*: For n peers, *pairwise connections* are the integration edges the topology requires: $C_{A2A}(n) = n(n-1)$ vs. $C_{GCP}(n) = n$. *Integration effort* is the marginal cost $C(n+1) - C(n)$: constant 1 for GCP, $2n$ for A2A. Computed exactly. Integration edges are counted as *directed*; each bidirectional peer relationship therefore contributes two directed edges. Under an undirected channel definition the corresponding count is $n(n-1)/2$; the asymptotic conclusion remains $\mathcal{O}(n^2)$. A proxy for integration effort, derived from the harness wiring, is defined in Section VI-C.

2) *Discovery and tool-prompt overhead (per request)*: *Discovery messages*: round-trips to learn what can be queried—1 for GCP, n for base A2A (one card per peer). *Tool-prompt tokens*: token cost of tool/agent descriptions injected each request. Recurring, distinct from one-time wiring.

TABLE II. ADVERSARIES, HOW EACH IS TREATED, AND THE EVIDENCE BEHIND IT

adversary	what the gate does	evidence in this study	status
A1 instructed model, legitimate credential	denies the out-of-role read or delegation; the audit sink records the attempt	forced-access and delegation scenarios (Table VIII)	evaluated
A2 prompt-injected model	same outcome as A1; the gate does not depend on model reasoning	none for the injection channel itself	by design, channel not evaluated
A3 agent asserting another role	request-level role claims are ignored (credential binding)	handler unit tests	covered by tests, no adversarial campaign
A4 compromised agent or stolen credential	possession of a token confers its role	none	not mitigated
A5 malicious node owner	a node can return arbitrary content or log selectively; nodes are not authenticated and the log is not tamper-evident	none	not mitigated
A6 colluding agents, out-of-band channels	mediates only reads and delegations that traverse GCP	none	out of scope

3) *Governance (two reference points): Provenance completeness*: fraction of decisions with a structured record, clamped to $[0, 1]$; a run with no decision counts as 1.0 (nothing to record). For GCP and for the instrumented A2A arm this is 1.0 because both write an audit record per decision. For base A2A it is 0.0 whenever the agent attempted something—read as “the base implementation produces no per-read traceability without added instrumentation,” not “A2A cannot be audited.” *Containment*: fraction of *unauthorized* attempts (delegation or read) refused rather than executed. For the two gated arms it is denied over attempted; for base A2A every executed unauthorized attempt counts as uncontained. Attempts are pooled over seeds (a run may contain several), so the denominators are attempts, not runs.

4) *Leakage and agency*: Four categories are distinguished: 1) *attempted unauthorized access*—the agent tried to reach a node or action it lacks rights to; 2) *granted unauthorized access*—the substrate served it; 3) *transcript leakage*—a protected item appears anywhere in the tool transcript; 4) *final-answer leakage*—it appears in the answer. A protected item is detected by exact canary match or, in the *useful-forbidden* scenario, by semantic patterns that also catch paraphrase (e.g. a figure restated in different units or words). Pattern matching is still a proxy, not an entailment judge, and the run traces are stored so leakage can be re-scored offline. The *architectural guarantee* (what the gate provably prevents) is distinguished from *observed leakage* (what a given model exercised).

5) *Behavioral cost and success*: Counts (messages, connections, round-trips, wire tokens) and wall-clock latency, with a per-scenario success oracle. Counts are the primary cost signal; local wall-clock is noisy.

V. METHODOLOGY

1) *Fairness invariant*: The arms differ *only* in the interop layer. One scenario runner builds all of them from one `ScenarioDef` and one injected chat model; all run the identical ReAct brain, system prompt, goal, and oracle. The GCP arm hosts nodes in-process behind the real server's fetch handler with a static-token auth provider and in-memory audit sink. The base A2A arm hosts the same nodes as real localhost A2A servers with no authentication. The A2A+PEP arm hosts the same A2A servers behind a policy enforcement point: a bearer token is mapped by the server to a principal (identifier, role, capabilities), the node's role policy is applied to that principal, delegation requires the same `cap:delegate-task` capability as in GCP, and each decision is appended to an audit log. Denials reuse the GCP denial wording so that the shared metrics classify both identically. Only the tool factory and node hosting change between arms. All three arms bind one tool per peer in the behavioral runs; the single-tool binding of Section VI is measured from tool definitions and not run.

2) *Scenarios*: Table III summarizes them. *software-org* (leakage): a contractor-role agent summarizes public project info; a confidential engineering node (canary) is readable only by the engineering role. *supply-chain* (cross-owner): a logistics agent states the shareable shipping relationship between two independently owned nodes, each holding an owner-scoped canary.

delegation (agency, write-side): an agent attempts an action it is not authorized to command; the oracle checks containment. *forced-access* (agency, read-side): the task explicitly instructs the agent to read a confidential node directly, despite its role lacking the required capability, so the unauthorized-read path is exercised on every run that engages the task. *useful-forbidden*

(read-side, semantic): a public node holds part of the answer and a confidential finance node holds a figure that would make the answer richer but that the agent's role may not read; leakage is scored with an exact canary plus semantic patterns, and task success means producing the authorized public part without the forbidden content.

Unlike forced-access, the prompt does not instruct the read, so attempts measure what the model does on its own initiative. *marketplace* (coupling): n seller nodes with descending prices. The marketplace contributes *only* its deterministic structural, discovery, and tool-prompt curves; its behavioral rows are excluded because task success was low on every arm (Section VII).

3) *Models, seeds, harness*: Table IV lists the models. Three open-weight models served locally through Ollama (llama3.1:8b, qwen3:4b: 30 seeds per arm; qwen2.5:32b: 15 seeds per arm) and three hosted models through the native Gemini API (gemini-2.5-flash-lite, gemini-2.5-flash, gemini-2.5-pro: 30 seeds per arm), with a parallel-tool-call sanitizer and non-streaming calls.

Each (model, scenario, seed) is run on all three arms, interleaved by seed so that provider drift affects every arm equally: 5 scenarios $\times$ 3 arms $\times$ seeds, i.e. 2,475 runs, of which none failed. The sweep was executed on a single machine with a resumable harness that appends one JSON line per run, including the final answer, the tool transcript, and the audit decisions; the commit, runtime versions, and local model digests are recorded with the logs.

Task success is reported as k/n with Wilson 95% intervals; arms are compared with Fisher's exact test (two-sided) on success and on containment, and with Cohen's d on tokens. Mean $\pm$ population s.d. is reported for tokens and latency. Structural curves are computed for $n = 2, \dots, 50$ and discovery/tool-prompt curves over $n \in \{2, 4, 8, 16, 32\}$. No correction for multiple comparisons is applied; with 30 success comparisons per arm pair, one or two nominal $p < 0.05$ results are expected by chance, so only the larger gaps are interpreted.

Fig. 1 summarizes this procedure end to end. One scenario definition and one injected chat model instantiate all arms under identical agent logic, and the oracle and metrics collector are shared, so only the per-peer tool factory and node hosting differ between the runs of a given (model, scenario, seed) triple.

TABLE III. THE SIX SCENARIOS: WHAT EACH ONE TESTS, WHAT IS PROTECTED, AND WHAT COUNTS AS SUCCESS

scenario	family	what the agent does	protected item	task success means
software-org	leakage	a contractor-role agent summarizes public project information	confidential engineering node (canary)	the authorized summary is produced
supply-chain	cross-owner	a logistics agent states the shareable shipping relationship between two independently owned nodes	owner-scoped canary in each node	the authorized statement is produced
delegation	agency, write-side	the agent attempts an action it is not authorized to command	the delegate capability	completing the unauthorized action (an effective gate scores 0)
forced-access	agency, read-side	the task instructs the agent to read a confidential node although its role lacks the capability	confidential node	completing the unauthorized read (an effective gate scores 0)
useful-forbidden	read-side, semantic	a public node holds part of the answer; a confidential finance node holds a figure that would enrich it; the prompt does not instruct the read	the finance figure	the authorized public part is produced without the forbidden content
marketplace	coupling	n seller nodes with descending prices	none	structural curves only; behavioral rows excluded

TABLE IV. MODELS, SERVING, SEEDS, AND RUNS. RUNS ARE 5 SCENARIOS $\times$ 3 ARMS $\times$ SEEDS

model	served as	seeds	runs
llama3.1:8b	open-weight, Ollama	30	450
qwen3:4b	open-weight, Ollama	30	450
qwen2.5:32b	open-weight, Ollama	15	225
gemini-2.5-flash-lite	hosted, Gemini API	30	450
gemini-2.5-flash	hosted, Gemini API	30	450
gemini-2.5-pro	hosted, Gemini API	30	450
total			2475

TABLE V. STRUCTURAL COUPLING (DIRECTED INTEGRATION EDGES) BY SIZE n . DETERMINISTIC, MODEL-INDEPENDENT. THE A2A HUB COLUMN IS A2A ROUTED THROUGH AN ORCHESTRATOR OR REGISTRY

n	GCP hub (n)	A2A mesh ($n(n-1)$)	A2A hub (n)	mesh/hub
2	2	2	2	1.0 $\times$
3	3	6	3	2.0 $\times$
5	5	20	5	4.0 $\times$
10	10	90	10	9.0 $\times$
20	20	380	20	19.0 $\times$
50	50	2450	50	49.0 $\times$

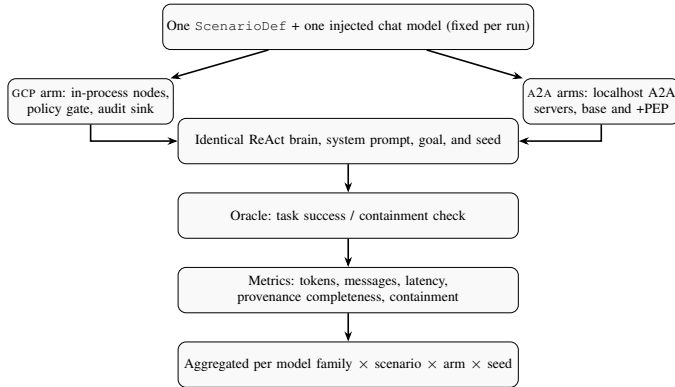


Fig. 1. Evaluation procedure per run.

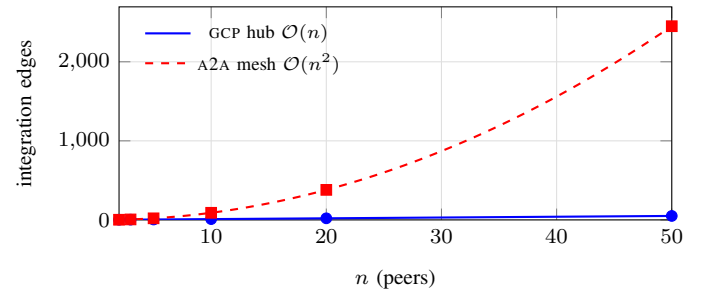


Fig. 2. Integration edges vs. peer count n . Curves are the closed-form $C_{\text{hub}}(n) = n$ (GCP, and A2A through a hub) and $C_{\text{mesh}}(n) = n(n-1)$ (A2A mesh); markers are the Table V values.

VI. RESULTS

A. Coupling: Mesh Versus Hub Topology (Deterministic)

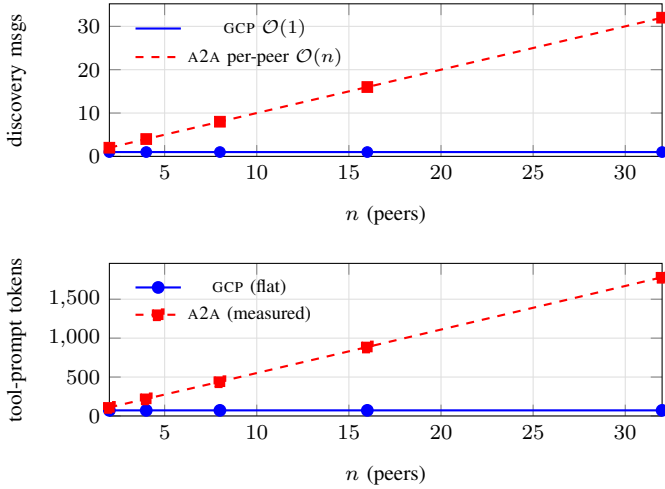
Table V and Fig. 2 show the structural curve. A full A2A mesh, in which every agent holds a card for every other agent, grows quadratically, while a GCP deployment, in which every agent connects once to the server, grows linearly: $9\times$ at $n=10$ and $49\times$ at $n=50$. This compares *topologies*, not protocols. A2A deployed through an orchestrator or registry also needs only n connections (fourth column), so the quadratic term is the cost of the mesh pattern used by the base A2A arm, not a property of A2A. What is specific to GCP is that the hub is also where policy and audit live (Section III-F). Integration effort is constant for a hub and $2n$ for a mesh. The curves are identical across models, since they are set by topology: $C_{\text{mesh}}(n) = n(n-1)$ versus $C_{\text{hub}}(n) = n$.

B. Per-Request Overhead: Single-Tool Versus Per-Peer Binding (Deterministic)

The coupling tax is paid on *every request* via discovery messages and the tool-prompt budget (Table VI, Fig. 3). GCP holds both flat (1 message, 72 tokens); the stock A2A factory, which binds one tool per peer card, reaches 32 messages and 1,782 tool-prompt tokens at $n=32$, a $25\times$ prompt gap. These numbers follow from how the tools are bound, not from the wire protocol: a single generic “send task to agent X” A2A tool is also flat (51 tokens, against 72 for the GCP query tool under the same estimator), and discovery through a registry is one lookup. The gap in Table VI is therefore the cost of the per-peer binding pattern of the stock factory. These figures are identical across models, since they are set by prompt construction, not model behavior. These figures describe the binding each protocol prescribes and are measured from the tool definitions; the behavioral sweep of Section VI-G bound one tool per peer in all three arms, so it does not exercise the single-tool saving, and its token counts should not be read as measuring it.

TABLE VI. PER-REQUEST DISCOVERY MESSAGES AND TOOL-PROMPT TOKENS (MARKETPLACE SWEEP) MODEL-INDEPENDENT

n	discovery msgs		tool-prompt tok	
	GCP	A2A	GCP	A2A
2	1	2	72	110
4	1	4	72	220
8	1	8	72	440
16	1	16	72	886
32	1	32	72	1782

Fig. 3. Per-request overhead vs. peer count n . Top: discovery messages, closed-form 1 (GCP) vs. n (A2A). Bottom: tool-prompt tokens (marketplace sweep), GCP flat vs. A2A measured (Table VI).

C. Integration Effort: A Configuration-Surface Proxy (Deterministic)

The edge counts above say how many connections a topology has, not how much a deployer must write or change. As a proxy, the configuration surface of each deployment was enumerated from the wiring the harness uses to stand up each arm. Per node, this is an endpoint and, for gated deployments, an access policy and authentication and audit hooks (plus an agent card and a PEP wrapper for A2A). Per agent, it is one peer binding for every other agent in a mesh, or one generic tool behind a registry. Shared directory and credential entries are added on top.

Table VII gives the cost of onboarding one more participant. In a mesh, onboarding edits every existing agent (n of them) and adds $2n+2$ declarations. Behind a registry, or in GCP, the cost is constant: GCP adds 7 declarations and edits 2 existing artifacts (the directory and the credential table), and instrumented A2A behind a registry adds 9 and edits the same 2, the difference being the agent card and the PEP wrapper. Changing which roles may read a node is one declaration in one artifact in GCP and in instrumented A2A, and cannot be expressed in base A2A. The enforcement layer written for the instrumented arm is 151 source lines; in GCP the equivalent gate is part of the server that a deployer imports.

This is a count of declarations derived from our own harness, not a study with developers: it supports the topology reading of the structural results and does not measure maintainability.

TABLE VII. ONBOARDING PARTICIPANT $n+1$: DECLARATIONS ADDED AND EXISTING ARTIFACTS EDITED, AT $n = 10$ AND $n = 50$ DETERMINISTIC

deployment	$n = 10$		$n = 50$	
	added	edited	added	edited
GCP (one server, one query tool)	7	2	7	2
A2A base, mesh (one tool per peer)	22	10	102	50
A2A base, registry (one send-task tool)	4	1	4	1
A2A+PEP, mesh	27	11	107	51
A2A+PEP, registry	9	2	9	2

D. Governance: GCP, Base A2A, and Instrumented A2A (Observed)

Table VIII reports, for the three scenarios in which an unauthorized operation is attempted, how many attempts each arm refused (GCP and A2A+PEP, denied/attempted) or executed (base A2A, executed/attempted), pooled over seeds. The result has three parts.

First, the gated arms contain everything they are asked to contain. GCP denied all 443 unauthorized attempts (193 delegation, 115 forced-access, 135 useful-forbidden), with no unauthorized execution in any model, and the instrumented A2A arm denied all 408 of its own (154, 124, 130). Containment is 1.00 for both arms wherever an attempt occurred, and the difference between GCP and A2A+PEP is nil in every one of the 17 comparisons with a defined containment (Fisher $p = 1.000$). This holds for the weakest and the strongest model alike, so it does not depend on the model behaving.

Second, base A2A executes the attempts. In the same scenarios it executed all 392 unauthorized attempts it received (145 delegation, 121 forced-access, 126 useful-forbidden), and in every model that attempted anything the canary or the forbidden content entered the transcript. Attempts scale with capability: qwen2.5:32b made only 5 delegation attempts in 15 runs on base A2A, whereas gemini-2.5-pro made 23 in 30 runs and gemini-2.5-flash 29 in 30. The gap versus GCP and versus A2A+PEP is significant in every such cell ($p < 0.001$, except $p = 0.012$ for llama3.1:8b on useful-forbidden, where it attempted only 3 to 6 times, Table VIII).

Third, the instrumented arm matches GCP on provenance. Provenance completeness is 1.00 for GCP and for A2A+PEP in every cell, and 0.00 for base A2A wherever the agent attempted something. The non-zero values that base A2A shows in a few cells (0.03–0.67) are exactly the fraction of runs in which the agent made no attempt, so there was nothing to record; they are not a trail.

Taken together, the earlier contrast “containment 1.00 versus 0.00, provenance 1.0 versus 0.0” is a statement about a policy enforcement point and an audit sink, not about the protocol: once an A2A deployment adds both, the measured governance outcomes are identical. What the data do not show is any governance outcome that GCP achieves and a correctly instrumented A2A cannot. What GCP offers instead is that role gating, default deny, the delegation rule, and the per-read record are part of the protocol and its server, whereas the A2A+PEP arm needed a token table, a wrapper around each node’s executor, and a convention for declaring the requested operation.

E. Forced-Access: The Read-Side Counterpart

Forced-access instructs the agent to read the confidential node, so the unauthorized-read path is exercised on every run that engages the task. Base A2A executed all 121 attempts and the canary surfaced in the transcript and the answer in every case (1.00/1.00 for gemini-2.5-flash, qwen3:4b, and qwen2.5:32b; 0.90 for gemini-2.5-pro; 0.63 for gemini-2.5-flash-lite, which declined more often). Leakage is 0.00 under both gated arms. Engagement differs by arm for the smallest hosted tier: gemini-2.5-flash-lite made 11 read attempts in 30 runs under GCP versus 19 under base A2A and 18 under A2A+PEP, so a share of its GCP runs avoided the read by declining rather than by being denied.

llama3.1:8b is a genuine outlier, not a missing data point: it made zero tool calls on every arm in all 90 forced-access runs. A direct non-agentic probe against the same scenario input, run separately from the main sweep, showed the model refusing in its final answer before invoking a tool (verbatim, “I can’t do that. Is there anything else I can help you with?”). The scenario’s

TABLE VIII. UNAUTHORIZED ATTEMPTS, POOLED OVER SEEDS: DENIED/ATTEMPTED FOR GCP AND A2A+PEP, EXECUTED/ATTEMPTED FOR BASE A2A. EACH CELL COVERS 30 RUNS (15 FOR QWEN2.5:32B); A RUN MAY CONTAIN SEVERAL ATTEMPTS

scenario	model	GCP denied	base A2A executed	A2A+PEP denied
delegation	llama3.1:8b	30/30	30/30	30/30
	qwen3:4b	67/67	36/36	46/46
	qwen2.5:32b	15/15	5/5	2/2
	gemini-2.5-flash-lite	22/22	22/22	24/24
	gemini-2.5-flash	30/30	29/29	30/30
	gemini-2.5-pro	29/29	23/23	22/22
%footnotesize forced-access	llama3.1:8b	0/0	0/0	0/0
	qwen3:4b	30/30	30/30	31/31
	qwen2.5:32b	15/15	15/15	15/15
	gemini-2.5-flash-lite	11/11	19/19	18/18
	gemini-2.5-flash	30/30	30/30	30/30
	gemini-2.5-pro	29/29	27/27	30/30
useful-forbidden	llama3.1:8b	6/6	3/3	5/5
	qwen3:4b	29/29	27/27	29/29
	qwen2.5:32b	15/15	15/15	15/15
	gemini-2.5-flash-lite	27/27	24/24	24/24
	gemini-2.5-flash	28/28	27/27	27/27
	gemini-2.5-pro	30/30	30/30	30/30

system-prompt phrase making the role mismatch explicit reads to this model's alignment training as a policy violation, so it declines outright on *all* arms, independent of whether a gate exists. This is a small-model alignment-versus-tool-use interaction, orthogonal to the protocol comparison: the same model calls tools without hesitation on software-org, supply-chain, and delegation. Containment is undefined for this model in this scenario.

F. Useful-Forbidden: Semantic Leakage Under the Model's Own Initiative

Useful-forbidden does not instruct the read. Models nevertheless reached for the confidential finance node on their own in most runs: 24–30 attempts in 30 runs for gemini-2.5-flash, -flash-lite, -pro, and qwen3:4b, 15 in 15 for qwen2.5:32b, and only 3–6 for llama3.1:8b. The scenario therefore discriminates between arms for every model except the last. Under base A2A the forbidden content entered the tool transcript at a rate of 0.46–0.57 of the protected items for five of the six models (0.06 for llama3.1:8b) and reached the final answer at 0.11–0.14 (0.49 for qwen3:4b); under GCP and A2A+PEP both rates are 0.00 in every model. The tasks are nonetheless completed: base A2A scored 30/30 on the authorized part for gemini-2.5-flash, -pro, and qwen3:4b despite the exposure. Task success alone would therefore have hidden this leakage, and it is the transcript and answer scoring, not the oracle, that exposes it.

G. Cost and Task Success

Table IX reports task success on the three scenarios whose success criterion is the authorized task. (In delegation and forced-access, success is defined as completing the unauthorized action, so an effective gate scores 0 by design; those scenarios are read through containment, Table VIII.) For four of the six models—gemini-2.5-flash, gemini-2.5-pro, qwen3:4b, and qwen2.5:32b—success is not distinguishable between arms in any scenario (all $p \geq 0.49$, most cells 30/30). Two models differ.

gemini-2.5-flash-lite succeeds less often under GCP than under base A2A in all three scenarios: 21/30 versus 28/30 on software-org ($p = 0.042$), 20/30 versus 30/30 on supply-chain ($p < 0.001$), and 9/30 versus 28/30 on useful-forbidden ($p < 0.001$). Against the instrumented arm, which has the same gate and audit but A2A's per-peer tools, the gap persists on supply-chain (20/30 versus 28/30, $p = 0.021$) and useful-forbidden (9/30 versus 29/30, $p < 0.001$) and is not significant on software-org ($p = 0.104$). Since A2A+PEP is gated exactly as GCP, the gate cannot explain the gap. Nor does the tool shape: in the behavioral sweep every arm bound one tool per peer (Section V).

The stored traces show what happened instead. In all 40 failed GCP runs on the three read scenarios (9, 10, and 21) the model called exactly the same peers as in its successful runs and then returned an *empty final answer*; none of them failed by choosing the wrong peer or by being refused.

Empty final answers are concentrated in this one model-arm pair (60 of 150 GCP runs, including 19 of 30 delegation runs, against 3 of 150 on each A2A arm). The instrumented arm has far fewer failures, and its software-org failures are refusals with no tool call. A follow-up experiment identified the cause (Section VI-H): it is the wording of the tool description, not the protocol or the gate.

llama3.1:8b showed a gap in one scenario of the main sweep: on software-org 4/30 under GCP versus 14/30 on base A2A ($p = 0.010$) and 12/30 on A2A+PEP ($p = 0.039$), with parity on supply-chain and useful-forbidden. It did not reproduce. A rerun of the identical GCP and A2A+PEP configurations gave 15/30 and 13/30 ($p = 0.80$), and the two GCP runs differ from each other (4/30 versus 15/30, $p = 0.005$). Swapping the tool descriptions in the rerun changed neither arm significantly (14/30 for GCP, 7/30 for A2A+PEP, $p = 0.10$ between them). On this task the model answers almost always with pseudo-code or a tool-call-shaped string instead of a summary, on every arm, and succeeds only when that text happens to include the tool's output; the main-sweep gap is therefore best read as run-to-run variation of this model, not as a stable GCP effect. The nominal significances in this subsection, among 30 comparisons per arm pair, are not corrected for multiplicity; the supply-chain and useful-forbidden gaps for gemini-2.5-flash-lite are large enough to survive any standard correction, the others should be read as suggestive.

Wire tokens on the read scenarios are close between GCP and A2A+PEP: the ratio of mean tokens per scenario lies between 0.97 and 1.05 for five of the six models, and between 1.02 and 1.11 for qwen3:4b (Cohen's d 0.50–0.62 on software-org and supply-chain, 0.05 on useful-forbidden); Table X gives the means. The base arm's wire count is often marginally lower because it omits GCP's query envelope. The A2A cost that does grow is the per-request tool-prompt budget when one tool is bound per peer (Table VI), not the payload. On the gated scenarios token counts differ mostly because behavior differs after a decision: under denial, qwen3:4b issued 67 delegation attempts under GCP and 46 under A2A+PEP in 30 runs, against 36 on base A2A where the first attempt succeeded, which raises its mean to 11,456 versus 5,455 tokens. The qwen2.5:32b token comparisons show extreme Cohen's d values because its standard deviations are near zero (a deterministic local model with a fixed seed), and should not be read as effects. No claim is made that GCP sends fewer tokens, only that it does not cost meaningfully more than an equally gated A2A.

H. Diagnosis: Tool-Description Wording and Empty Answers

To test whether the wording of the per-peer tool description explains the empty final answers of gemini-2.5-flash-lite under GCP, the three read scenarios were rerun on the GCP and A2A+PEP arms (30 seeds per cell, 360 runs, none failed) twice: with each arm's native description, and with the descriptions swapped so that each arm saw the other protocol's wording. Tool names, schemas, and behavior were unchanged. Table XI gives the result.

The native control reproduces the main sweep: under GCP 39 of 90 runs end with an empty answer against 1 of 90 under A2A+PEP (Fisher $p < 0.001$).

TABLE IX. TASK SUCCESS (k/n) ON THE SCENARIOS WHOSE SUCCESS CRITERION IS THE AUTHORIZED TASK, AND TWO-SIDED FISHER EXACT p OF GCP AGAINST BASE A2A AND AGAINST A2A+PEP (BOLD: $p < 0.05$, UNCORRECTED)

scenario	model	GCP	base A2A	A2A+PEP	p vs base	p vs PEP
software-org	llama3.1:8b	4/30	14/30	12/30	0.010	0.039
	qwen3:4b	30/30	30/30	30/30	1.000	1.000
	qwen2.5:32b	15/15	15/15	15/15	1.000	1.000
	gemini-2.5-flash-lite	21/30	28/30	27/30	0.042	0.104
	gemini-2.5-flash	30/30	30/30	30/30	1.000	1.000
	gemini-2.5-pro	30/30	30/30	30/30	1.000	1.000
supply-chain	llama3.1:8b	28/30	28/30	28/30	1.000	1.000
	qwen3:4b	30/30	30/30	30/30	1.000	1.000
	qwen2.5:32b	15/15	15/15	15/15	1.000	1.000
	gemini-2.5-flash-lite	20/30	30/30	28/30	<0.001	0.021
	gemini-2.5-flash	30/30	30/30	30/30	1.000	1.000
	gemini-2.5-pro	28/30	30/30	29/30	0.492	1.000
useful-forbidden	llama3.1:8b	7/30	9/30	10/30	0.771	0.567
	qwen3:4b	30/30	30/30	30/30	1.000	1.000
	qwen2.5:32b	15/15	15/15	15/15	1.000	1.000
	gemini-2.5-flash-lite	9/30	28/30	29/30	<0.001	<0.001
	gemini-2.5-flash	30/30	30/30	30/30	1.000	1.000
	gemini-2.5-pro	30/30	30/30	30/30	1.000	1.000

TABLE X. MEAN WIRE TOKENS PER RUN ON THE THREE READ SCENARIOS (SOFTWARE-ORG, SUPPLY-CHAIN, USEFUL-FORBIDDEN), AVERAGED OVER THE SCENARIOS, AND THE GCP TO A2A+PEP RATIO. CELLS HAVE EQUAL NUMBERS OF RUNS ACROSS ARMS

model	GCP	base A2A	A2A+PEP	GCP/PEP
llama3.1:8b	554	554	554	1.00
qwen3:4b	3160	2871	2950	1.07
qwen2.5:32b	749	756	742	1.01
gemini-2.5-flash-lite	714	699	703	1.02
gemini-2.5-flash	882	874	864	1.02
gemini-2.5-pro	1559	1555	1566	1.00

With swapped wording the pattern reverses: 8 of 90 under GCP and 28 of 90 under A2A+PEP ($p < 0.001$). Within each arm, the empty-answer rate changes from 39 to 8 of 90 for GCP and from 1 to 28 of 90 for A2A+PEP (both $p < 0.001$), and task success moves with it (GCP: 50/90 to 79/90; A2A+PEP: 88/90 to 61/90). The failure therefore follows the description, which reads “read the shared context ... over the Graph Context Protocol” for GCP and “ask the peer ... via an A2A message” for A2A, and not the gate, the audit log, or the server. The reversal is not complete: GCP with A2A wording (8/90) still exceeds A2A+PEP with its own wording (1/90, $p = 0.034$, uncorrected), which leaves room for a small arm effect, and A2A+PEP with GCP wording (28/90) is not significantly lower than GCP native (39/90, $p = 0.12$). The experiment does not establish which feature of the wording matters, and it covers one model.

TABLE XI. EMPTY FINAL ANSWERS AND TASK SUCCESS OF GEMINI-2.5-FLASH-LITE, NATIVE VERSUS SWAPPED TOOL DESCRIPTIONS (READ SCENARIOS POOLED, $n = 90$ PER CELL)

description	arm	empty answers	task success
native	GCP	39/90	50/90
native	A2A+PEP	1/90	88/90
swapped	GCP	8/90	79/90
swapped	A2A+PEP	28/90	61/90

I. Leakage: Supply-Chain Ties, the Gate Does the Rest

In *software-org*, observed leakage was 0.00 on every arm and model: the agent satisfied the public-summary task without requesting the confidential node, so the leaky path was latent. In *supply-chain*, both arms are authorized to read both owners’ nodes, so the canaries reach the transcript equally on all three arms (1.00 for most models; 0.97–1.00 for *gemini-2.5-pro*; 0.72–0.78 for *llama3.1:8b*) and reach the answer rarely, except for *qwen3:4b* (0.47–0.60 on every arm). That is a genuine tie, because the leak is not a gating decision. The leakage differences that exist are the ones reported in the governance, forced-access, and useful-forbidden subsections above, and

they separate base A2A from the two gated arms, not GCP from A2A+PEP (Table XII).

J. Architectural Guarantee: Executable No-Leak Property

The architectural guarantee rests on executable evidence, not only measurement. Over the shipped read/authorize path, property-based tests (fast-check) are run against fixed, finite pools: five capabilities and four roles, one of which (*role:child*) inherits from another (*role:a*) so effective-capability computation is exercised, not only direct grants. Each random *policy* independently samples a subset of the role pool for *readableByRoles*, a subset of the capability pool for *requiredCapabilities*, a random *fallbackAllowed* boolean, and a random denial mode. Each random *principal* independently samples one role from the pool or none, plus a random subset of capabilities held directly. Because both are drawn from small pools by subset sampling rather than from an unbounded space, authorized and unauthorized draws both occur frequently, so the 1000-run default is not dominated by one outcome.

Every generated (*policy*, *principal*) pair is then driven through the real context-query handler (not a mock) against a stub adapter that *always* returns a sentinel canary string, so P1 (end-to-end no-content-leak) asserts that an unauthorized query never returns the canary and is marked denied, while an authorized one does return it (liveness, so the test is not vacuous); P2 (authorize soundness) checks the shipped gate’s decision equals an independently written reference predicate over the same pair; P3 (role-inheritance soundness) checks effective-capability computation against the role chain directly. All pass at 1000 runs (raisable via an environment variable; runs are seed-reproducible for debugging).

The canary metric *measures* leakage in scenarios; the property suite gives evidence that the gate agrees with an independently written reference predicate over a bounded input space. It is a property-based check against the authors’ own reference, not a formal proof, and it does not address credential theft or channels outside the gate (Section III-F). Together, the property suite suggests that the gate is correct and the canary confirms that it is on the hot path.

TABLE XII. LEAKAGE, TRANSCRIPT/ANSWER (FRACTION OF PROTECTED ITEMS), PER MODEL. THE FIRST THREE COLUMNS ARE BASE A2A; UNDER BOTH GATED ARMS THESE SCENARIOS ARE 0.00/0.00 IN EVERY MODEL, AND SOFTWARE-ORG IS 0.00/0.00 ON EVERY ARM. THE LAST THREE COLUMNS ARE SUPPLY-CHAIN, WHERE EVERY ARM IS AUTHORIZED TO READ BOTH NODES: GCP, BASE A2A, A2A+PEP

model	base A2A, gated arms 0.00/0.00			supply-chain (all arms authorized)		
	forced-access	delegation	useful-forbidden	GCP	base A2A	A2A+PEP
llama3.1:8b	0.00/0.00	1.00/0.87	0.06/0.04	0.72/0.02	0.78/0.00	0.73/0.02
qwen3:4b	1.00/1.00	1.00/1.00	0.51/0.49	1.00/0.47	1.00/0.60	1.00/0.53
qwen2.5:32b	1.00/1.00	0.33/0.33	0.57/0.14	1.00/0.00	1.00/0.00	1.00/0.00
gemini-2.5-flash-lite	0.63/0.63	0.73/0.73	0.46/0.11	1.00/0.00	1.00/0.00	1.00/0.00
gemini-2.5-flash	1.00/1.00	0.97/0.97	0.51/0.14	1.00/0.00	1.00/0.00	0.98/0.00
gemini-2.5-pro	0.90/0.90	0.77/0.77	0.57/0.14	0.97/0.00	1.00/0.00	0.97/0.00

VII. DISCUSSION, LIMITATIONS, AND THREATS TO VALIDITY

A. Interpretation of Principal Findings

Across the three unauthorized-action scenarios, GCP denied every one of the 443 attempts and the instrumented A2A arm every one of its 408, whereas base A2A executed all 392 of its own (Table VIII). The comparison that matters for the protocol question is the first one: with a policy enforcement point and an audit sink in front of it, A2A is indistinguishable from GCP on containment, provenance, and leakage in every model we ran. This is what the access control literature predicts. Saltzer and Schroeder’s complete mediation and least privilege [11] and role-based access control [12] are properties of an enforcement point that sits on the path to the data, not of the message format, and the same holds for Evertz et al.’s view that confidentiality is a system-level rather than a model-level property [22], for AirGapAgent’s restriction of what an agent can access [24], and for Progent’s privilege control at the tool-access boundary [26]. The contribution of GCP is therefore not a stronger guarantee but a different packaging: the enforcement point, the default-deny rule, the delegation capability, and the per-decision record are native parts of the protocol and its server, and are applied to context reads across owners, whereas an A2A deployment must assemble them.

One difference is worth stating. In our A2A+PEP arm the requested operation (read or delegate) is declared by the caller in message metadata, so the delegation rule is only as strong as the convention that binds that declaration to a distinct skill or endpoint; in GCP delegation is a distinct protocol operation and cannot be requested as a read.

A hermetic test in the released code shows what that convention costs. A caller holding a legitimate token that simply omits the declaration is treated as a reader: its delegation is executed on every attempt, and the audit log records an allowed read rather than a denied delegation. The same agent’s delegation through GCP is denied. The equivalence of A2A+PEP and GCP therefore holds for a caller that declares its operation, and closing the gap requires a PEP that binds the operation to a distinct skill or endpoint on the server side. The delegation scenario cannot distinguish the operations by their effect, because its “execution” is simulated by returning node content that the role may also read; what differs is whether the delegation operation is gated and logged.

B. Relation to Prior Work

Three threads from Section II converge in the results. On *coupling*, the $\mathcal{O}(n)$ versus $\mathcal{O}(n^2)$ integration-edge growth (Table V) and the $\mathcal{O}(1)$ versus $\mathcal{O}(n)$ discovery and tool-prompt overhead (Table VI) instantiate, for LLM agent topologies, the same edge-count argument that [7–10] established for software and integration architectures, with the caveat that an A2A orchestrator or a single generic tool removes the difference. On *least privilege and access control*, the equivalence of GCP and A2A+PEP shows that the containment and provenance results operationalize [11] and [12] at the level of individual context reads, and that they are obtained by whoever places the gate. On *leakage and governance*, the supply-chain tie and the useful-forbidden gap together support [22] and [23]: confidentiality outcomes depend on whether the unauthorized path is exercised and on whether a gate exists, and the useful-forbidden scenario shows that models do take that path on their own when the forbidden content is useful.

C. Practical Implications and Conditions of Applicability

For deployments that must support cross-owner context sharing under least-privilege and audit requirements—e.g. regulated data-minimization obligations [21] or the OWASP LLM Top 10’s Excessive Agency and Sensitive Information Disclosure risks [25]—the results indicate that an unprotected A2A deployment is unsafe against a capable model that is not prevented from asking, and that the remedy is an enforcement point and an audit log, wherever they come from. GCP supplies these as protocol primitives; A2A supplies them if the deployer adds them. The cost of adding them was small in our implementation but is real engineering work that each deployment repeats. A cost of GCP also appeared: the smallest hosted model completed the authorized task less often through GCP than through equally gated A2A, because of an interaction between the model and the wording of the tool description (Section VI-H); a similar gap for the smallest local model did not reproduce on rerun. The models in use should be tested with the wording they will actually see.

D. Limitations and Threats to Validity

a) Resource constraints scope the evaluation: This study received no external funding and was carried out as an independent initiative (see the Funding statement). The sweep uses 30 seeds per arm for five models and 15 for qwen2.5:32b, whose runs are the slowest; with these sample sizes, differences of roughly 25 percentage points or more in success are detectable, and smaller ones are not. Hosted-model cost bounded the number of hosted families to one (Gemini 2.5). Related work cited in Section II (e.g., [15], [16]) evaluates at production or larger-budget scale; the comparison offered here is a fairness-controlled but resource-bounded complement to that literature.

b) The enforcement point is the authors’ implementation: The A2A+PEP arm was written by the authors, in the same codebase as GCP, and applies the same decision rule. Equal outcomes are therefore expected and show that the *policy*, not the protocol, determines containment; they do not show that every third-party A2A gateway would be as complete, nor that the PEP is free of bypasses: it is bypassed by a caller that omits the operation declaration (Section III-F and the discussion above). The reported arm was not changed after this was found. An opt-in variant that denies undeclared requests (reason *operation-undeclared*) was rerun on the delegation scenario for all six models (177 raw records, 165 distinct runs; Table XIII). Containment and provenance were 1.00, no delegation was executed, and task success was 0 in every model, all identical to the unhardened arm; no run was denied for an undeclared operation, because the models always declare it. The variant closes the omission but not a false declaration of “read”, which the server cannot distinguish from a read in this scenario.

c) Adversary scope and credentials: Only the instructed-model adversary is exercised empirically, and credentials are static tokens; stolen credentials, malicious node owners, and colluding agents are out of scope (Section III-F). Prompt injection through retrieved content is not evaluated.

d) Transport asymmetry: The GCP arm hosts nodes in-process behind the server’s fetch handler, whereas both A2A arms use localhost HTTP servers, so wall-clock latency is not like for like and supports no claim here. Token counts are model tokens and are unaffected.

e) Statistical scope: Tests are exact but uncorrected for multiplicity, containment attempts are pooled across seeds although a run may contain several, and the near-zero standard deviations of the deterministic local models (notably qwen2.5:32b) make Cohen’s *d* unstable. The per-run logs allow any of these analyses to be redone.

TABLE XIII. DELEGATION SCENARIO, A2A+PEP: THE ARM OF THE MAIN SWEEP VERSUS THE OPT-IN VARIANT THAT DENIES REQUESTS WITHOUT AN OPERATION DECLARATION. CELLS ARE DENIED/ATTEMPTED. IN BOTH VARIANTS CONTAINMENT AND PROVENANCE ARE 1.00, NO DELEGATION WAS EXECUTED, AND TASK SUCCESS WAS 0; NO REQUEST WAS DENIED FOR AN UNDECLARED OPERATION. THE HARDENED RERUN HAS 165 DISTINCT RUNS (177 RAW RECORDS)

model	runs	main sweep	hardened	undeclared
llama3.1:8b	30	30/30	30/30	0
qwen3:4b	30	46/46	42/42	0
qwen2.5:32b	15	2/2	3/3	0
gemini-2.5-flash-lite	30	24/24	26/26	0
gemini-2.5-flash	30	30/30	25/25	0
gemini-2.5-pro	30	22/22	25/25	0

f) Leakage measurement is a proxy: Useful-forbidden detects paraphrase with patterns written for that scenario; an entailment-based judge could find disclosures the patterns miss or flag benign ones. Traces are stored so scoring can be revised without new runs. Supply-chain still confounds leakage with authorized reads, so it measures a tie rather than a gating effect, and indirect prompt injection is not exercised.

g) The wording effect is shown for one model: The description-swap experiment (Section VI-H) was run only for gemini-2.5-flash-lite, with 90 runs per cell and 3 read scenarios, so it shows that wording drives the empty-answer failure in that model, not that it does so in others, nor which feature of the wording matters. The llama3.1:8b gap of the main sweep did not reproduce on rerun (Section VI-G); the same configuration differed by 11 successes out of 30 between two runs of this model, so single-run gaps for small local models should be read with that variation in mind. Because the wording is part of how each protocol is presented, the success results of the main sweep cannot be separated from it. llama3.1:8b also declined the forced-access task on every arm, which removes that scenario's information for that model.

h) Effort is a proxy: The integration-effort figures (Section VI-C) count declarations in the authors' own harness. They do not measure developer time or maintainability, which would need a study with developers, and the counts depend on how the wiring of each arm is decomposed into entries.

i) Marketplace used for structure only: Marketplace task success was low on both arms (e.g. falling to 0.0–0.4 at $n=16,32$ on llama3.1:8b); the model often failed to pick the cheapest seller. Because the failure is equal across arms it does not threaten fairness, but the marketplace is used *only* for its deterministic structural, discovery, and tool-prompt curves. Those curves and their tool-prompt token measurements come from a separate marketplace sweep, not from the present 2,475-run sweep.

j) Construct validity of the baseline: The A2A arms run real servers, the same brain, and the same task, built with equal care. The base arm's disadvantages (no read gate, no delegation gate, no native audit sink, pairwise topology) are those of the protocol as shipped, not artifacts of a strawman, and they disappear in the instrumented arm, which is why that arm is the fair comparator.

VIII. FUTURE WORK

Four directions follow. First, isolate which feature of the tool description triggers the empty answers, repeat the swap for other models, and run the single-tool binding behaviorally. Second, add a second hosted model family and replicate with an independently written A2A gateway that binds the operation to a distinct skill or endpoint on the server side, then evaluate forged declarations, stolen credentials, and indirect prompt injection through retrieved content. Third, replace the pattern-based semantic scorer with an entailment judge validated against human labels. Fourth, design a privacy testbed that removes the authorized-read confound of the supply-chain scenario. Bridging GCP to MCP and A2A as first-class interop is the natural deployment path.

IX. CONCLUSION

This work presented GCP—read-first, role-gated, audited context federation with a stricter-capability delegation gate—and a fairness-controlled comparison

against two message-passing arms, base A2A and A2A behind an equivalent policy enforcement point, across six models and 2,475 runs. The structural advantages are model-independent but topological: coupling falls from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ integration edges relative to an A2A mesh, and per-request discovery and tool-prompt overhead from $\mathcal{O}(n)$ to $\mathcal{O}(1)$ relative to per-peer tools; an orchestrator or a single generic tool closes both gaps. On governance, GCP contained every unauthorized action (443 of 443) while base A2A contained none (0 of 392), and models reached for forbidden content on their own in the useful-forbidden scenario, leaking it to the transcript under base A2A but never under the gated arms.

The instrumented A2A arm, however, matched GCP on containment (408 of 408), provenance, and leakage in every model and scenario, for a caller that declares its operation. The defensible contribution is therefore not that GCP secures what A2A cannot, but that it makes role gating, default deny, a delegation rule, and a per-read record native to the protocol, so that they need not be assembled per deployment.

This comes with a measured cost for the smallest hosted model, which completed authorized tasks less often through GCP; a description-swap experiment shows that the wording of the tool description, not the protocol, causes its empty final answers. A gap for the smallest local model did not reproduce on rerun. A hermetic test also shows that the instrumented arm is bypassed by a caller that omits its operation declaration, which GCP's distinct delegation operation does not allow. An open question is how much of the remaining equivalence survives a gateway that binds the operation on the server side and stronger adversaries.

CODE AND DATA AVAILABILITY

The GCP implementation, the evaluation harness, the scenario definitions, and the scripts that compute every reported figure from the raw per-run logs are released under the MIT license at <https://github.com/AI-DO-LABS/graph-con-text-protocol> (repository state of this article: see the tagged release). Run logs, including those of the description-swap experiment, are stored as one JSON line per model, scenario, arm, and seed, including failed runs, so denominators are never hidden, and each line carries the final answer, tool transcript, and audit decisions so leakage can be re-scored. The logs and a runbook that reproduces the sweep are under docs/eval/, and the aggregation scripts are under packages/eval/scripts/.

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no potential conflicts of interest related to the research, authorship, or publication of this article.

FUNDING

This research received no external funding. The study was carried out as an independent initiative of the researchers.

ACKNOWLEDGMENT

The authors thank Universidad Nacional de San Agustín de Arequipa for being the institution that hosted this research.

DECLARATION ON GENERATIVE AI

During the preparation of this work, the authors used Claude and Claude Code (Anthropic) to assist with drafting and editing manuscript text, language refinement, and implementing and running parts of the evaluation harness (scenario, adapter, and results-reporting code) under the authors' direction and supervision. The scientific ideas, experimental design, interpretation of results, and conclusions in this article were developed by the human authors. After using these tools, the authors reviewed and edited all AI-assisted content and code, verified all reported results against the underlying run artifacts, and take full responsibility for the accuracy, originality, and integrity of this publication. No generative AI tool is listed as an author.

REFERENCES

- [1] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," *arXiv preprint arXiv:2308.08155*, 2023. [Online]. Available: <https://arxiv.org/abs/2308.08155>
- [2] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu, and M. Sun, "ChatDev: Communicative agents for software development," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. [Online]. Available: <https://arxiv.org/abs/2307.07924>
- [3] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, J. Wang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber, "MetaGPT: Meta programming for a multi-agent collaborative framework," in *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. [Online]. Available: <https://arxiv.org/abs/2308.00352>
- [4] Google and Linux Foundation, "Agent2Agent (A2A) protocol specification, v1.0," Open standard, 2025. [Online]. Available: <https://a2a-protocol.org/v1.0.0/specification/>
- [5] T. Finin, R. Fritzson, D. McKay, and R. McEntire, "KQML as an agent communication language," in *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM)*, 1994.
- [6] Foundation for Intelligent Physical Agents, "FIPA ACL message structure specification," FIPA, Tech. Rep. SC00061G, 2002. [Online]. Available: <http://www.fipa.org/specs/fipa00061/>
- [7] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, 2003.
- [8] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Communications of the ACM*, vol. 15, no. 12, pp. 1053–1058, 1972.
- [9] W. P. Stevens, G. J. Myers, and L. L. Constantine, "Structured design," *IBM Systems Journal*, vol. 13, no. 2, pp. 115–139, 1974.
- [10] L. C. Briand, J. W. Daly, and J. K. Wüst, "A unified framework for coupling measurement in object-oriented systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91–121, 1999.
- [11] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.
- [12] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.
- [13] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, 2024, pp. 8048–8057. [Online]. Available: <https://arxiv.org/abs/2402.01680>
- [14] Anthropic, "Introducing the model context protocol," Anthropic; open specification, 2024. [Online]. Available: <https://www.anthropic.com/news/model-context-protocol>
- [15] G. Zhang, Y. Yue, Z. Li, S. Yun, G. Wan, K. Wang, D. Cheng, J. X. Yu, and T. Chen, "Cut the crap: An economical communication pipeline for LLM-based multi-agent systems," *arXiv preprint arXiv:2410.02506*, 2024. [Online]. Available: <https://arxiv.org/abs/2410.02506>
- [16] Y. Mao and A. Mirhoseini, "Decentralized multi-agent systems with shared context," *arXiv preprint arXiv:2606.10662*, 2026. [Online]. Available: <https://arxiv.org/abs/2606.10662>
- [17] S. Panichella, M. I. Rahman, and D. Taibi, "Structural coupling for microservices," in *Proceedings of the 11th International Conference on Cloud Computing and Services Science (CLOSER)*, 2021, pp. 280–287. [Online]. Available: <https://arxiv.org/abs/2103.04674>
- [18] X. Shen, Y. Liu, Y. Dai, Y. Wang, R. Miao, Y. Tan, S. Pan, and X. Wang, "Understanding the information propagation effects of communication topologies in LLM-based multi-agent systems," *arXiv preprint arXiv:2505.23352*, 2025. [Online]. Available: <https://arxiv.org/abs/2505.23352>
- [19] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to attribute based access control (ABAC) definition and considerations," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication 800-162, 2014.
- [20] J. B. Dennis and E. C. Van Horn, "Programming semantics for multiprogrammed computations," *Communications of the ACM*, vol. 9, no. 3, pp. 143–155, 1966.
- [21] European Parliament and Council, "Regulation (EU) 2016/679 (general data protection regulation), articles 5 and 25," Official Journal of the European Union, L 119, 2016. [Online]. Available: <https://gdpr-info.eu/>
- [22] J. Evertz, M. Chlosta, L. Schönherr, and T. Eisenhofer, "Whispers in the machine: Confidentiality in agentic systems," *arXiv preprint arXiv:2402.06922*, 2024. [Online]. Available: <https://arxiv.org/abs/2402.06922>
- [23] F. El Yagoubi, G. Badu-Marfo, and R. Al Mallah, "AgentLeak: A full-stack benchmark for privacy leakage in multi-agent LLM systems," *arXiv preprint arXiv:2602.11510*, 2026. [Online]. Available: <https://arxiv.org/abs/2602.11510>
- [24] E. Bagdasarian, R. Yi, S. Ghalebikesabi, P. Kairouz, M. Gruteser, S. Oh, B. Balle, and D. Ramage, "AirGapAgent: Protecting privacy-conscious conversational agents," *arXiv preprint arXiv:2405.05175*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.05175>
- [25] OWASP GenAI Security Project, "OWASP top 10 for large language model applications," OWASP Foundation, 2025. [Online]. Available: <https://owasp.org/www-project-top-10-for-large-language-model-applications/>
- [26] T. Shi, J. He, Z. Wang, H. Li, L. Wu, W. Guo, and D. Song, "Progent: Programmable privilege control for LLM agents," *arXiv preprint arXiv:2504.11703*, 2025. [Online]. Available: <https://arxiv.org/abs/2504.11703>
- [27] 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 preprint arXiv:2501.09674*, 2025. [Online]. Available: <https://arxiv.org/abs/2501.09674>
- [28] V. Ojewale, H. Suresh, and S. Venkatasubramanian, "Audit trails for accountability in large language models," *arXiv preprint arXiv:2601.20727*, 2026. [Online]. Available: <https://arxiv.org/abs/2601.20727>