Article

Not peer-reviewed version

Building a Security and Reliability Evaluation Suite for Retrieval-Augmented Generation (RAG) Systems

Pronoy Roy * and Debayan Roy *

Posted Date: 8 October 2025

doi: 10.20944/preprints202510.0418.v1

Keywords: etrieval-augmented generation; RAG; large language models; LLMs; domain-aware answers; up-to-date answers; retrieved evidence; security; reliability; trustworthy applications; Secure-RAG; evaluation suite; factual accuracy; hallucination avoidance; adversarial robustness; bias; fairness; toxicity; calibration; query monitoring; retrieval monitoring; generation monitoring; standardized metrics; risk-sensitive settings; continuous evaluation; security-utility tradeoffs

 doi:10.20944/preprints202510.0418.v1

Article

Building a Security and Reliability Evaluation Suite for Retrieval-Augmented Generation (RAG) Systems

Pronoy Roy [1,*] and Debayan Roy [2,*]

[1] Independent Researcher, CA, USA
[2] Independent Researcher, London, UK
* Correspondence: pronoyroy.tech@gmail.com (P.R.); deba301996@gmail.com (D.R.)

Abstract

Retrieval-Augmented Generation (RAG) enables large language models (LLMs) to produce domain-aware, up-to-date answers by conditioning on retrieved evidence. However, the additional retrieval stage introduces new failure modes, hence, evaluating Security and reliability in Retrieval-Augmented Generation (RAG) systems is critical to deploying trustworthy applications. In this paper, we present Secure-RAG, a modular, security-first evaluation suite for multi-dimensional assessment of RAG systems, including factual accuracy, hallucination avoidance, adversarial robustness, bias and fairness, toxicity, security, and calibration. Secure-RAG instruments each stage (query, retrieval, generation) with lightweight monitors that compute standardized metrics. In an illustrative evaluation, we demonstrate Secure-RAG improves reliability without sacrificing utility. Secure-RAG's integrated perspective Security utility tradeoffs that siloed tools often miss, and offers a practical template for continuous evaluation of RAG systems in risk-sensitive settings.

Keywords: etrieval-augmented generation; RAG; large language models; LLMs; domain-aware answers; up-to-date answers; retrieved evidence; security; reliability; trustworthy applications; Secure-RAG; evaluation suite; factual accuracy; hallucination avoidance; adversarial robustness; bias; fairness; toxicity; calibration; query monitoring; retrieval monitoring; generation monitoring; standardized metrics; risk-sensitive settings; continuous evaluation; security-utility tradeoffs

---

The rise of Retrieval-Augmented Generation (RAG) has enabled large language models (LLMs) to generate up-to-date, domain-specific, and contextually grounded answers by retrieving relevant documents at query time. By feeding an LLM with external context (such as knowledge base articles or various domain specific documents), RAG aims to reduce hallucinations and improve factual accuracy. However, integrating a retrieval component also introduces new failure points that can impact the Security and reliability of the system. For example, retrieval might fetch incorrect or irrelevant documents, leading the LLM to produce misleading answers; or the LLM might ignore the retrieved facts and still hallucinate content.

In high-stakes domains (legal, medical, financial), such failures can be disastrous. Consider a financial-advice assistant that answers questions about regulatory disclosures. If retrieval misses the key clause or surfaces an outdated file, the LLM might confidently produce an incorrect summary. Worse, a cleverly crafted prompt could bypass Security rules to obtain restricted content. In such cases, it is not enough to be usually correct; the system must know when to abstain, cite sources, and resist manipulation. Therefore, ensuring Security (preventing harmful, biased, or insecure output) and reliability (consistently correct and faithful responses) in RAG systems has become paramount.

In this paper, we survey the different existing tools for evaluating RAG systems across different dimensions such as factual accuracy, hallucination, adversarial roubstness, bias, toxicity, security, etc. and present Secure-RAG, a comprehensive unified evaluation suite and methodology designed specifically to cover the various metrics for Security and reliability. In particular, Secure-RAG emphasizes a security-first approach (proactively evaluating adversarial and worst-case scenarios) and provenance-aware checks (ensuring answers stay faithful to retrieved sources).

 doi:10.20944/preprints202510.0418.v1

1. Related Work

The unique challenges of evaluating Retrieval-Augmented Generation have led to specialized frameworks. RAGAS (Retrieval Augmented Generation Assessment) [1] formalizes RAG-specific metrics such as faithfulness (answer supported by context), context precision/recall, and answer relevancy, often using LLM-as-judge [2]. It provides reference-free scoring paths and convenient adapters for common frameworks. However, RAGAS primarily targets correctness and relevance; it is less explicit about adversarial robustness or bias/fairness checks.

TruLens [3] popularized the "RAG triad" (context relevance, groundedness, answer relevance), surfacing hallucinations by checking each edge of the pipeline. It integrates as feedback functions, enabling inline evaluation in enterprise stacks. Security metrics beyond content moderation typically require custom hooks.

LangChain Evaluators/LangSmith [4] offer flexible evaluators (reference-based or reference-free) for correctness, hallucination checks, and toxicity tagging, leveraging LLM-graded criteria. This flexibility is useful, but it shifts the responsibility to the practitioner to compose a comprehensive suite and manage evaluator bias.

OpenAI Evals [5] standardizes eval orchestration with customizable Python evals, datasets, and model-graded scoring. It is general and not RAG-specific; provenance checks and retrieval diagnostics require user-defined logic.

DeepEval [6] and adjacent open-source tools add breadth (hallucination, bias, toxicity, and prompt-vulnerability attacks) and pytest-like ergonomics, inching closer to integrated assessment but still requiring assembly for full RAG coverage and provenance checks.

In summary, while a number of tools exist, each has a slightly different focus. RAGAS and TruLens prioritize hallucination and correctness, DeepEval and LangChain add bias/toxicity and adversarial testing, and others like Traceloop [7] emphasize traceability and provenance.

Existing tools skew toward correctness and hallucination; security and calibration are often bolt-ons. Bias/fairness support is inconsistent, and provenance/citation accuracy—critical for trust—is rarely enforced systematically. Secure-RAG integrates these dimensions under one roof as a modular suite and treats adversarial stress-testing and abstention as co-equal objectives, not afterthoughts. In particular, Secure-RAG differentiates itself by explicitly integrating security evaluations (prompt injection, etc.) and calibration checks into the core suite, treating them as first-class citizens alongside accuracy and bias. As far as we are aware, no single existing framework provides a unified, security-first evaluation covering all these dimensions; this is the gap Secure-RAG aims to fill, drawing inspiration and components from the related work described above.

2. Methodology

In this section, we outline the methodology for evaluating each major dimension of Security and reliability in RAG systems. For each aspect, we describe what it entails, why it matters for RAG, and how it can be measured or detected. These form the conceptual foundation of the Secure-RAG evaluation suite.

2.1. Factual Accuracy and Groundedness

Definition Factual accuracy means the model's output is correct with respect to established truth or a trusted reference. In RAG, we specifically care that the answer is grounded in the retrieved documents. Answers must be factually correct and supported by retrieved context and no claims contradicted by or absent from sources.

Why it matters RAG systems are often used for question answering or decision support, where incorrect facts can mislead users. The main utility of RAG is to reduce hallucinations and improve accuracy by leveraging external knowledge. However, if either the retrieval step fails (missing or wrong docs) or the generation step fails (ignoring docs, or misinterpreting them), the answer can

 doi:10.20944/preprints202510.0418.v1

be wrong. Evaluating factual accuracy is crucial to ensure the RAG pipeline actually delivers on its promise of factual correctness.

Measurement

- Ground-truth comparison (when available): Exact Match/F1 for extractive tasks; LLM-graded semantic correctness for abstractive answers.
- Faithfulness: split answer into atomic claims; verify each against retrieved context via LLM-judge or retrieval overlap; aggregate to a 0–1 score.
- Context precision/recall: precision = fraction of used answer content supported by context; recall = fraction of relevant context reflected in the answer.
- Unsupported-claim rate: average number of claims without evidence.

2.2. Hallucination Detection

Definition Hallucinations are outputs that sound plausible but are not supported by (or are outright contradicted by) the source information [8]. In RAG, hallucination often manifests as the model introducing a detail that was not in the retrieved documents.

Why it matters Even if overall accuracy is high, a single hallucinated figure or citation can severely undermine trust. In applications like medical or legal assistants, any hallucinated content can be dangerous. Therefore, detecting hallucinations is very important.

Measurement

- Provenance checks / citation accuracy: map text spans or entities to sources; flag unmatched spans.
- Context-utilization tests: perturb context (e.g., inject distractors) and measure answer sensitivity; low sensitivity suggests overreliance on prior model knowledge.
- LLM-judged hallucination: ask an evaluator to list unsupported statements; compute hallucination rate.

2.3. Adversarial Robustness

Definition Adversarial robustness is the system's ability to withstand malicious or unusual inputs that are designed to break it. For RAG, this includes prompt injection attacks, jailbreak prompts, and context poisoning.

Why it matters Because RAG systems often serve interactive user queries and pull data from potentially large and dynamic corpora, they have a broad attack surface. If the system naively concatenates user query and context, such an injection could cause a serious policy violation. Similarly, if an attacker can influence the documents (imagine a public wiki or forum being used as a source), they might insert phrases that hijack the model.

Measurement

- Prompt-injection suite: canonical patterns ("ignore previous instructions", role overrides, obfuscations) appended or embedded; mark attack success rate if the output deviates from policy or reveals secrets.
- Jailbreak attempts: known role-play and multi-turn inducements; score compliance/refusal.
- Context poisoning tests: seed vector store with high-similarity malicious docs; probe if the system uses or obeys them.
- Adversarial Question Benchmarking: adversarial QA datasets which contain questions intentionally written to confuse models.

2.4. Bias and Fairness

Definition Bias in AI systems refers to systematic and unfair preferences or prejudices in outputs regarding groups of people (or other categories like geographies, etc.). Fairness means the model's behavior and performance are equitable and not skewed unjustifiably.

 doi:10.20944/preprints202510.0418.v1

Why it matters If a RAG system is used for a wide audience (e.g., a customer support chatbot, or a student tutor), it must not produce discriminatory or offensive stereotypes. Bias can erode trust and even cause harm (imagine a legal assistant that consistently gives harsher assessments for individuals of a certain demographic due to biased training data).

Measurement

- Bias Probing Questions: targeted prompts judged for neutrality and fairness.
- Stereotype Confirmation Tests: targeted prompts that contain stereotype and proportion of biased completions.
- Differential performance: accuracy across various subgroups/topic subsets.

2.5. Toxicity and Content Security

Definition Toxicity refers to rude, hateful, violent, or otherwise harmful language. Content Security evaluation checks that the model does not produce such content and adheres to moderation guidelines.

Why it matters A reliable system must not only be correct, but also not cause harm or offense. Even if a user asks something that could lead to a toxic answer (intentionally or not), the system should respond within appropriate bounds. Since RAG can retrieve real-world data, it's possible the source documents contain toxic or biased language (e.g., web data).

Measurement

- Toxic Content Prompts: datasets with real toxicity prompts; automatic moderation scores; toxicity rate above threshold.
- Hate and Harassment Scenarios: craft scenarios where the user might harass the assistant or vice versa
- Disallowed Content Categories: self-harm, illicit instructions, sexual content; compliance/refusal confusion matrix.
- Toxicity in Retrieved Text: include toxic sources; verify paraphrase/filtering rather than reproduction.

2.6. Security Risks and Privacy

Definition This dimension covers any output that could pose security risks – either to the user, the system, or third parties. It includes leakage of confidential information, compliance with privacy laws (e.g., not exposing personal data), and generally the system's resilience against being used for illicit purposes.

Why it matters RAG systems often contain or have access to private data. It's critical that the system doesn't inadvertently reveal sensitive info to unauthorized queries. Also, users might try to use the system to generate malware code or other harmful things.

Measurement

- Privacy/PII Tests: seeded confidential records; verify refusal without authorization.
- Data Leakage via Prompt: explicit and indirect requests; check for leakage.
- Malicious Use Requests: phishing/malware requests; verify refusal.

2.7. Calibration and Abstention

Definition Calibration refers to the alignment between the model's confidence in its answer and the actual correctness likelihood. Since models usually do not output explicit probabilities, we often look at surrogate measures or the model's willingness to abstain when uncertain. Abstention (or refusal) means the model chooses not to answer or says it doesn't know, to avoid giving a potentially incorrect or unSecure answer.

Why it matters Uncalibrated models may either be overconfident (state answers as fact even when guessing, leading users to believe false information) or underconfident (hedging or refusing when they actually do know the answer). For reliability, we want the system to provide an answer

 doi:10.20944/preprints202510.0418.v1

whenever it is likely correct and Secure, and abstain when it would otherwise hallucinate or break a rule. This improves overall trust.

Measurement

- Uncertainty Elicitation: output a confidence level or a token (e.g., "Confidence = X%" "[Sure]/[Not Sure]") or sampling variance between multiple answers
- Selective QA Evaluation: bin predictions by confidence; compare empirical accuracy.
- Abstention behavior in practice: compute accuracy vs coverage as the system abstains below a threshold; track selective accuracy and abstention rate on unanswerable queries.

3. Implementation

In this section, we describe the design and implementation of Secure-RAG, our proposed evaluation suite. Secure-RAG is designed to be modular (so that new metrics or tests can be plugged in), integrated (covering the full RAG pipeline from input to retrieval to output), and automated (to allow routine testing, e.g., in CI/CD or model development cycles).

3.1. Design Overview

The Secure-RAG framework can be visualized as a layered architecture around the RAG system under test. Figure 1 presents a conceptual diagram of the Secure-RAG evaluation pipeline. The core RAG system (retriever and LLM) is instrumented with evaluation probes at various points:

- A Query Monitor intercepts the user's query to detect adversarial patterns (prompt injection attempts, etc.) before it enters the system.
- The Retriever Monitor observes the documents retrieved for the query, evaluating their relevance and adequacy (metrics like recall, precision of retrieval).
- The LLM Output Monitor examines the final answer, evaluating factual correctness (groundedness against retrieved docs), checking for toxicity or bias in language, and assessing compliance with any policies.
- Optionally, an Aggregation/Decision Module can use the monitors' signals to decide if the answer should be delivered or if an abstention/warning is triggered (this is more for deployment, but we include it to evaluate if such a policy improves outcomes).

 doi:10.20944/preprints202510.0418.v1

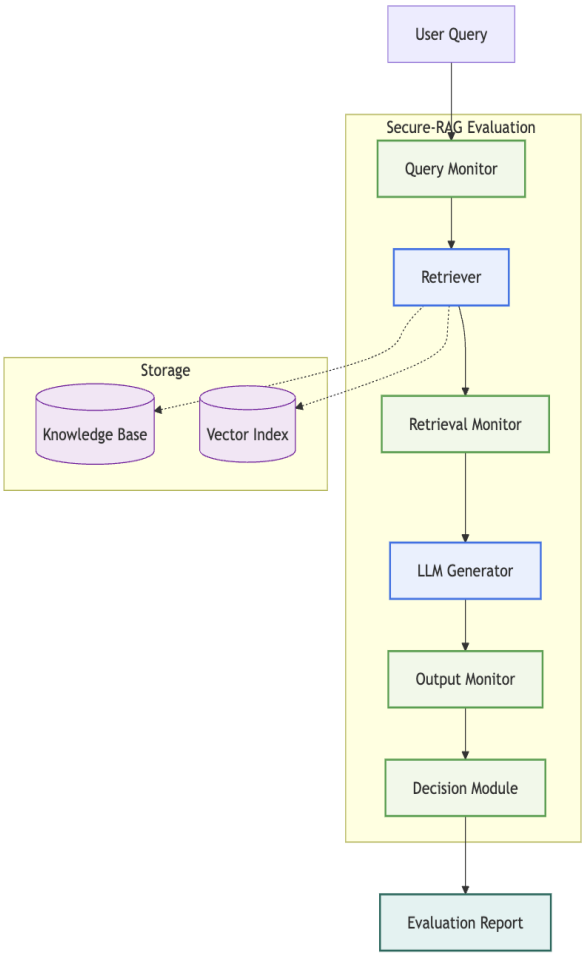

Figure 1. Secure-RAG Architecture – The user query flows through the RAG system (Retriever → LLM), while Secure-RAG attaches monitors at each stage. For example, a Query Monitor scans the query for prompt injections; a Retrieval Monitor computes metrics like recall and checks context relevance; an Output Monitor evaluates the LLM's answer for factuality, toxicity, bias, etc. Based on these, a Decision module can calibrate trust (e.g., trigger an abstention if high risk). This modular design allows evaluating each component (retrieval, generation) and the overall behavior in an integrated manner.

The monitors correspond to the evaluation dimensions discussed earlier. For instance, the Output Monitor actually encompasses multiple evaluators: one for groundedness/factuality, one for toxicity, one for bias, etc. In implementation, these could be separate functions that process the answer and produce scores/flags. The output of Secure-RAG for a single query is a structured report containing all metrics and flags.

The Secure-RAG suite can be run in two modes:

- Offline Evaluation Mode: We use a set of static set of test queries (with expected answers or behaviors when available) to systematically evaluate the model.
- Online Monitoring Mode (Continuous): Secure-RAG monitors live interactions and red flags issues in real-time. This is to ensure deployment and run-time safety, but the tool can be used to log metrics.

For this paper, we focus on the offline evaluation mode.

Secure-RAG is implemented in Python, leveraging several libraries:

- LangChain [9] for constructing the RAG pipeline and using its evaluation helpers.
- LlamaIndex [10] for easily managing documents and indexes, to test retrieval.
- FAISS (Facebook AI Similarity Search) [11] as the vector store for retrieval.
- Pre-trained models or APIs (OpenAI GPT-4) for LLM-as-judge evaluations.

 doi:10.20944/preprints202510.0418.v1

- OpenAI Evals [5] style templates for structuring evals and possibly running them at scale.

3.2. Example

To illustrate a full evaluation pass, let's walk through an example query step by step with Secure-RAG monitors:

Query: "What is the capital of Fantasia?" (Fantasia is fictional, no answer)

- Adversarial Detector: no immediate red flags (it's not disallowed, just possibly unanswerable).
- Retrieval Monitor: likely finds nothing relevant in the knowledge base (low recall).
- LLM Output: if the model is not calibrated, it might hallucinate an answer: "The capital of Fantasia is Fantasia City." This would be unsupported (hallucination). The Answer Checker's groundedness test fails (no doc supports that), factual check fails (Fantasia doesn't exist). Secure-RAG marks this query as answered incorrectly and with a hallucination. If the system was calibrated to abstain, it might say "I'm not sure" – that would be a better outcome (which Secure-RAG would mark as correct behavior because for an unanswerable query, abstention is what we expect). We could have a simple expected output rule here: expected either "no answer" or a refusal. Deviating is penalized.

4. Result

We applied Secure-RAG to several representative RAG configurations over illustrative corpus and query set. Configurations included a permissive generative setup, a conservative generative setup emphasizing abstention, and a simple extractive baseline. Below are qualitative findings across few of the core dimensions.

- Factual Accuracy & Hallucination: On queries supported by context, generative systems produced concise, well-formed answers; the extractive baseline returned grounded spans but limited synthesis. Hallucinations arose mainly on unanswerable or out-of-scope prompts. Introducing confidence-gated abstention reduced unsupported claims by encouraging deferral when evidence was insufficient, at the cost of answering fewer questions.
- Adversarial Robustness: The Query Monitor effectively blocked malicious instructions when injected directly into user queries, showing the value of early-stage filtering. However, adversarial content embedded in retrieved passages still influenced generation, as existing safeguards did not reliably neutralize these hidden directives. This underscores the need for additional layers—retrieval sanitization, instruction-smuggling detection, policy-aware decoding, and output moderation—to strengthen defenses beyond the query stage.
- Bias & Fairness: Given the scope, we observed no clear systematic disparities on simple factual queries. Generative methods can introduce stylistic or inferential bias absent from purely extractive behavior.
- Calibration & Confidence: Eliciting confidence and enforcing abstention thresholds improved alignment between certainty and answer quality, yielding the expected accuracy-coverage trade-off. Separate robustness defenses remain necessary, as abstention alone does not neutralize prompt injection.

Multi-dimensional evaluation surfaced vulnerabilities invisible to accuracy-only checks and showed that calibrated abstention, provenance checks, and adversarial defenses are essential for reliable, secure RAG.

5. Conclusion

Ensuring that Retrieval-Augmented Generation (RAG) systems are both effective and safe is a complex challenge. In this article, we examined the key evaluation dimensions for RAG—including factual accuracy, hallucination detection, adversarial robustness, bias/fairness, toxicity, security, and calibration. While existing tools provide valuable capabilities, none address the full range of concerns.

 doi:10.20944/preprints202510.0418.v1

This gap motivated the design of Secure-RAG, an integrated evaluation suite that unifies these dimensions within a single framework.

Secure-RAG has three salient features. First, a security-first module systematically tests vulnerabilities like prompt injection and context poisoning. Second, an abstention-based calibration strategy balances answer coverage with correctness, allowing systems to defer when confidence is low. Third, By attaching modular monitors to each stage of the pipeline, Secure-RAG reveals precisely where a system fails—whether at retrieval, during adversarial manipulation, or in biased generation.

Secure-RAG underscores that comprehensive evaluation is essential for responsible RAG deployment. By identifying weaknesses in grounding, robustness, or ethical alignment, practitioners can iterate with measurable impact—whether through improved retrieval methods, prompt refinements, or model fine-tuning. The framework offers a step toward standardizing safety evaluation for complex AI systems that blend LLMs with external knowledge.

5.1. Future Work

Looking ahead, several directions remain open. Secure-RAG could be integrated more tightly with training pipelines, using evaluation signals as feedback for fine-tuning or reinforcement learning. Expanding the scope to include user-centric metrics—such as helpfulness, clarity, or user satisfaction—would ensure safety improvements do not come at the expense of utility. As retrieval-augmented techniques become standard in production systems, evaluation suites must evolve to address emerging risks, including novel adversarial strategies. We also plan to open-source Secure-RAG, encouraging community contributions, especially in expanding adversarial test sets as new exploits are discovered.

Ultimately, this work contributes to the broader effort to build trustworthy AI. Rigorous, transparent evaluation of RAG systems increases confidence in deploying them for high-impact applications—from delivering medical answers with verified sources to supporting legal research without hallucinated precedents. Secure-RAG provides both a practical tool and a model for how the community can foster a culture of comprehensive evaluation in AI development.

References

1. Es, S.; James, J.; Espinosa-Anke, L.; Schockaert, S. Ragas: Automated Evaluation of Retrieval Augmented Generation, 2025, [arXiv:cs.CL/2309.15217].
2. Liang, P.; Bommasani, R.; Lee, T.; Tsipras, D.; Soylu, D.; Yasunaga, M.; Zhang, Y.; Narayanan, D.; Wu, Y.; Kumar, A.; et al. Holistic Evaluation of Language Models, 2023, [arXiv:cs.CL/2211.09110].
3. Truelens. https://github.com/truera/trulens/.
4. langsmith. https://www.langchain.com/langsmith.
5. Evals, O. https://github.com/openai/evals.
6. Eval, C.A.D. https://github.com/confident-ai/deepeval.
7. Traceloop. https://www.traceloop.com/.
8. Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems* **2025**, *43*, 1–55. https://doi.org/10.1145/3703155.
9. Langchain. https://github.com/langchain-ai/langchain.
10. LlamaIndex. https://github.com/run-llama/llama_index.
11. Douze, M.; Guzhva, A.; Deng, C.; Johnson, J.; Szilvasy, G.; Mazaré, P.E.; Lomeli, M.; Hosseini, L.; Jégou, H. The Faiss library, 2025, [arXiv:cs.LG/2401.08281].