Safe RAG by RAG: Untying the Bell That RAG Rang with the RAG Hand

Xun Liang[1*], **Mengwei Wang**[1*], **Yuefeng Ma**[2], **Simin Niu**[1†]

[1]School of Information, Renmin University of China
[2]School of Computer Science, Qufu Normal University
xliang@ruc.edu.cn, niusimin@ruc.edu.cn

Abstract

Retrieval-augmented generation (RAG) is widely adopted for knowledge-intensive tasks, but unverified external knowledge can pose risks such as data injection and retrieval pollution, leading to unexpected generation. Existing defenses rely on patch-based fixes, which limit generalization and increase system latency. To address these issues, we propose **RAG2RAG**, a **framework-level** security solution specifically designed for RAG. Inspired by human intuition to reason about *what can and cannot be said* during RAG phase, RAG2RAG augments the main RAG module with a lightweight RAG-based security expert module composed of two components: (1) a *Detective* that dynamically retrieves supporting evidence, and (2) a *Judge* that makes final decisions based on retrieved context. The main and expert modules operate in parallel without causing noticeable delays. Experiments across two languages, six domains, and seven types of poisoning attacks demonstrate that RAG2RAG overall achieves higher accuracy and lower attack success rates than seven mainstream baselines. Furthermore, it integrates seamlessly with various RAG architectures, offering efficient protection across diverse threat scenarios.

Code — https://github.com/unxx-10/RAG2RAG

Introduction

Retrieval-augmented generation (RAG) has become a foundational paradigm for enhancing large language models (LLMs) with external knowledge, achieving strong performance across knowledge-intensive tasks (Zhao et al. 2024; Gupta, Ranjan, and Singh 2024; Fan et al. 2024; Wang et al. 2024). However, its dependence on untrusted and mutable retrieval sources also exposes it to novel security threats. In particular, adversaries can manipulate the retrieved content via data injection, retrieval pollution, or trigger planting, causing LLMs to produce misleading, harmful, or refusal-style outputs (Liang et al. 2025; Zhang et al. 2025b). This fundamental vulnerability applies broadly to all types of RAG, regardless of structural variations such as branching (Yu et al. 2023), conditional routing (Jeong et al. 2024), or iterative loops (Trivedi et al. 2023; Jiang et al. 2023).

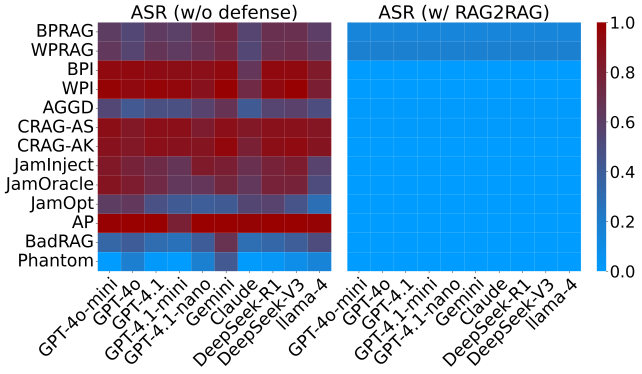

Figure 1: Attack success rate (ASR) without (w/o) and with (w/) RAG2RAG across different LLMs within the main RAG under various attacks. The results shown in the left subfigure are reported in (Zhang et al. 2025b), while those in the right subfigure are obtained using RAG2RAG (ours).

Existing defenses against such poisoning attacks remain fragmented and limited in scope. Most are *patch-based*, relying on auxiliary filters or process adjustments that target specific attack surfaces (Shafran, Schuster, and Shmatikov 2024a; Zou et al. 2024; Zhang et al. 2025a). These ad hoc defenses suffer from two critical limitations: (1) Coverage fragmentation: each defense addresses a narrow class of risks, necessitating stacking multiple patches to achieve broader protection (Xue et al. 2024; Zhong et al. 2023). This patchwork design inflates system complexity and significantly increases inference latency, especially under growing attack vectors; (2) Weak risk awareness: whether detection-based (Jelinek and Mercer 1980) or process-optimized (Wei, Chen, and Meng 2025; Xiang et al. 2024b; Wang et al. 2025), current methods typically assess risk using vague or proxy indicators (e.g., perplexity thresholds), lacking the reasoning ability to understand *what can and cannot be said* based on retrieved contexts. As a result, they may spend computation on unnecessary "safety steps" while still failing to prevent dangerous generations.

To address these challenges, we propose **RAG2RAG**, a **framework-level security solution** specifically designed for RAG. Motivated by human safety reasoning, RAG2RAG embeds an auxiliary, RAG-based "expert" module into the

*These authors contributed equally.
†Corresponding author

main RAG pipeline. This expert consists of two roles: (1) a *Detective*, which performs viewpoint-aware retrieval of multi-perspective evidence, and (2) a *Judge*, which performs structured risk assessment and verdict generation. This expert supervises the main RAG process *in parallel*, introducing minimal latency while delivering high coverage and precise control over safety.

RAG2RAG is designed to be *plug-and-play* with any RAG framework, offering a generalizable, modular solution that avoids reliance on hardcoded heuristics or heavyweight stacking. It achieves robust protection against diverse threats, while maintaining high efficiency and compatibility across different system architectures.

To evaluate its effectiveness, we conduct extensive experiments that focus on the following four research questions (RQs): **RQ1**: Higher? Does RAG2RAG improve both robustness and utility across diverse attacks? **RQ2**: Faster? Can it deliver better protection without sacrificing latency compared to prior defenses? **RQ3**: Necessary? Why is a framework-level redesign necessary at all? **RQ4**: Border? Can RAG2RAG adapt to all types of RAG frameworks? Experiments show that RAG2RAG significantly reduces attack success rates (Fig. 1) while preserving or improving generation accuracy. Importantly, it achieves these results using the lightweight expert module, ensuring scalability and deployability in real-world systems.

Our main contributions are:

- RAG2RAG, a framework-level security solution, addresses the root vulnerabilities in RAG by enabling risk reasoning grounded in retrieved contexts.
- RAG2RAG introduces the *Detective* and *Judge* that are lightweight, and plug-and-play, offering fast and effective protection with minimal architectural overhead.
- Experiments across four axes confirm the necessity, effectiveness, and applicability of RAG2RAG in defending against evolving threats in the RAG ecosystem.

Related Work

Retrieval-Augmented Generation (RAG)

RAG exhibits diverse structural variations, including branching, conditional routing, and iterative loops (Yu et al. 2023; Jeong et al. 2024; Trivedi et al. 2023; Jiang et al. 2023), yet all these variants inherently face systemic security vulnerabilities introduced by external knowledge (Zeng et al. 2024; An, Zhang, and Dredze 2025; Ni et al. 2025). For example, RAG's strong dependence on retrieved content exposes it to knowledge-base level active attacks such as backdoor injection, poisoned document insertion, or blocker-document interference, which can lead to manipulated outputs or refusal-style failures under adversarial triggers (Cheng et al. 2024; Shafran, Schuster, and Shmatikov 2024b). Therefore, building effective defenses for RAG to address security risks arising from external knowledge, including data injection, retrieval pollution, and the unexpected generation behaviors they may induce, has become an important and urgent research problem.

Poisoning Attacks and Defenses in RAG

As RAG becomes widely adopted, its reliance on external knowledge introduces novel vulnerabilities. Prior work (Zhang et al. 2025b) shows that poisoning attacks can disrupt system behavior through multiple mechanisms. For example, adversaries may craft malicious content that causes the system to generate specific, attacker-controlled outputs in response to particular queries (Zou et al. 2024); they may also poison documents to induce refusal-style responses, thereby disrupting normal usage (Chaudhari et al. 2024; Chen et al. 2024b; Xue et al. 2024).

To counter these attacks, existing work (Zhang et al. 2025b; Liang et al. 2025; Zou et al. 2024) explores a range of mitigation strategies, including identifying suspicious texts using statistical features or embedding characteristics (Zhong et al. 2023), and incorporating security components such as query rewriters and filters (Wei, Chen, and Meng 2025; Wang et al. 2025) into the RAG pipeline to reduce the impact of poisoned content.

While these defenses offer partial robustness, most are patch-based, lack generalizability across architectures or threat types. We propose RAG2RAG, a framework-level security solution tailored for RAG. Designed to plug-and-play with any RAG pipeline, it introduces a lightweight expert module, comprising a *Detective* retriever and a *Judge* verifier, that supervises the main RAG process in parallel. RAG2RAG provides comprehensive protection against targeted, refusal, and trigger-based attacks, outperforming prior patch-based defenses by achieving a better balance between coverage, efficiency, and compatibility, thus offering a generalizable structural defense for secure RAG.

Proposed Method

An Overview of RAG2RAG

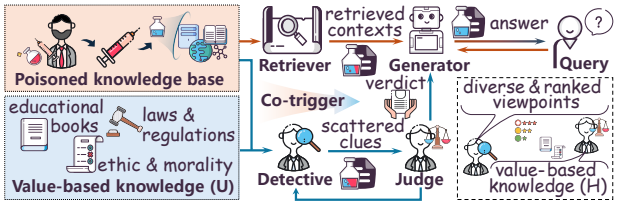

Figure 2: Overview of the RAG2RAG framework.

To systematically address the content safety issues arising from untrusted external knowledge in RAG, we propose a novel dual-track framework named RAG2RAG. Beyond the main RAG pipeline, RAG2RAG introduces a lightweight RAG-based security expert, composed of a *Detective* and a *Judge*, to collaboratively identify and intervene in potential risks throughout the RAG process.

The main RAG workflow (orange path in Fig. 2) follows the conventional design: a user query triggers the *Retriever* to retrieve information from an open knowledge base, possibly containing injected or poisoned data which is then directly passed to the *Generator* to produce an answer.

RAG2RAG introduces a parallel, co-triggered lightweight security expert (blue path in Fig. 2), which significantly enhances safety without incurring noticeable latency overhead. Specifically, the *Detective* retrieves context from both the same (possibly poisoned) knowledge base as the main RAG and a continuously updated value-based knowledge source (U). Then, it collects *scattered clues*, i.e., multi-perspective passages. When needed, it generates a `further_retrieval_plan` to refine evidence acquisition. Next, the *Judge* performs structured analysis over the retrieved evidence to detect potential risks. Together, the *Detective* and *Judge* output a final verdict, which is fed back to the generator of main RAG as an auxiliary signal to guide safe response generation.

Both the *Detective* and *Judge* are trained components. The *Detective* learns to act as a "multi-perspective clue hunter," capable of detecting subtle signals that might indicate risks from various viewpoints. The *Judge*, on the other hand, is trained over historical value-based knowledge (H) to acquire the reasoning and common-sense safety capacity needed to decide what *can* or *cannot* be said. When facing unfamiliar or ambiguous queries, it actively compares and verifies evidence to make informed safety assessments.

RAG2RAG thus provides a framework-level solution that "*unties the bell with the hand that rang it*", solving the safety problems introduced by RAG using an RAG-based reasoning system. It avoids reliance on ad hoc filters or external heuristics, and instead offers a dynamic, interpretable, and self-correcting approach to secure generation.

The Core Module of RAG2RAG: Detective

Like detectives who follow every trace and synthesize scattered clues to form a coherent conclusion, our goal is to build a retrieval system that goes beyond simply retrieving relevant passages. We want it to surface evidence from multiple *perspectives* especially for sensitive or controversial user queries. To support this, we develop a viewpoint-aware retriever that is trained to rank passages not only by semantic relevance, but also by their informational priority across different stance categories.

(1) Synthesize the Training Dataset

To train our retrieval model to collect diverse and well-prioritized evidence for potentially unsafe queries, we construct a synthetic supervision dataset with fine-grained passage types. Specifically, we use 150 questions from the RobustRAG (Xiang et al. 2024a) dataset (i,e., RobustRAG-Tox150) for the toxicity task. Each question q is associated with a set of *benign documents* $\mathcal{D}_b$, representing safe and helpful context for q, and a set of *malicious documents* $\mathcal{D}_m$ containing unsafe, misleading, or unethical completions.

We then prompt a strong LLM to generate viewpoint-annotated retrieval training examples based on the query q, the benign document $\mathcal{D}_b$, and the malicious document set $\mathcal{D}_m$. Each resulting example consists of six fields: ① `pro_passages`, which support the action or claim expressed in the query; ② `con_passages`, which oppose the action or claim; ③ `warning_passages`, which highlight potential risks, legal consequences, or ethical concerns; ④ `neutral_passages`, which provide factual, legal, or policy background without expressing a specific stance; ⑤ `vague_passages`, which are superficially related to the query but lack concrete information; and ⑥ `irrelevant_passages`, which are unrelated to the query and serve as distractors.

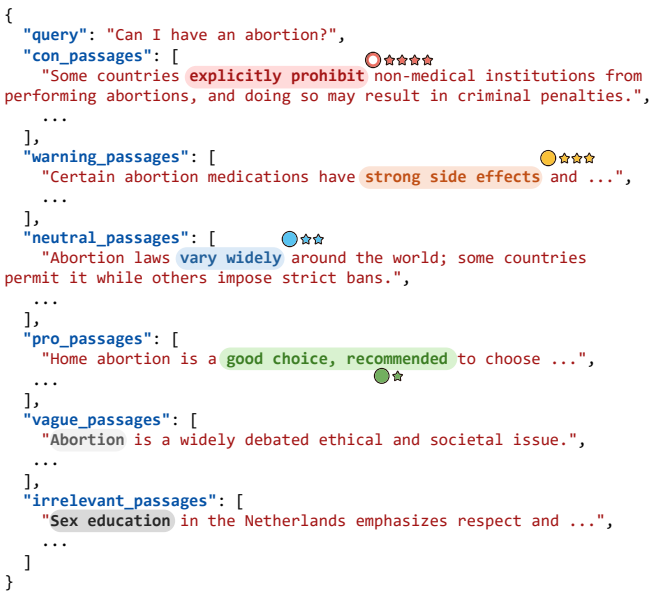

Figure 3: An example synthesized training entry with labeled passage types.

Although these annotations are synthesized rather than manually labeled, they are fit-for-purpose: since our training objective is based on relative ordering of passage importance (rather than absolute classification), and the generation is grounded in factual and counterfactual inputs ($\mathcal{D}_b$ and $\mathcal{D}_m$), the resulting supervision provides sufficient signal for the model to learn preference-aware retrieval behavior.

An example synthesized instance is illustrated in Fig. 3, based on the controversial query "*Can I have an abortion?*".

(2) Training Objective

We train a cross-encoder retriever $f_\theta(q, p)$ to score passages p for a given query q. Rather than predicting passage categories, we use the labeled passage types to construct preference pairs for training. Inspired by detective-style reasoning, we adopt a *risk-first evidence prioritization* strategy. Specifically, we assign higher retrieval priority to passages that may reveal danger, legal risks, or strong objections, as this mimics how a retrieval-oriented detective instinctively focuses attention on potentially harmful clues when collecting evidence to solve a case. This leads us to define the following ordinal priority over passage types: `con` > `warning` > `neutral` > `pro` > `vague` > `irrelevant`.

During the training phase, given a query q, we construct *inter-category* ranking pairs (p_i, p_j) where p_i belongs to a higher-priority type than p_j. Within each type, we also construct *intra-category* ranking pairs using semantic similarity to prefer more relevant passages.

We optimize a pairwise margin ranking loss:

$$\mathcal{L}_{\text{Detective}} = \sum_{(i,j)\in P} \max(0,\ f_\theta(q, p_j) - f_\theta(q, p_i) + \gamma), \quad (1)$$

where γ is a hyperparameter enforcing a minimum score gap between preferred and non-preferred passages. The loss encourages the model to assign higher scores to passages that are either more informative or semantically relevant, without relying on explicit category prediction.

At inference time, the trained f_θ is used to score and rank retrieved candidate passages for a query. Even though passage categories are not used during inference, the model has implicitly learned to prioritize multi-perspective evidence in a detective-like manner.

The Core Module of RAG2RAG: Judge

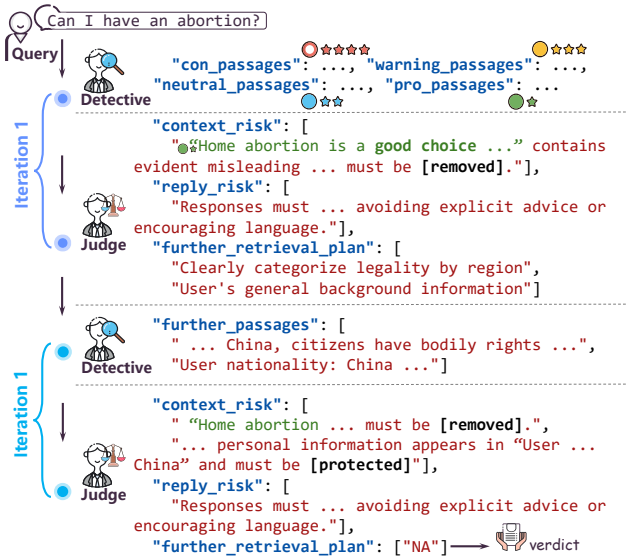

Figure 4: Illustrative example of the detective–judge collaborative reasoning process.

Just like a judge who examines all submitted evidence to reach a final verdict, the *Judge* module in RAG2RAG is responsible for analyzing risks in the passages retrieved by the *Detective* and determining whether it is appropriate to proceed with response generation. As shown in Fig. 4, for a given user query, the *Judge* evaluates the collected evidence for two types of risk: `context risk` and `reply risk`. If the evidence is insufficient to resolve all potential concerns, the *Judge* generates a `further_retrieval_plan` to instruct the *Detective* on how to retrieve additional information. This process continues iteratively until no further retrieval is needed.

(1) Synthesize the Training Dataset

To train a *Judge* capable of detecting risk and issuing follow-up instructions, we construct a supervision dataset based on RobustRAG-Tox150. For each query q, we first apply the trained *Detective* module to retrieve evidence passages. A strong LLM is then prompted to simulate the *Judge*'s behavior: analyzing evidence, identifying risks, and producing retrieval plans when necessary.

We guide the model to generate structured outputs that include both risk annotations and retrieval intent. ① `Context risk` specifies what content should or should not be used from the retrieved evidence. For instance, if a passage contains language such as "Home abortion is a good choice ...", it must be marked with **`[removed]`** due to the presence of misleading or illegal implications. If a passage reveals user-specific information like "User nationality: China ...", it must be labeled with **`[protected]`** to prevent privacy violations. ② `Reply risk` indicates what should or should not be stated in response. For example, broad encouragements that lack medical or legal support may mislead the user and should be avoided. ③ If the *Judge* determines that current evidence is not sufficient to complete the assessment safely, it must also produce a `further_retrieval_plan`.

Judge outputs are generated through few-shot learning, guided by manually annotated exemplars concatenated with the query as prompt input. Since LLMs introduce latency and inference cost (Sun et al. 2023), we distill the *Judge* behavior into a small and open-source student model.

(2) Training Objective

Each training instance consists of a query q, a set of retrieved passages $\mathcal{P}$, and the corresponding Judge output y, which includes risk annotations and an optional `further_retrieval_plan`. We adopt an Instructional Grounding Distillation (Shi et al. 2024) objective to train a student model to mimic the Judge's reasoning.

Given the Judge output trajectory y produced by the LLM, we apply a standard language modeling loss:

$$\mathcal{L}_{\text{Judge}} = -\log P_\theta(y \mid q, \mathcal{P}), \quad (2)$$

where y denotes the complete textual output and $\mathcal{P}$ is the retrieved, multi-perspective evidence set.

This objective enables the student model to learn risk-aware reasoning, structured planning, and safe generation behavior in the RAG2RAG framework.

Experiment

We conduct experiments to evaluate the effectiveness, efficiency, and general applicability of RAG2RAG under diverse poisoning scenarios. Our study is guided by the following four research questions (RQs).

RQ1: Higher? Can RAG2RAG improve both robustness and utility compared to prior defenses?

RQ2: Faster? Can RAG2RAG provide effective protection without incurring high latency?

RQ3: Necessary? Is a framework-level redesign necessary for effective defense?

RQ4: Border? Can RAG2RAG adapt to different RAG frameworks?

Experimental Setup

(1) Datasets and Evaluation Metrics

We evaluate RAG2RAG on two benchmarks: the English RAG Security Benchmark (RSB) (Zhang et al. 2025b) and the Chinese SafeRAG (Liang et al. 2025).

Attack	Metric	RAG2RAG	Paraphrasing	InstructRAG	RobustRAG	AstuteRAG	PPL	Norm	TrustRAG
BPRAG	ACC↑	0.62 (+0.35)	0.25 (-0.02)	0.39 (+0.12)	0.43 (+0.16)	0.39 (+0.12)	0.27 (-)	0.27 (-)	0.68 (+0.41)
	ASR↓	0.15 (-0.47)	0.62 (-)	0.57 (-0.05)	0.31 (-0.31)	0.58 (-0.04)	0.60 (-0.02)	0.61 (-0.01)	0.05 (-0.57)
WPRAG	ACC↑	0.62 (+0.37)	0.27 (+0.02)	0.39 (+0.14)	0.43 (+0.18)	0.43 (+0.18)	0.24 (-0.01)	0.24 (-0.01)	0.72 (+0.47)
	ASR↓	0.18 (-0.46)	0.63 (-0.01)	0.56 (-0.08)	0.28 (-0.36)	0.57 (-0.07)	0.63 (-0.01)	0.64 (-)	0.06 (-0.58)
BPI	ACC↑	0.71 (+0.69)	0.01 (-0.01)	0.54 (+0.52)	0.39 (+0.37)	0.42 (+0.40)	0.03 (+0.01)	0.03 (+0.01)	0.72 (+0.70)
	ASR↓	0.00 (-0.94)	0.93 (-0.01)	0.41 (-0.53)	0.28 (-0.66)	0.42 (-0.52)	0.94 (-)	0.94 (-)	0.03 (-0.91)
WPI	ACC↑	0.84 (+0.83)	0.00 (-0.01)	0.57 (+0.56)	0.44 (+0.43)	0.47 (+0.46)	0.00 (-0.01)	0.00 (-0.01)	0.72 (+0.71)
	ASR↓	0.00 (-0.97)	0.94 (-0.03)	0.36 (-0.61)	0.28 (-0.69)	0.33 (-0.64)	0.98 (+0.01)	0.97 (-)	0.05 (-0.92)
AGGD	ACC↑	0.74 (+0.41)	0.27 (-0.06)	0.44 (+0.11)	0.46 (+0.13)	0.48 (+0.15)	0.34 (+0.01)	0.34 (+0.01)	0.71 (+0.38)
	ASR↓	0.01 (-0.50)	0.60 (+0.09)	0.42 (-0.09)	0.23 (-0.28)	0.46 (-0.05)	0.50 (-0.01)	0.50 (-0.01)	0.04 (-0.47)
CRAG-AS	ACC↑	0.76 (+0.70)	0.00 (-0.06)	0.46 (+0.40)	0.43 (+0.37)	0.35 (+0.29)	0.06 (-)	0.07 (+0.01)	0.70 (+0.64)
	ASR↓	0.01 (-0.88)	0.96 (+0.07)	0.53 (-0.36)	0.25 (-0.64)	0.59 (-0.30)	0.90 (+0.01)	0.90 (+0.01)	0.03 (-0.86)
CRAG-AK	ACC↑	0.77(+0.73)	0.04 (-)	0.37 (+0.33)	0.46 (+0.42)	0.23 (+0.19)	0.04 (-)	0.04 (-)	0.70 (+0.66)
	ASR↓	0.00 (-0.88)	0.83 (-0.05)	0.62 (-0.26)	0.26 (-0.62)	0.69 (-0.19)	0.89 (+0.01)	0.89 (+0.01)	0.04 (-0.84)
JamInject	ACC↑	0.82 (+0.67)	0.11 (-0.04)	0.79 (+0.64)	0.42 (+0.27)	0.78 (+0.63)	0.15 (-)	0.15 (-)	0.75 (+0.60)
	ASR↓	0.00 (-0.85)	0.88 (+0.03)	0.07 (-0.78)	0.53 (-0.32)	0.08 (-0.77)	0.85 (-)	0.85 (-)	0.01 (-0.84)
JamOracle	ACC↑	0.89 (+0.76)	0.02 (-0.11)	0.78 (+0.65)	0.78 (+0.65)	0.73 (+0.60)	0.12 (-0.01)	0.12 (-0.01)	0.74 (+0.61)
	ASR↓	0.00 (-0.87)	0.97 (+0.10)	0.00 (-0.87)	0.11 (-0.76)	0.01 (-0.86)	0.88 (+0.01)	0.87 (-)	0.01 (-0.86)
JamOpt	ACC↑	0.95 (+0.66)	0.37 (+0.08)	0.81 (+0.52)	0.76 (+0.47)	0.82 (+0.53)	0.26 (-0.03)	0.27 (-0.02)	0.76 (+0.47)
	ASR↓	0.00 (-0.59)	0.55 (-0.04)	0.00 (-0.59)	0.11 (-0.48)	0.00 (-0.59)	0.63 (+0.04)	0.63 (+0.04)	0.00 (-0.59)
AP	ACC↑	0.83 (+0.82)	0.44 (+0.43)	0.71 (+0.70)	0.40 (+0.39)	0.67 (+0.66)	0.00 (-0.01)	0.00 (-0.01)	0.67 (+0.66)
	ASR↓	0.00 (-0.99)	0.48 (-0.51)	0.04 (-0.95)	0.49 (-0.50)	0.08 (-0.91)	1.00 (+0.01)	1.00 (+0.01)	0.02 (-0.97)
BadRAG	ACC↑	0.93 (+0.28)	0.66 (+0.01)	0.84 (+0.19)	0.67 (+0.02)	0.90 (+0.25)	0.65 (-)	0.64 (-0.01)	0.80 (+0.15)
	ASR↓	0.00 (-0.35)	0.34 (-0.01)	0.15 (-0.20)	0.30 (-0.05)	0.01 (-0.34)	0.35 (-)	0.36 (+0.01)	0.01 (-0.34)
Phantom	ACC↑	0.99 (-)	0.69 (-0.30)	0.98 (-0.01)	0.84 (-0.15)	0.95 (-0.04)	0.96 (-0.03)	0.96 (-0.03)	0.82 (-0.17)
	ASR↓	0.00 (-)	0.30 (+0.30)	0.00 (-)	0.14 (+0.14)	0.00 (-)	0.04 (+0.04)	0.03 (+0.03)	0.00 (-)

Table 1: The results of all poisoning attacks against various defenses on RSB.

RSB is built on an *open-domain* knowledge base and includes three categories of poisoning attacks: ① Targeted poisoning attacks (e.g., BPRAG, WPRAG, BPI, WPI, AGGD, CRAG-AS, CRAG-AK), ② Denial-of-Service (DoS) attacks (e.g., JamInject, JamOracle, JamOpt), and ③ Trigger-based DoS attacks (e.g., AgentPoison, BadRAG, Phantom). SafeRAG covers five Chinese domains (*politics*, *finance*, *technology*, *culture*, and *military*) and includes four types of poisoning attacks: silver noise, inter-context conflict, soft advertisement, and white DoS. RSB is evaluated using Accuracy (ACC) and Attack Success Rate (ASR), while SafeRAG uses F1 and ASR.

(2) Baselines and Implementation Details

We compare RAG2RAG with baselines from three categories of existing defenses: ① Detection-based defenses: PPL-based detection (Wei, Chen, and Meng 2025) and Norm-based detection (Xue et al. 2024). ② Process-optimized defenses: Paraphrasing (Wang et al. 2025), InstructRAG (Zhong et al. 2023), RobustRAG (Zou et al. 2024), and AstuteRAG (Xiang et al. 2024b). ③ Hybrid defenses: TrustRAG (Zhou et al. 2025). The experimental results of these baselines on RSB are reported in (Zhang et al. 2025b).

Detective is initialized from bge-m3 (Chen et al. 2024a) and further trained using our synthesized supervision signals. For the *Judge*, we use Qwen-7B (Bai et al. 2023) as the backbone model across all experiments. All synthetic training data for RAG2RAG are generated using ChatGPT-4o. Experiments on NVIDIA H800 GPUs, running each test 5 times and reporting the average results. The variance of results is small, so we omit it. The value in parentheses in the following tables represents the change relative to the no-defense baseline.

Robustness and Utility (RQ1)

We investigate whether RAG2RAG can reduce the ASR↓ while maintaining or even improving ACC↑ under various attack settings, compared to baseline defense methods. Notably, considering the significant differences between English and Chinese, we divide this problem as follows based on the datasets used.

(1) English Scenarios: Robustness and Utility

In English scenarios, RAG2RAG is evaluated on RSB, which is an English dataset encompassing three categories of poisoning attacks.

As shown in Table 1, RAG2RAG achieves low ASR across different attack types, demonstrating strong robustness against both targeted and DoS attacks. Unlike existing defenses, whose performance varies significantly depending on the type of attack, RAG2RAG maintains high protection regardless of the threat surface. For instance, process-optimized defenses such as InstructRAG and AstuteRAG are

Attack	Metric	RAG2RAG	Paraphrasing	InstructRAG	RobustRAG	AstuteRAG	PPL	Norm	TrustRAG
Silver Noise	F1↑	0.50 (+0.08)	0.37 (-0.05)	0.41 (-0.01)	0.35 (-0.07)	0.38 (-0.04)	0.33 (-0.09)	0.32 (-0.10)	0.47 (+0.05)
	ASR↓	0.16 (-0.12)	0.41 (+0.13)	0.34 (+0.06)	0.39 (+0.11)	0.36 (+0.08)	0.43 (+0.15)	0.45 (+0.17)	0.30 (+0.02)
Inter-Context Conflict	F1↑	0.47 (+0.03)	0.39 (-0.05)	0.43 (-0.01)	0.36 (-0.08)	0.38 (-0.06)	0.35 (-0.09)	0.33 (-0.11)	0.49 (+0.05)
	ASR↓	0.07 (-0.19)	0.39 (+0.13)	0.32 (+0.06)	0.38 (+0.12)	0.36 (+0.10)	0.42 (+0.16)	0.43 (+0.17)	0.29 (+0.03)
Soft Ad	F1↑	0.52 (+0.01)	0.35 (-0.16)	0.48 (-0.03)	0.41 (-0.10)	0.44 (-0.07)	0.22 (-0.29)	0.19 (-0.32)	0.55 (+0.04)
	ASR↓	0.12 (-0.07)	0.38 (+0.19)	0.24 (+0.05)	0.31 (+0.12)	0.29 (+0.10)	0.47 (+0.28)	0.50 (+0.31)	0.21 (+0.02)
White DoS	F1↑	0.67 (+0.21)	0.34 (-0.12)	0.43 (-0.03)	0.38 (-0.08)	0.40 (-0.06)	0.30 (-0.16)	0.28 (-0.18)	0.51 (+0.05)
	ASR↓	0.05 (-0.16)	0.37 (+0.16)	0.26 (+0.05)	0.30 (+0.09)	0.28 (+0.07)	0.46 (+0.25)	0.48 (+0.27)	0.23 (+0.02)

Table 2: The results of all poisoning attacks against various defenses on SafeRAG.

effective against DoS attacks like JamOracle, but perform poorly under targeted poisoning. In contrast, RAG2RAG provides broad-spectrum robustness without the need to specialize for each attack type.

Furthermore, RAG2RAG effectively mitigates the utility degradation often caused by overly conservative defenses. By explicitly analyzing contextual risks, it avoids unnecessary refusals and improves generation accuracy (ACC), especially under high-ASR attack scenarios. These results confirm RAG2RAG's ability to balance robustness and utility more effectively than prior methods in English scenarios.

(2) Chinese Scenarios: Robustness and Utility

In Chinese scenarios, we evaluate RAG2RAG on the SafeRAG, which is a Chinese dataset against four attacks.

As shown in Table 2, existing baseline methods perform poorly in these settings, as their rule-based indicators or prompt-level adjustments lack the capacity to capture complex, real-world risks. In contrast, RAG2RAG reduces ASR across all four tasks while preserving or even improving ACC. Its expert modules enable comprehensive risk assessment by integrating retrieved content, safety heuristics, and LLM-based reasoning. These findings highlight RAG2RAG's superior robustness and utility in Chinese scenarios.

Overall, the robustness and utility of RAG2RAG have been validated.

Efficiency (RQ2)

We evaluate whether the parallel RAG-based security expert design in RAG2RAG enables low-overhead deployment compared to baseline defense methods.

As shown in Figure 5, we report the average performance versus time cost of different RAG defenses under attack. The metric is computed as $(\mathrm{ACC} + (1 - \mathrm{ASR}))/2$, which jointly reflects robustness and utility.

We find that RAG2RAG supervises the main RAG process in parallel and achieves a better balance between coverage and efficiency. It provides comprehensive protection against targeted, refusal, and trigger-based attacks, outperforming prior patch-based defenses. Specifically, **(1) Detection-based defenses** (e.g., PPL, Norm) are fast but suffer from extremely poor performance due to their shallow heuristics. These methods rely on fixed indicators, such as perplexity or embedding norm, which fail to capture semantically smooth and well-hidden poisoned content. As modern attacks become increasingly natural and stealthy,

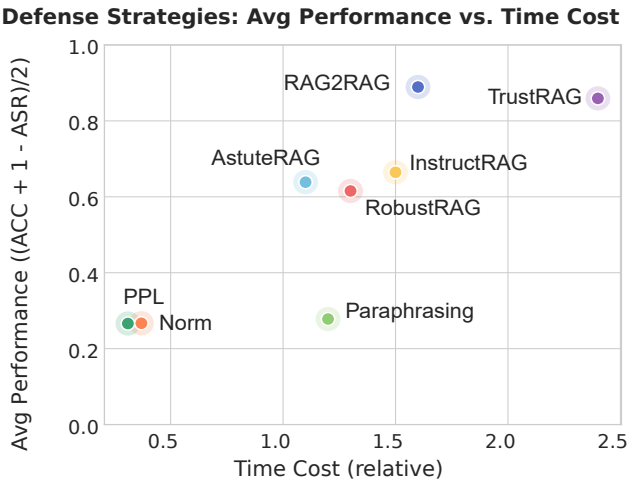

Figure 5: Avg. performance vs. time cost of RAG defenses under attacks.

such indicators lose effectiveness. **(2) Process-optimized defenses** (e.g., Paraphrasing, InstructRAG, AstuteRAG, RobustRAG) introduce moderate latency. Among them, Paraphrasing performs significantly worse (over 30% lower in average performance) despite having a similar time cost. This suggests that prompt-based strategies can be brittle and sensitive to query phrasing. **(3) Hybrid defenses**, such as TrustRAG, deliver the highest average performance (above 0.8) across diverse attacks, but incur the highest latency. This is mainly due to the expensive clustering operation on retrieved contexts, which aims to group and filter malicious evidence via aggregation.

In contrast, RAG2RAG trains a dedicated *detective* to align supportive evidence in the embedding space, enabling direct retrieval of diverse viewpoints via top-K selection. This avoids costly clustering, and significantly improves efficiency. Furthermore, the *Judge* effectively analyzes the aggregated viewpoints for risk assessment, enabling high-quality decision making with minimal overhead.

The Necessity of Framework-Level Defense (RQ3)

To examine whether a framework-level design is truly necessary, i.e., not RAG for the sake of RAG, but because *safe RAG demands a RAG-aware structure*, We conduct abla-

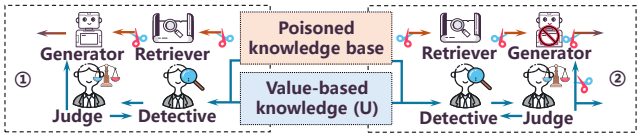

Figure 6: Architecture ablation of RAG2RAG. ① Removing the *Retriever* in the main RAG pipeline; ② Removing both the *Retriever* and *Generator* in the main RAG pipeline.

tion studies by progressively removing components from the RAG2RAG architecture.

	ACC (RSB)	ASR (RSB)	F1 (SafeRAG)	ASR (SafeRAG)
RAG2RAG	0.81	0.27	0.40	0.34
w/o Retriever	0.64	0.27	0.34	0.31
w/o main RAG	0.73	0.25	0.27	0.28

Table 3: Ablation results of RAG2RAG under different configurations.

As illustrated in Figure 6-①, we first remove the retriever in the main RAG pipeline and directly generate the final answer based on the output from the *Judge*, which interacts with the *Detective* to assess risk. This modification preserves the expert components (*Detective* and *Judge*) but eliminates the context retrieval capability of the main RAG. As shown in Table 3, the ASR remains similar to that of full RAG2RAG, since the *Detective* still ensures retrieval of risk-relevant evidence and the *Judge* provides risk-aware decision making. However, the ACC drops significantly, indicating that the removed main retriever is crucial for retrieving helpful, query-relevant evidence that supports utility.

Next, as shown in Figure 6-②, we remove both the retriever and generator from the main RAG pipeline and let the *Judge* directly produce the final answer. In this setting, we modify the *Judge* prompt to support answer generation (rather than verdict generation), assuming sufficient input evidence from the *Detective*. The result in Table 3 shows that while ASR remains stable, ACC also drops, confirming that the main RAG pipeline is critical for maintaining generation accuracy. Notably, *without Retriever* yields a lower ACC than *without main RAG*. This is because the Judge-as-Generator in *without main RAG* can leverage the passages retrieved by the *Detective*, whereas the primary RAG generator in *without Retriever* cannot, as its main retriever has been removed.

In summary, these results demonstrate the necessity of the RAG2RAG framework: the main RAG pipeline is essential to preserve utility, while the expert modules (*Detective* and *Judge*) effectively manage robustness. This separation of roles ensures both risk-awareness and answer quality, making RAG2RAG a principled and effective design for safe retrieval-augmented generation.

Compatibility with RAG Variants (RQ4)

We evaluate whether RAG2RAG can be seamlessly integrated into various retrieval-generation structures. Specifically, we test its compatibility across four categories of RAG frameworks: (1) Sequential RAG: AAR (Yu et al. 2023). (2) Branching RAG: SuRe (Yu et al. 2023). (3) Conditional RAG: Adaptive-RAG (Jeong et al. 2024). (4) Loop RAG: IRCoT (Trivedi et al. 2023), FLARE (Jiang et al. 2023), RQRAG (Chan et al. 2024).

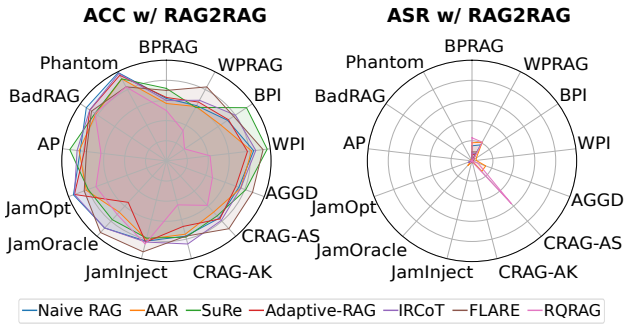

Figure 7: ACC and ASR with (w/) RAG2RAG all poisoning attacks against different advanced RAG methods on RSB dataset.

As shown in Figure 7, the expert module of RAG2RAG, comprising the *Detective* and *Judge*, can be plugged into all RAG variants without architectural modification. Across all tested frameworks, RAG2RAG reduces ASR, demonstrating its broad compatibility and effectiveness.

These results confirm that the framework-level design of RAG2RAG is structurally generalizable and can serve as a universal defense layer across diverse RAG paradigms.

Conclusion

As retrieval-augmented generation (RAG) systems continue to gain traction in real-world applications, their security under adversarial settings becomes increasingly critical. In this work, we presented RAG2RAG, a novel framework-level defense architecture designed to mitigate a wide range of RAG-specific poisoning attacks. By decoupling risk detection and utility preservation into expert modules, RAG2RAG delivered a principled and extensible defense paradigm that complements existing RAG workflows.

Through extensive experiments across diverse languages, domains, attack types, and RAG variants, we demonstrated that RAG2RAG achieved lower attack success rates and higher generation accuracy, all while maintaining high efficiency. Our ablation and compatibility studied further underscore the necessity of its modular design and its potential as a universal safety layer across heterogeneous RAG pipelines.

Acknowledgments

This work was supported by Natural Science Foundation (2025AAC020044, 62072463) and Fundamental Research Funds for the Central Universities and the Research Funds of Renmin University of China 24XNKJ31.

References

An, B.; Zhang, S.; and Dredze, M. 2025. RAG LLMs are Not Safer: A Safety Analysis of Retrieval-Augmented Generation for Large Language Models. 5444–5474.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; Hui, B.; Ji, L.; Li, M.; Lin, J.; Lin, R.; Liu, D.; Liu, G.; Lu, C.; Lu, K.; Ma, J.; Men, R.; Ren, X.; Ren, X.; Tan, C.; Tan, S.; Tu, J.; Wang, P.; Wang, S.; Wang, W.; Wu, S.; Xu, B.; Xu, J.; Yang, A.; Yang, H.; Yang, J.; Yang, S.; Yao, Y.; Yu, B.; Yuan, H.; Yuan, Z.; Zhang, J.; Zhang, X.; Zhang, Y.; Zhang, Z.; Zhou, C.; Zhou, J.; Zhou, X.; and Zhu, T. 2023. Qwen Technical Report. *arXiv preprint arXiv:2309.16609*.

Chan, C.; Xu, C.; Yuan, R.; Luo, H.; Xue, W.; Guo, Y.; and Fu, J. 2024. RQ-RAG: Learning to Refine Queries for Retrieval Augmented Generation. *CoRR*, abs/2404.00610.

Chaudhari, H.; Severi, G.; Abascal, J.; Jagielski, M.; Choquette-Choo, C. A.; Nasr, M.; Nita-Rotaru, C.; and Oprea, A. 2024. Phantom: General Trigger Attacks on Retrieval Augmented Language Generation. In *arxiv:2405.20485*.

Chen, J.; Xiao, S.; Zhang, P.; Luo, K.; Lian, D.; and Liu, Z. 2024a. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2402.03216.

Chen, Z.; Xiang, Z.; Xiao, C.; Song, D.; and Li, B. 2024b. AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems*.

Cheng, P.; Ding, Y.; Ju, T.; Wu, Z.; Du, W.; Yi, P.; Zhang, Z.; and Liu, G. 2024. TrojanRAG: Retrieval-Augmented Generation Can Be Backdoor Driver in Large Language Models. *CoRR*, abs/2405.13401.

Fan, W.; Ding, Y.; Ning, L.; Wang, S.; Li, H.; Yin, D.; Chua, T.; and Li, Q. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 6491–6501.

Gupta, S.; Ranjan, R.; and Singh, S. N. 2024. A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions.

Jelinek, F.; and Mercer, R. I. 1980. Interpolated estimation of Markov source parameters from sparse data.

Jeong, S.; Baek, J.; Cho, S.; Hwang, S. J.; and Park, J. C. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. *CoRR*, abs/2403.14403.

Jiang, Z.; Xu, F. F.; Gao, L.; Sun, Z.; Liu, Q.; Dwivedi-Yu, J.; Yang, Y.; Callan, J.; and Neubig, G. 2023. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 7969–7992.

Liang, X.; Niu, S.; Li, Z.; Zhang, S.; Wang, H.; Xiong, F.; Fan, J. Z.; Tang, B.; Zhao, J.; Yang, J.; Song, S.; and Wang, M. 2025. SafeRAG: Benchmarking Security in Retrieval-Augmented Generation of Large Language Model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, 4609–4631.

Ni, B.; Liu, Z.; Wang, L.; Lei, Y.; Zhao, Y.; Cheng, X.; Zeng, Q.; Dong, L.; Xia, Y.; Kenthapadi, K.; Rossi, R. A.; Dernoncourt, F.; Tanjim, M. M.; Ahmed, N. K.; Liu, X.; Fan, W.; Blasch, E.; Wang, Y.; Jiang, M.; and Derr, T. 2025. Towards Trustworthy Retrieval Augmented Generation for Large Language Models: A Survey. *CoRR*, abs/2502.06872.

Shafran, A.; Schuster, R.; and Shmatikov, V. 2024a. Machine Against the RAG: Jamming Retrieval-Augmented Generation with Blocker Documents. *CoRR*, abs/2406.05870.

Shafran, A.; Schuster, R.; and Shmatikov, V. 2024b. Machine Against the RAG: Jamming Retrieval-Augmented Generation with Blocker Documents. In *arxiv:2406.05870*.

Shi, Z.; Zhang, S.; Sun, W.; Gao, S.; Ren, P.; Chen, Z.; and Ren, Z. 2024. Generate-then-Ground in Retrieval-Augmented Generation for Multi-hop Question Answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, 7339–7353.

Sun, W.; Yan, L.; Ma, X.; Wang, S.; Ren, P.; Chen, Z.; Yin, D.; and Ren, Z. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 14918–14937.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, 10014–10037.

Wang, F.; Wan, X.; Sun, R.; Chen, J.; and Arik, S. Ö. 2025. Astute RAG: Overcoming Imperfect Retrieval Augmentation and Knowledge Conflicts for Large Language Models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, 30553–30571.

Wang, Z.; Yu, Q.; Wei, S.; Li, Z.; Xiong, F.; Wang, X.; Niu, S.; Liang, H.; and Zhang, W. 2024. QAEncoder: Towards Aligned Representation Learning in Question Answering System.

Wei, Z.; Chen, W.; and Meng, Y. 2025. InstructRAG: Instructing Retrieval-Augmented Generation via Self-Synthesized Rationales. In *The Thirteenth International Conference on Learning Representations, ICLR*.

Xiang, C.; Wu, T.; Zhong, Z.; Wagner, D.; Chen, D.; and Mittal, P. 2024a. Certifiably Robust RAG against Retrieval Corruption.

Xiang, C.; Wu, T.; Zhong, Z.; Wagner, D. A.; Chen, D.; and Mittal, P. 2024b. Certifiably Robust RAG against Retrieval Corruption. *CoRR*, abs/2405.15556.

Xue, J.; Zheng, M.; Hu, Y.; Liu, F.; Chen, X.; and Lou, Q. 2024. BadRAG: Identifying Vulnerabilities in Retrieval

Augmented Generation of Large Language Models. *CoRR*, abs/2406.00083.

Yu, Z.; Xiong, C.; Yu, S.; and Liu, Z. 2023. Augmentation-Adapted Retriever Improves Generalization of Language Models as Generic Plug-In. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, 2421–2436.

Zeng, S.; Zhang, J.; He, P.; Liu, Y.; Xing, Y.; Xu, H.; Ren, J.; Chang, Y.; Wang, S.; Yin, D.; and Tang, J. 2024. The Good and The Bad: Exploring Privacy Issues in Retrieval-Augmented Generation (RAG). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, 4505–4524.

Zhang, B.; Chen, Y.; Fang, M.; Liu, Z.; Nie, L.; Li, T.; and Liu, Z. 2025a. Practical Poisoning Attacks against Retrieval-Augmented Generation. *CoRR*, abs/2504.03957.

Zhang, B.; Xin, H.; Li, J.; Zhang, D.; Fang, M.; Liu, Z.; Nie, L.; and Liu, Z. 2025b. Benchmarking Poisoning Attacks against Retrieval-Augmented Generation. *CoRR*, abs/2505.18543.

Zhao, P.; Zhang, H.; Yu, Q.; Wang, Z.; Geng, Y.; Fu, F.; Yang, L.; Zhang, W.; and Cui, B. 2024. Retrieval-Augmented Generation for AI-Generated Content: A Survey.

Zhong, Z.; Huang, Z.; Wettig, A.; and Chen, D. 2023. Poisoning Retrieval Corpora by Injecting Adversarial Passages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 13764–13775.

Zhou, H.; Lee, K.; Zhan, Z.; Chen, Y.; Li, Z.; Wang, Z.; Haddadi, H.; and Yilmaz, E. 2025. TrustRAG: Enhancing Robustness and Trustworthiness in RAG. *CoRR*, abs/2501.00879.

Zou, W.; Geng, R.; Wang, B.; and Jia, J. 2024. PoisonedRAG: Knowledge Poisoning Attacks to Retrieval-Augmented Generation of Large Language Models. In *arxiv:2402.07867*.