ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

Trustworthy Agentic AI: A Survey and Taxonomy of Secure Coordination and Hallucination Mitigation in Multi-Agent Large Language Model Systems

Tharakesvulu Vangalapat[1]; Samreen Iftekhar Shaikh[2]

[1]Sr. Principal Data Scientist, AI Architect Broadridge Austin, TX, USA
[2]Instructor, Lewis University Romeoville, Illinois USA

Publication Date: 2026/02/27

Abstract:

- ***Background:***
Large language model (LLM)-based agentic systems are evolving beyond single-turn generators into autonomous, tool-using, multi-agent workflows with persistent memory and self-directed planning. When these agents collaborate, hallucinations no longer remain local; they can propagate across agent boundaries and trigger real-world operational failures.

- ***Objective:***
This paper provides a structured foundation for building secure, reliable, and hallucination-resilient agentic AI by surveying failure modes and proposing a layered trust taxonomy with enforceable controls.

- ***Methods:***
We reviewed literature on LLM reasoning, retrieval-augmented generation (RAG), hallucination detection, multi-agent frameworks, and AI governance (including zero-trust security principles) and catalogued failure modes across reliability and security dimensions.

- ***Results:***
We present a seven-layer trust taxonomy spanning identity, planning, communication, memory, retrieval, execution, and oversight. From this taxonomy, we derive six reusable secure-coordination design patterns and propose a model-agnostic reference architecture for auditable, policy-enforced agentic workflows.

- ***Conclusion:***
Trustworthiness in agentic AI is fundamentally a system property, not merely a model property. The proposed taxonomy and design patterns provide practical, implementation-independent guidance for securing multi-agent LLM deployments in research and high-assurance enterprise contexts.

- ***Plain Language Summary:***
AI systems built from large language models can now plan tasks, use tools, and work together as teams of specialized agents. This collaboration creates new dangers: when one agent fabricates information, the other agents may act on it as though it were true, spreading errors throughout the system. This paper maps where trust breaks down, from agent identity and message passing to memory and tool use, and proposes design rules and a system blueprint for more reliable and secure AI teams. agentic AI, multi-agent systems, trustworthy AI, hallucination mitigation, retrieval-augmented generation, tool use, zero trust, AI governance.

How to Cite: Tharakesvulu Vangalapat; Samreen Iftekhar Shaikh (2026) Trustworthy Agentic AI: A Survey and Taxonomy of Secure Coordination and Hallucination Mitigation in Multi-Agent Large Language Model Systems.
International Journal of Innovative Science and Research Technology, 11(2), 1660-1669.
https://doi.org/10.38124/ijisrt/26feb1090

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

I. INTRODUCTION

Large language models (LLMs) are now foundational to modern AI, driven by advances in transformer architectures [1] and large-scale pretraining [2, 3]. Beyond generating fluent text, these models demonstrate emergent capabilities (multi-step reasoning, code synthesis, and structured decision-making [4, 5]) that have reshaped AI deployment. LLMs are now embedded in *agentic* systems: goal-directed components that plan actions, invoke tools, observe outcomes, and refine behavior across multiple steps [6, 7, 8]. Multi-agent architectures extend this further, coordinating networks of specialized LLMs (planners, executors, critics, and retrievers) to tackle problems beyond any single model's reach [9, 10].

This capability growth carries proportional risk. In single-turn settings, a hallucinated response is a local failure: a human can recognize and discard the error. In agentic workflows, hallucination becomes *operational*: a fabricated result may trigger a real tool call, corrupt a shared memory record, or become the premise of another agent's plan. Across a network of agents, the risk compounds through *propagation*: one agent's unsupported assumption becomes another's trusted input, spreading error before any human can intervene. This compounding dynamic is the central concern of this paper:

Trustworthiness in agentic AI is not a property of any individual model; it is a system-level property that emerges from how agents coordinate, share memory, and execute actions.

➢ *Scope and Contributions*

This paper is a survey and position paper. We do not report experimental results; instead, we synthesize the literature to identify where trust breaks down and how it can be enforced. Our contributions are:

- Survey: A synthesis spanning LLM reasoning and tool use, RAG grounding, hallucination detection, multi-agent orchestration, and AI governance and security.
- Taxonomy: A seven-layer trust taxonomy covering the identity, planning, communication, memory, retrieval, execution, and oversight boundaries in multi-agent systems.
- Design patterns: Six reusable secure-coordination patterns: policy-enforced tool gateways, evidence-carrying messages, typed protocols, verifiable memory, critic loops, and propagation-aware gating.
- Reference architecture: A model-agnostic, end-to-end architecture that maps each trust boundary to an enforceable system component.
- Research agenda: Open problems and benchmark gaps, including propagationaware evaluation metrics and formal verification of agent plans.

II. METHODS

➢ *Literature Search and Inclusion Criteria*

We conducted a structured review of peer-reviewed publications, preprints, and technical reports published between 2017 and 2024. Sources were drawn from Google Scholar, arXiv, ACL Anthology, and NIST databases using the search terms *agentic AI*, *multi-agent LLM*, *hallucination detection*, *retrieval-augmented generation*, *tool use*, *prompt injection*, and *zero trust*. Inclusion criteria required that works (a) described an LLM-based agentic or multi-agent system, (b) analyzed reliability or security failures, or (c) proposed controls, benchmarks, or governance frameworks for agentic deployments. Works focused solely on singleturn generation without agentic components were excluded. Findings were synthesized by identifying recurring failure modes, grouping controls by mechanism, and mapping both onto a coherent trust-boundary framework.

➢ *Foundational Building Blocks*

Understanding agentic failure modes requires familiarity with their underlying components. Below, we review four primary building blocks: structured reasoning, retrieval grounding, tool execution, and multi-agent orchestration.

- Reasoning and Structured Prompting. Chain-of-thought (CoT) prompting encourages models to produce explicit intermediate reasoning steps, improving performance on multi-step tasks [11]. A key limitation is that reasoning chains can be fluent and internally consistent yet factually wrong; the trace alone provides no correctness guarantee. ReAct partially addresses this by interleaving reasoning with external tool queries, grounding inferences in real observations [6]. Program-aided methods such as PAL offload computation to executable code, eliminating certain classes of arithmetic and symbolic error [8]. However, each technique also opens new attack surfaces: adversaries can craft inputs that steer action selection, corrupt parameters, or inject misleading observations.
- Retrieval-Augmented Generation. Rather than relying on knowledge encoded in model weights, RAG systems retrieve relevant passages at inference time and condition generation on them [12], substantially reducing hallucination for knowledge-intensive tasks. REALM extends this principle to pretraining [13], and fusion-in-decoder architectures combine multiple retrieved passages into a single generation [14]. In multi-agent settings, RAG introduces a distinct hazard: agents may retrieve contradictory evidence or exploit retrieved context to justify unsupported claims. We term this practice "citation laundering." The retrieval backend (BM25 or dense retrieval) determines what evidence surfaces and introduces risk when sources can be adversarially influenced [15, 16].
- Tool Use and Action Execution. A model that can call external tools is qualitatively more capable, and more dangerous, than one confined to text generation. Toolformer demonstrates that models can learn to invoke APIs and external functions by generating call annotations during pretraining [7]. WebGPT extends this

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

to browser interaction, demonstrating how tool access changes evaluation and safety requirements [17]. In enterprise settings, agentaccessible tools (databases, ticketing systems, and code execution environments) demand rigorous authorization controls and auditable execution logs. These verification requirements have no equivalent in pure text generation.

- Multi-Agent LLM Frameworks. AutoGen provides a general framework in which specialized agents communicate to decompose and solve complex tasks [9]. Generative Agents demonstrates that agent communities equipped with memory and planning can exhibit rich emergent behaviors [10]. Role decomposition improves task performance but increases coordination risk: agents can misinterpret outputs, fabricate agreement, or forward incorrect results as verified facts.
- AI Safety, Governance, and Security Context. The AI safety community has developed alignment techniques such as Constitutional AI [18] and analyzed the risks of foundation models at scale [19]. The security community has codified zero trust as a framework for continuous, identity-based verification in distributed systems [20]. NIST's AI Risk Management Framework synthesizes these perspectives into actionable governance guidance [21]. Agentic systems require both AI alignment's safety guarantees and security engineering's operational controls.

III. RESULTS

A. Failure Mode Analysis

Hallucination in individual language models is well documented [22], and several benchmarks and detection methods exist to measure and mitigate it [23, 24]. However, how hallucination interacts with the coordination machinery of multiagent systems remains poorly understood. In these settings, a false claim is not merely an incorrect answer; it is a potential action trigger, a memory entry, or a message that other agents treat as ground truth. We identify eight failure modes, grouped into reliability and security failures.

➢ *Reliability Failures*

- (R1) Hallucinated intermediate artifacts. An agent fabricates an object that should have been produced by a prior step, such as a file, a database record, a log entry, or a citation. Downstream agents cannot verify that the artifact was actually created; they accept it and continue, compounding the original error.
- (R2) Hallucinated coordination. Agents may assert that a prior step was completed, or that another agent granted approval, when neither occurred. Such fabrication is common when agents communicate in free-form natural language, which offers no structural guarantee that claimed state transitions are real.
- (R3) Hallucination propagation. Once an unsupported claim enters the workflow (through a message, a memory write, or a plan artifact), it can become the premise of subsequent decisions. Such multi-agent amplification transforms a single wrong answer into a chain of decisions built on a false foundation.
- (R4) Stale or drifting memory. Persistent memory is essential for longhorizon tasks but introduces staleness risk. Without provenance tracking, the system cannot determine whether a retrieved memory entry reflects current reality or an outdated assumption from an earlier task.

➢ *Security Failures*

- (S1) Prompt injection. When agents retrieve content from untrusted sources (web pages, documents, and user-provided inputs), that content may contain crafted instructions that hijack agent behavior [25]. The retrieved "data" becomes a covert instruction channel.
- (S2) Capability escalation. An agent operating outside its designated role (for instance, an "analyst" that should only read data but instead executes a privileged write) constitutes a capability escalation. Escalation typically occurs when tool interfaces lack agent-identity scoping and policy enforcement.
- (S3) Tool output fabrication. An agent may report a false tool result without calling the tool, or call the tool but report a different outcome. Without cryptographic attestation linking outputs to actual executions, fabricated and genuine results are indistinguishable.
- (S4) Memory poisoning. If an attacker can influence shared memory writes (through a compromised agent or crafted input), the poisoned content persists and corrupts future tasks.

➢ *Why Multi-Agent Systems Amplify Failures*

In a single-agent system, failure is bounded: one model produces one wrong output, and a human reviewer can catch it. In a multi-agent pipeline, that error propagates as trusted input. The next agent acts on it, writes the consequence to shared memory, and sends an updated artifact downstream. By the time the mistake surfaces, it has been amplified through multiple layers of confident-seeming agent activity. Tracing the error to its source requires inspecting the entire execution history. The asynchronous, conversational structure of most multi-agent frameworks compounds this: without explicit state transitions, agents routinely assume that prior steps succeeded and that implied approvals are real.

B. Taxonomy of Trust Boundaries

To organize these failure modes, we propose a taxonomy built around *trust boundaries*: system interfaces where one component depends on another and where failures cause harm. Seven boundaries emerge from the analysis, together spanning the full lifecycle of an agentic workflow. Figure 1 presents the taxonomy as a mind map, and Table 1 summarizes each boundary with its failure modes and controls.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

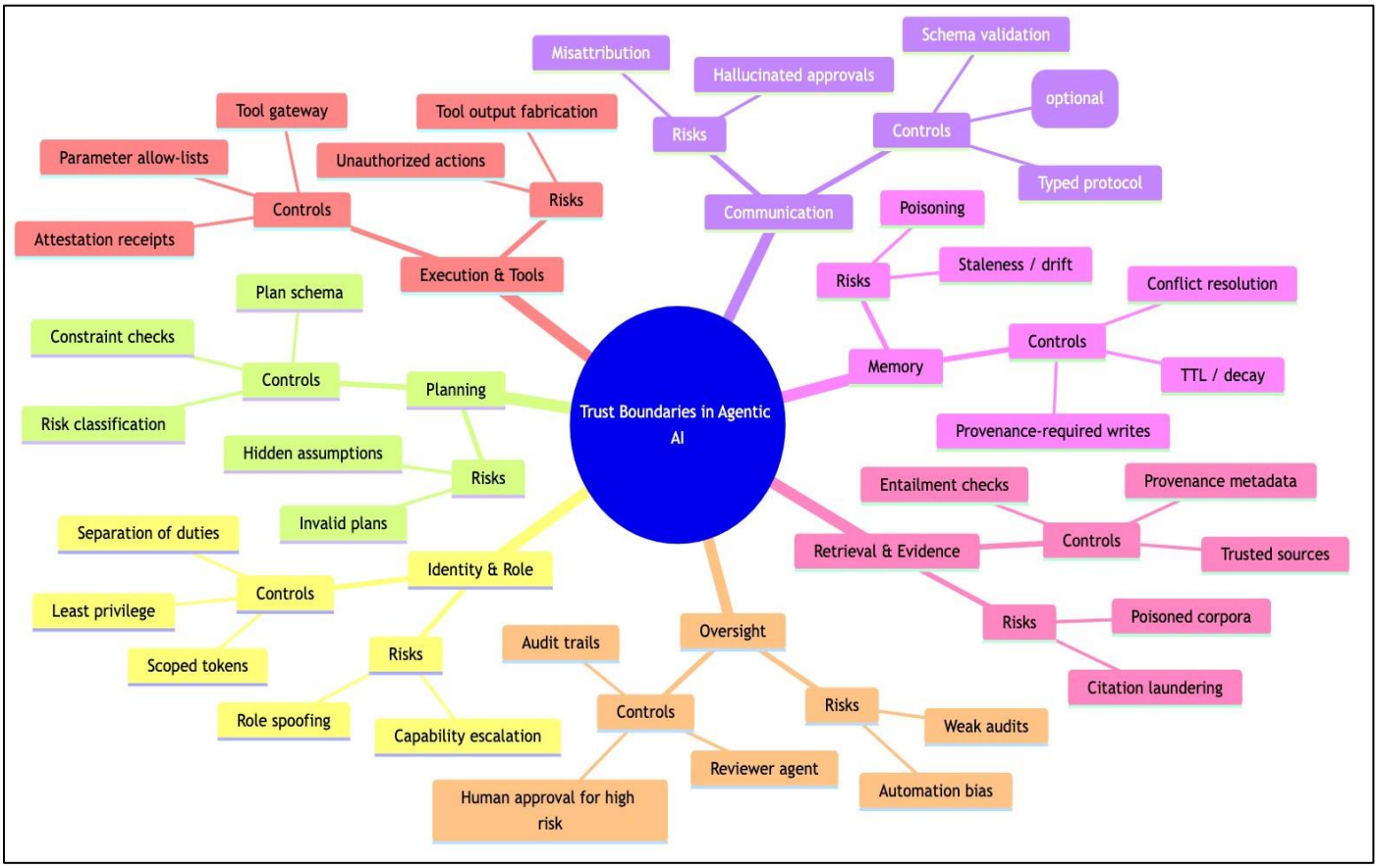

Fig 1 Trust Boundary Taxonomy for Agentic Multi-Agent LLM Systems. Each Boundary Defines a Distinct Verification Surface and Maps to Enforceable Controls.

➢ *Identity and Role Trust*

In conventional systems, identity refers to a user account; in agentic systems, it must also encode the agent's role and permissible action scope. A planner agent should not invoke the same tools as an executor; an analyst should not write to shared memory. Zero trust provides the right framing: rather than trusting authenticated agents by default, the system verifies each request against the agent's role and policy [20, 21].

➢ *Planning Trust*

Even well-aligned models can produce subtly invalid plans: plans that assume nonexistent resources, skip required preconditions, or pursue.

Table 1 Taxonomy of Trust Boundaries in Agentic Multi-Agent LLM Systems

Boundary What is Trusted?	Failure Modes	Representative Controls
Identity Agent iden- & tity, role, Role capability scope	Spoofed roles, escalation	Scoped credentials, least privilege, separation of duties [20]
Planning Plan validity, constraints	Invalid plans, unsafe goals	Plan schemas, constraint checks, risk classification [11, 6]
Communi Mcaetsisoanges And approvals	Hallucinated approvals, ambiguity	Typed schemas, attested messages, quorum approvals
Memory Persistent Facts and state	Poisoning, staleness, drift	Provenancerequired writes, TTL/decay, conflict resolution
Retrieval Retrieved & Ev-sources and Idence citations	Citation laundering, poisoned corpora	Trusted corpora, provenance metadata, entailment checks [12, 13]
ExecutionTool access & and outputs Tools	Fabrication, unauthorized actions	Tool gateway, output attestation, sandboxing [7, 17]
Oversight Humanand automated review	Automation bias, weak audits	Critic agents, audit logs, policy-driven approvals [24]

Goals inconsistent with stated policy. Chain-of-thought reasoning can make such plans appear convincing without improving their correctness [11]. Treating plans as structured artifacts (with explicit steps, preconditions, evidence requirements, and expected outputs) enables automated validation before execution begins.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

➢ *Communication Trust*

Natural language is expressive but ambiguous, and that ambiguity becomes a security vulnerability in multi-agent systems. When agents communicate in free-form prose, claims about what happened, what was approved, and what was agreed cannot be verified. Typed schemas with explicit state transitions enforce shared semantics, while integrity mechanisms such as digital signatures prevent spoofing.

➢ *Memory Trust*

Shared memory enables agents to collaborate on longhorizon tasks but also creates a single point of failure. Any agent (or any adversary who can influence an agent) can introduce false information into the shared record. Provenance-linked writes, combined with evidence backing, allow the system to distinguish well-supported entries from unverified assertions.

➢ *Retrieval and Evidence Trust*

RAG systems condition outputs on retrieved evidence, which is only as trustworthy as its source corpus. Evidence trust requires provenance metadata (source identifier and retrieval timestamp) along with verification that retrieved content genuinely supports each claim, not merely that it is topically related. Dense retrieval methods influence what evidence is available and thus directly affect this boundary [16].

➢ *Execution and Tool Trust*

When a tool is called, two properties must be verified: that the call was authorized and that the result is genuine. Without output attestation, real and fabricated tool results are indistinguishable. Parameter safety checks (rejecting inputs that could trigger destructive operations) and sandboxed execution environments add further protection.

➢ *Oversight Trust*

Human and automated reviewers form the last line of defense but face their own failure mode: automation bias. A reviewer who reads the agent's fluent reasoning summary without independently checking evidence and attestations provides only the appearance of oversight. Structured review processes that specify what must be checked (not merely that review should occur) address this gap [24].

C. Survey of Controls Mapped to the Taxonomy

With the taxonomy established, we organize existing controls from the literature by the trust boundaries they address.

➢ *Model-Level Controls (Necessary But Insufficient)*

Reinforcement learning from human feedback (RLHF) and related alignment techniques reduce the likelihood of harmful or false outputs [26, 27]. Constitutional AI extends this approach, using normative principles to constrain model behavior [18]. These controls are valuable yet operate at the model boundary, not the system level. They cannot verify tool output authenticity or prevent poisoned memory entries from influencing future decisions. System-level controls are therefore essential complements, not alternatives.

➢ *Grounding and Factuality Controls*

RAG and retrieval-augmented pretraining reduce hallucination by anchoring generation to retrieved evidence [12, 13]. Factuality benchmarks (TruthfulQA [23], FActScore [28]) quantify output groundedness. In agentic systems, grounding must extend beyond final answers: fact-asserting messages should carry evidence and provenance so that downstream agents can verify rather than merely accept them.

➢ *Verification and Self-Checking Controls*

Self-refinement pipelines improve output quality by having a model critique and revise its own drafts [29]. SelfCheckGPT detects hallucination by sampling multiple outputs and checking consistency [24]. In multi-agent systems, these ideas generalize naturally to dedicated critic agents. The missing piece is *propagation-aware verification*: mechanisms that detect when an incorrect intermediate claim is about to become a dependency treated as established fact.

➢ *Security Controls and Zero Trust*

Zero trust is the most mature framework for the identity and tool trust boundaries [20]. Adversarial attack research underscores this necessity: prompt injection, jailbreaking, and adversarial perturbations demonstrate that safety cannot rely on model behavior alone [25, 30, 31]. In a multi-agent system, every tool call must be authorized against the calling agent's role and current policy, regardless of prior trust. An executor agent may call tools within its scope with explicit, logged approvals; an analyst may read from the knowledge base but not write to it.

IV. DISCUSSION

The preceding taxonomy and failure-mode analysis are descriptive. This section addresses the prescriptive question: given where trust breaks down, how should agentic systems be built? We present a reference architecture (Section 4.1), six design patterns (Section 4.2), an enterprise control mapping (Section 4.3), and open benchmark gaps (Section 4.4).

➢ *Reference Architecture*

Figure 2 shows a reference architecture that operationalizes the seven trust boundaries as concrete system components. The architecture is model-agnostic: it can overlay any LLM backend, provided that agents emit structured messages and accept orchestration constraints.

Each component maps to one or more trust boundaries. At the security perimeter, the Identity & Role Manager issues scoped credentials and maintains role assignments (identity boundary), while the Policy Engine evaluates every action and message against current policy (planning and execution boundaries). For inter-agent coordination, the Typed Message Bus enforces message schemas and tracks workflow state transitions (communication boundary). On the data side, the Retrieval Layer attaches provenance metadata to retrieved evidence (evidence boundary), and the Memory Store persists facts with provenance links, TTL, and conflict-resolution logic (memory bound- ary). The Tool Gateway mediates every tool execution and attaches attestation receipts to results (execution boundary). For quality assurance, the

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

Reviewer/Critic Service performs structured, evidence-driven verification for high-risk steps (oversight boundary). Finally, the Audit Log records all decisions, inputs, and outputs for compliance review and reproducibility.

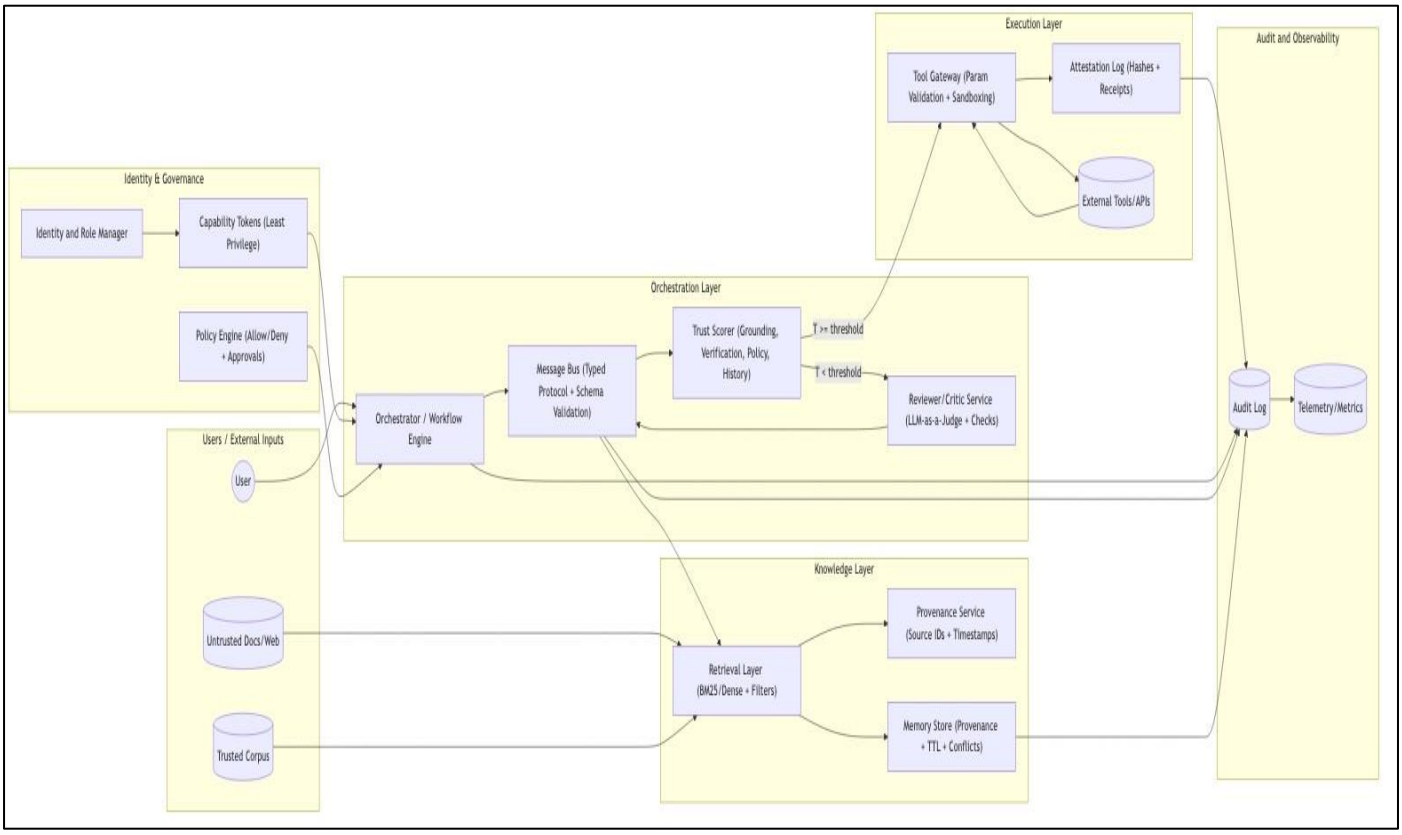

Fig 2 Reference Architecture Mapping Trust Boundaries to Enforceable Components: Identity/Role Management, Policy Engine, Typed Message Bus, Provenance-Aware Retrieval, Verifiable Memory, Tool Gateway with Attestations, and Audit/Telemetry Layer.

➢ *Design Patterns for Secure Coordination*

Six design patterns describe how these components should be used in practice. These patterns distill recurring recommendations from the literature and apply regardless of the underlying technology stack.

- Pattern P1: Policy-enforced tool gateway. Every tool call, without exception, passes through a centralized gateway. The gateway verifies agent identity and role scope, validates parameters against an allow-list, and executes in a sandboxed context where possible. It returns each result with a tamperevident attestation record [7]. Enforcement is unconditional: no agent, however trusted, may bypass the gateway.
- Pattern P2: Evidence-carrying messages. When an agent makes a claim affecting downstream decisions (about the world's state, a previous step's output, or a prior agent's intent), it must attach an explicit evidence reference. Every hop in the message chain carries its own provenance, letting downstream agents trace each claim to its source.
- Pattern P3: Typed coordination protocol. Instead of free-form natural language, the system enforces a typed protocol with six message types: PLAN, REQUEST, APPROVE, EXECUTE, RESULT, and REVIEW. Each type has a defined schema and implies specific state transitions. An agent cannot claim that approval was granted without a signed APPROVE message existing in the message log. Figure 3 illustrates the full protocol as a sequence diagram.
- Pattern P4: Verifiable memory with provenance. Every write to shared memory records four metadata fields alongside the content: (1) the source type (tool output, retrieved evidence, or agent assertion), (2) a provenance identifier linking to the originating event, (3) a timestamp, and (4) an optional time-to-live (TTL). After the TTL expires, the entry is considered stale. Such metadata enables the system to trace any memory entry to its origin, supporting both automated staleness checks and human forensic investigation.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

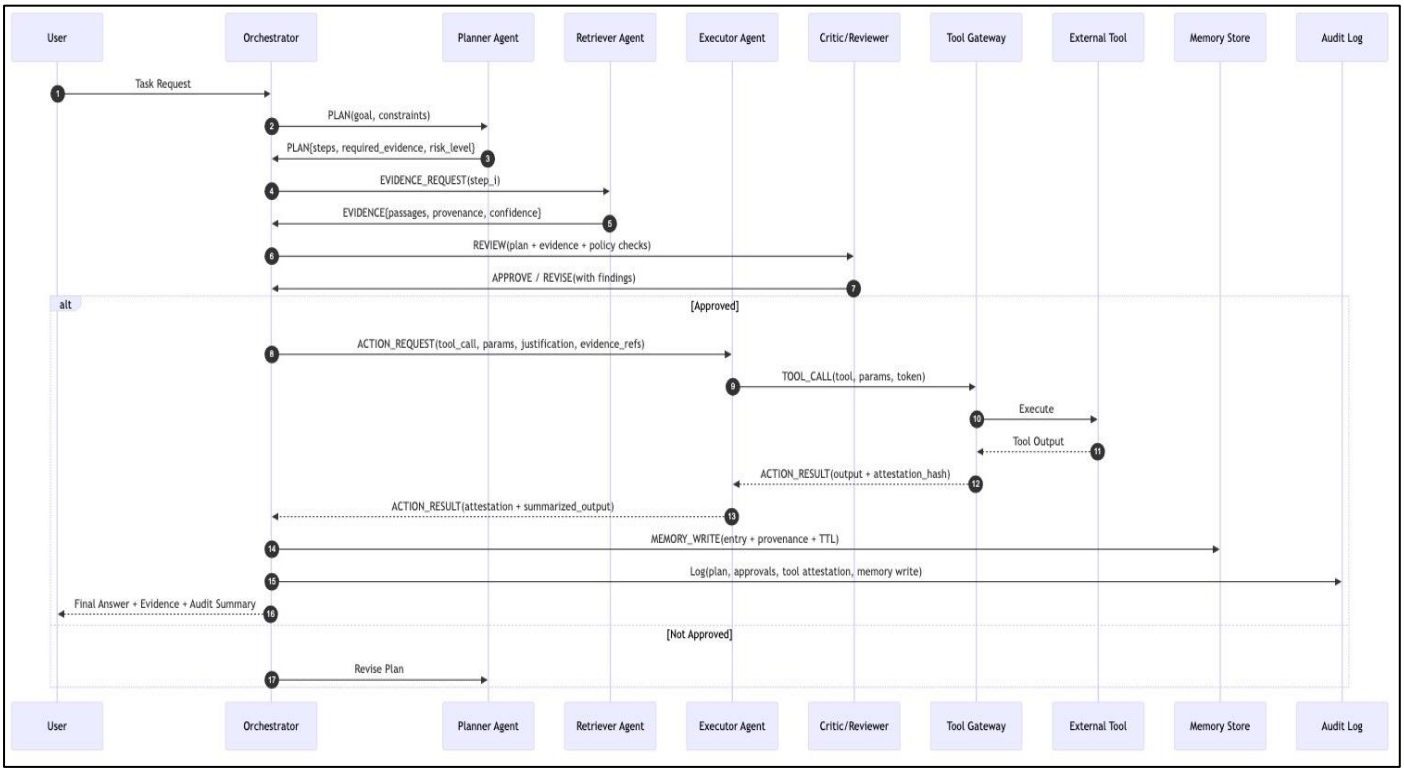

Fig 3 Typed Coordination Protocol Showing the Six Message Types (Plan, Evidence Retrieval, Review, Tool-Gateway Execution, and Audited Memory Update) and the State Transitions they Imply.

- Pattern P5: Critic/judge loop for high-risk actions. Any consequential action (a privileged tool call, a memory write, or an external message) is routed to a reviewer agent or human approver before execution [24]. The reviewer checks specific artifacts: the evidence reference, the policy compliance record, and the tool attestation. This structured process separates genuine oversight from the automation bias described in Section 3.2.
- Pattern P6: Propagation-aware gating. The orchestrator monitors the dependency graph of intermediate claims. When a downstream plan references an upstream claim lacking an evidence reference or tool attestation, the orchestrator quarantines it and requests re-verification before the workflow proceeds. This pattern directly addresses hallucination propagation (R3) and is, to our knowledge, absent from all existing multi-agent benchmarks.

➢ *Enterprise And High-Assurance Mapping*

The proposed taxonomy aligns with established enterprise security frameworks, easing compliance for regulated deployments:

- IAM and least privilege: The identity trust boundary maps directly to identity and access management (IAM) principles and zero-trust architecture [20].
- Change management: Typed protocols and plan schemas supply the structured change records that change-management processes require.
- Data governance: Provenance-linked memory writes and audit logs satisfy data lineage and accountability requirements [21].
- Operational risk: Propagation-aware gating and the critic/judge loop provide the compensating controls that risk frameworks demand for high-impact automated actions.

By exposing verifiable artifacts (attestation records, evidence links, and approval trails), agentic systems become legible to existing governance processes. Compliance evidence then emerges as a byproduct of secure operation rather than a post-hoc assembly.

➢ *Benchmark Gaps and Open Research Challenges*

A recurring gap in the literature is between what existing benchmarks measure and what matters for agentic systems. Most benchmarks evaluate finalanswer correctness or single-turn truthfulness [23]; neither captures workflowlevel properties that determine multi-agent trustworthiness. Figure 4 illustrates propagation-aware gating, the pattern most urgently needing evaluation infrastructure.

We identify four evaluation dimensions that current benchmarks largely ignore:

- Propagation-aware metrics: whether unsupported intermediate claims become dependencies that shape downstream decisions.
- Tool authenticity checks: whether reported tool outputs are genuinely attested or agent-fabricated.
- Policy compliance: whether unauthorized tool calls or approval bypasses occur during workflow execution.
- Audit completeness: whether the audit log suffices to reconstruct every decision made during execution.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

Table 2 maps the six design patterns against the eight failure modes from Section 3.1.

Beyond benchmark design, five research problems remain unresolved.

- Propagation-aware verification. Existing methods assess output truthfulness [22] but do not track claims across a pipeline. New logging formats, dependency-tracking infrastructure, and claim-decomposition methods are required.
- Formal verification of agent plans. When a plan can be represented as an executable graph with explicit preconditions and invariants, temporal logic methods can verify correctness before execution. This is standard in safetycritical software engineering, but applying it to LLM-generated plans (which are noisy, semi-structured, and often implicit) remains largely unexplored.

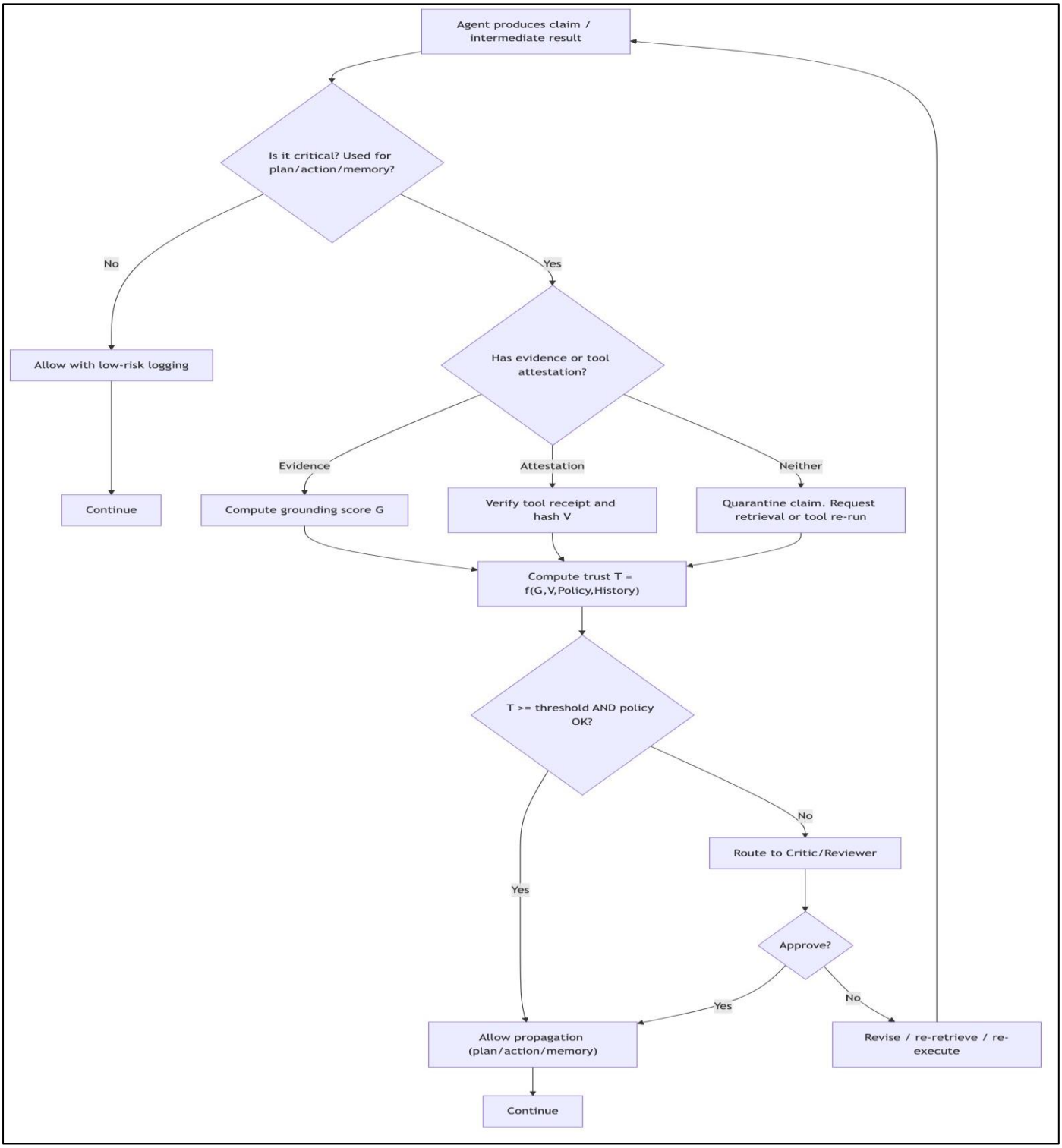

Fig 4 Propagation-Aware Gating: Intermediate Claims are Checked Against Evidence and Attestation Records, Scored for Trust, and Either Forwarded to the Next Agent or Routed to a Reviewer for Re-verification.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

Table 2 Qualitative Mapping: Design Patterns vs. Failure Modes

Control / Pattern	Tool fabrication	Propagation cascades	Memory poisoning	Capability escalation
Policy-enforced gateway (P1)	Strong	Indirect	Indirect	Strong
Evidence messages (P2)	Indirect	Strong	Indirect	Indirect
Typed protocol (P3)	Indirect	Strong	Indirect	Strong
Verifiable memory (P4)	Indirect	Strong	Strong	Indirect
Critic/judge loop (P5)	Strong	Strong	Moderate Strong	
Propagation gating (P6)	Moderate Strong		Moderate Indirect	

- Robustness to adversarial retrieval and prompt injection. Prompt injection attacks demonstrate that when instruction-following models process untrusted text, that text can override intended behavior [25, 30]. Building retrieval pipelines that strictly separate data from instructions, even with adversarially crafted content, remains an open problem.
- Trust calibration and uncertainty quantification. Propagation-aware gating requires a trust score that accurately reflects confidence in each intermediate claim. Current LLM confidence estimates are often poorly calibrated; future approaches may need to combine self-consistency sampling, entailment scores, and tool-based cross-validation to produce reliable signals.
- Privacy and data minimization. In enterprise deployments, agentic retrieval corpora and memory stores frequently contain sensitive personal or commercial data. Differential privacy and data minimization techniques [32, 33] provide principled exposure limits, but integrating them into high-throughput agentic pipelines without sacrificing utility remains unsolved.

V. CONCLUSION

The shift from single-turn language models to networked agentic systems is not merely a quantitative expansion of capability; it is a qualitative change in risk. Hallucinations that were once local and correctable become operational failures that propagate through pipelines, corrupt shared memory, and trigger real-world actions before any human can intervene. Addressing this challenge requires treating trustworthiness as a system-level concern, not merely a modellevel one.

This paper contributes a seven-layer trust taxonomy mapping the failure surfaces of multi-agent LLM systems and six design patterns providing implementationindependent guidance. It also proposes a model-agnostic reference architecture showing how these patterns compose into a complete, auditable system. We identified benchmark gaps and open challenges (propagation-aware evaluation, formal plan verification, robust retrieval, calibrated uncertainty, and privacypreserving memory) that the field must address. The taxonomy and patterns offer a shared foundation for researchers and practitioners working toward that goal.

➢ *Reproducibility Statement*

This paper is a survey and position paper based solely on publicly available literature. No proprietary datasets or restricted experiments were used.

➢ *Ethics Statement*

This research reviews and synthesizes existing published literature; it does not involve human participants, human tissue, personal data, or animal subjects. All procedures comply with applicable ethical standards, and no ethics committee approval was required.

➢ *Data Access Statement*

No new datasets were generated or analyzed in this study. All cited literature is publicly available from the referenced sources.

➢ *Funding*

This research received no external funding.

➢ *Conflict of Interest*

The author declares no conflict of interest.

REFERENCES

[1]. A. Vaswani *et al.*, “Attention is all you need,” in *NeurIPS*, 2017.
[2]. T. Brown *et al.*, “Language models are few-shot learners,” in *NeurIPS*, 2020.
[3]. C. Raffel *et al.*, “Exploring the limits of transfer learning with t5,” *JMLR*, 2020.
[4]. OpenAI, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023. [5] S. Bubeck *et al.*, “Sparks of agi,” *arXiv*, 2023.
[5]. S. Yao *et al.*, “React: Synergizing reasoning and acting in llms,” *ICLR*, 2023.
[6]. T. Schick *et al.*, “Toolformer: Language models can teach themselves to use tools,” *NeurIPS*, 2023.
[7]. L. Gao *et al.*, “Program-aided language models,” *ICML*, 2023.
[8]. Q. Wu *et al.*, “Autogen: Multi-agent conversation framework,” *arXiv*, 2023. [10] J. S. Park *et al.*, “Generative agents,” in *UIST*, 2023.
[9]. J. Wei *et al.*, “Chain-of-thought prompting elicits reasoning in llms,” *NeurIPS*, 2022.
[10]. P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp,” *NeurIPS*, 2020.
[11]. K. Guu *et al.*, “Realm: Retrieval-augmented language model pre-training,” *ICML*, 2020.
[12]. G. Izacard and E. Grave, “Leveraging passage retrieval with generative models,” *arXiv*, 2021.
[13]. S. Robertson and H. Zaragoza, “Bm25 and beyond,” *Foundations and Trends in IR*, 2009.
[14]. O. Khattab and M. Zaharia, “Colbert: Efficient passage search,” *SIGIR*, 2020.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1090

[15]. R. Nakano *et al.*, "Webgpt," *arXiv*, 2021.
[16]. Y. Bai *et al.*, "Constitutional ai," *arXiv*, 2022.
[17]. R. Bommasani *et al.*, "On the opportunities and risks of foundation models," *arXiv*, 2021.
[18]. S. Rose *et al.*, "Zero trust architecture," NIST, Tech. Rep., 2020.
[19]. NIST, "Ai risk management framework," Tech. Rep., 2023.
[20]. Z. Ji *et al.*, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, 2023.
[21]. S. Lin *et al.*, "Truthfulqa," *ACL*, 2022.
[22]. P. Manakul *et al.*, "Selfcheckgpt," *EMNLP*, 2023.
[23]. N. Carlini *et al.*, "Prompt injection attacks against llm applications," *arXiv*, 2023.
[24]. P. Christiano *et al.*, "Deep reinforcement learning from human preferences," *NeurIPS*, 2017.
[25]. L. Ouyang *et al.*, "Training lms to follow instructions," *arXiv*, 2022.
[26]. S. Min *et al.*, "Factscore," *ICML*, 2023.
[27]. A. Madaan *et al.*, "Self-refine: Iterative refinement," *NeurIPS*, 2023.
[28]. A. Zou *et al.*, "Universal and transferable adversarial attacks on aligned llms," *arXiv*, 2023.
[29]. D. Ganguli *et al.*, "Red teaming language models," *arXiv*, 2022.
[30]. C. Dwork, "Differential privacy," *ICALP*, 2006.
[31]. M. Abadi *et al.*, "Deep learning with differential privacy," *CCS*, 2016.