THROUGH THE STEALTH LENS: RETHINKING ATTACKS AND DEFENSES IN RAG

Sarthak Choudhary[1] **Nils Palumbo**[1] **Ashish Hooda**[1]
Krishnamurthy Dj Dvijotham[2] **Somesh Jha**[1]

[1]University of Wisconsin-Madison [2]ServiceNow Research

June 12, 2025

ABSTRACT

Retrieval-augmented generation (RAG) systems are vulnerable to attacks that inject poisoned passages into the retrieved set, even at low corruption rates. We show that existing attacks are not designed to be stealthy, allowing reliable detection and mitigation. We formalize stealth using a distinguishability-based security game. If a few poisoned passages are designed to control the response, they must differentiate themselves from benign ones, inherently compromising stealth. This motivates the need for attackers to rigorously analyze intermediate signals involved in generation—such as attention patterns or next-token probability distributions—to avoid easily detectable traces of manipulation. Leveraging attention patterns, we propose a passage-level score—the **Normalized Passage Attention Score**—used by our **Attention-Variance Filter** algorithm to identify and filter potentially poisoned passages. This method mitigates existing attacks, improving accuracy by up to $\sim \mathbf{20}\%$ over baseline defenses. To probe the limits of attention-based defenses, we craft stealthier adaptive attacks that obscure such traces, achieving up to $\mathbf{35}\%$ attack success rate, and highlight the challenges in improving stealth[1].

1 Introduction

Large Language Models (LLMs) have revolutionized various applications with their remarkable generative abilities. However, their reliance on internal knowledge can lead to inaccuracies due to outdated information or hallucinations [1, 2, 3]. Retrieval-Augmented Generation (RAG) [4, 5] has emerged as a leading technique to address these limitations by integrating LLMs with external (non-parametric) knowledge retrieved from databases [6, 7]. It retrieves a set of relevant passages from a knowledge database, denoted as *retrieved set*, and incorporates them into the model's input.

[1]Our implementation is available at `https://github.com/sarthak-choudhary/Stealthy_Attacks_Against_RAG`.

This powerful approach underpins critical real-world systems, including Google Search with AI overviews [8], WikiChat [9], Bing Search [10], Perplexity AI [11], and LLM agents [12, 13, 14, 15].

The reliance of RAG systems on the retrieved set, however, introduces a significant new security vulnerability: the knowledge database becomes an additional attack surface. Malicious actors can inject harmful content—for example, by manipulating Wikipedia pages, spreading fake news on social media, or hosting malicious websites— to corrupt the information retrieved by the RAG system [16]. Consequently, the retrieval of malicious passages by a RAG system, followed by their incorporation into response generation, constitutes a *retrieval corruption attack* [17]. Recent instances, such as the PoisonedRAG attack [18], demonstrate easily exploitable vulnerabilities: the attacker simply prompts GPT-4 to create the malicious context and inject it into the retrieved set, successfully manipulating the answer by corrupting only a small fraction of the retrieved set (e.g., one or two out of ten) [19, 18, 17].

Although existing attacks on RAG systems often achieve high success with low corruption rates, they are typically not designed with stealth in mind, leaving them susceptible to detection and mitigation. Ideally, a robust aggregation mechanism within the RAG system can identify inconsistencies between the model's output and the dominant (benign) signal in the retrieved set. A significant divergence suggests undue influence from a small, potentially malicious subset of passages. Crucially, to override the benign context, adversarial passages must disproportionately influence the model's response, which may necessitate detectable differences from benign passages—leaving behind a malicious trace. The presence of such malicious traces becomes more likely when the adversary cannot compromise the majority of the retrieved set—a particularly challenging task when retrieval is performed over large, diverse corpora like Google Search or Wikipedia [17, 18, 19], or when the retriever is designed to be robust. However, existing attacks largely overlook stealth, relying on weak signals such as perplexity [20, 21, 22]. This raises a fundamental question: *Are existing attacks truly stealthy? If not, can they be detected and mitigated, and how can we develop more sophisticated strategies to enhance their stealth?* We challenge the notion of effortless stealth in current attacks and formally define it through a distinguishability security game. We introduce the **Normalized Passage Attention Score**, a metric that aggregates the attention weights assigned to tokens in each passage from the model's response. Using this metric, we demonstrate that existing low-effort attacks leave detectable traces, as adversarial passages attract disproportionately high attention, typically due to phrases containing or strongly implying the attacker's target answer.

Leveraging the skewed distribution of normalized passage attention scores across the retrieved set, we propose the **Attention-Variance Filter (AV Filter)**—an outlier filtering algorithm that removes passages corresponding to extreme outliers in normalized passage attention scores (See Figure 1 for an overview). The AV Filter effectively distinguishes malicious passages from benign ones, enabling robust defenses by filtering out potentially malicious passages. To rigorously explore the limits of this defense, we extend jailbreak methodologies to create adaptive attacks that optimize for obscuring attention-based traces and evading the AV Filter, marking progress toward stealthier attacks. Our findings highlight the ongoing arms race between attacks and robust RAG systems by formalizing a security game, demonstrating effective mitigation of existing low-stealth attacks, and revealing the challenges of improving stealth through adaptive attacks. We summarize our contributions as follows:

- We formalize stealth in RAG attacks via a distinguishability-based security game.

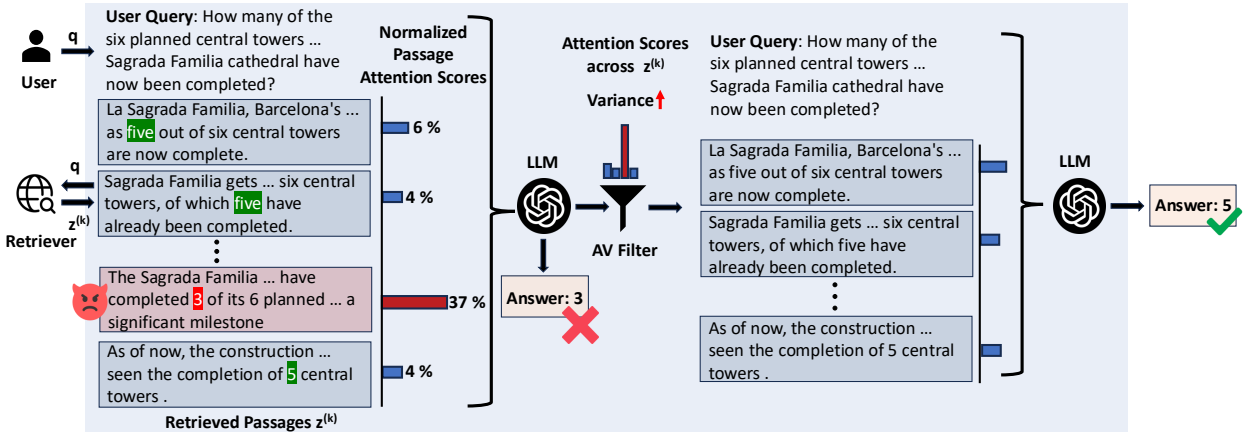

Figure 1: **AV Filter Overview.** In this example, the retriever returns a set of passages $z^{(k)}$, one of which is poisoned and disproportionately influences the response. This leads to a skewed distribution of normalized passage attention scores and elevated variance across passages. AV Filter mitigates this by removing passages with anomalously high attention scores, indicative of potential poisoning.

- We introduce the **Normalized Passage Attention Score** to quantify passage-response dependencies and show that its distribution becomes skewed under corruption.
- Leveraging this score, we design a successful defender for the security game and propose the **Attention-Variance Filter** to identify and remove potentially malicious passages.
- We design stealthier adaptive attacks by leveraging optimization techniques for jailbreaking to evade the Attention-Variance Filter, highlighting the trade-offs in improving stealth.

2 Background and Related Work

Notations and Definitions.

1. Let $\mathcal{Q}$ denote the space of queries and $\mathcal{S}$ the space of possible responses. A query is denoted by $q \in \mathcal{Q}$ and a valid response by $s \in \mathcal{S}$. We also define a target response $s' \in \mathcal{S}$ chosen by an adversary such that $s \neq s'$.
2. Let $\mathcal{Z}$ denote the space of knowledge databases. A knowledge database $z \in \mathcal{Z}$ is a collection of passages $z = \{z_1, z_2, \ldots, z_n\}$, where each z_i is a passage.
3. Let $z^{(k)} \subset z$ represent a subset of k retrieved passages from z, and $\mathcal{Z}^{(k)}$ denote the space of all such k-sized subsets.
4. Let Θ be the space of RAG architectures, where each architecture $\theta \in \Theta$ is defined as $\theta = (\text{Ret}_\theta, \text{Gen}_\theta, \text{LLM}_\theta)$ consisting of a retriever, a generator, and a language model.

Retrieval-Augmented Generation. A RAG pipeline comprises four key components: a knowledge database, a retrieval function, a generation function, and an LLM. The knowledge database consists of a collection of passages sourced from diverse repositories such as Google Search or Wikipedia.

Step I – Knowledge Retrieval: The retriever selects the top-k passages relevant to q, based on semantic similarity between embeddings or web search rankings.

Step II-Generation: The generation function utilizes the retrieved set and the LLM, often with an instructional prompt $\mathcal{I}$, to generate the final response.

A Retrieval-Augmented Generation (RAG) system can be formally defined as the function $f_{\text{RAG}} : \mathcal{Q} \times \mathcal{Z} \times \Theta \to \mathcal{S}$, where $f_{\text{RAG}}(q, z, \theta) = \text{Gen}_\theta(q, \text{Ret}_\theta(q, z)) = s$, with:

- $\text{Ret}_\theta : \mathcal{Q} \times \mathcal{Z} \to \mathcal{Z}^{(k)}$ retrieves the top k relevant passages (***Step I***).
- $\text{Gen}_\theta : \mathcal{Q} \times \mathcal{Z}^{(k)} \to \mathcal{S}$ generates the final response s (***Step II***).

In a standard RAG pipeline [5], the retriever assigns relevance scores to passages independently and selects the top-k passages based on these scores. Specifically, the retriever's output is:

$$\text{Ret}_\theta(q, z) = z^{(k)} = \{z_{i_1}, z_{i_2}, \ldots, z_{i_k}\}$$

Next, the generation function processes a concatenated sequence consisting of the instructional prompt, the retrieved passages, and the query, to produce a response. This is formulated as:

$$\text{Gen}_\theta\left(q, z^{(k)}\right) = \text{LLM}_\theta\left(\text{Concat}(\mathcal{I}, z^{(k)}, q)\right) = \text{LLM}_\theta(\mathcal{I} \oplus z_{i_1} \oplus z_{i_2} \oplus \cdots \oplus z_{i_k} \oplus q),$$

where $\oplus$ denotes the concatenation of text sequences. While alternative designs exist, such as reranking [23] or multi-hop retrieval [24], we focus on this pipeline due to its widespread adoption.

Vulnerabilities in RAG Systems. An adversary targeting a specific response s' can craft a set of adversarial passages $z^{(\text{adv})}$ by simultaneously maximizing the following two objectives:

$$\Pr_\theta\left[z^{(\text{adv})} \subset \text{Ret}_\theta\left(q, z \cup z^{(\text{adv})}\right)\right] \quad \text{and} \quad \Pr_\theta\left[\text{LLM}_\theta\left(\text{Concat}(\mathcal{I}, z^{(k)}, q)\right) = s' \;\middle|\; z^{(\text{adv})} \subset z^{(k)}\right]$$

These represent attacks on *Step I* (retrieval) and *Step II* (generation), respectively. **This work focuses on the robustness of Generation stage (*Step II*)**. We argue that a stronger notion of stealth can enhance robustness, enabling the generation step to tolerate limited corruption—a critical goal given the unresolved challenge of building perfectly robust retrievers [25, 26, 27].

Existing Works. The vulnerability of Question Answering (QA) models to disinformation attacks has been a growing concern [28, 29, 30, 31], with recent threats specifically targeting Retrieval-Augmented Generation (RAG) systems. **PoisonedRAG (Poison)** [18] introduces adversarial contexts containing incorrect answers into the knowledge database to mislead generation. **Prompt Injection Attack (PIA)** [19] embed direct instructions in retrieved passages to generate incorrect responses. GARAG [32] exploits subtle malicious typos within the knowledge database

to elicit inaccurate responses. To defend against these threats, approaches such as query paraphrasing [33], misinformation detection [34], and perplexity-based filters [20, 21, 22] have been proposed. However, these often suffer from limited efficacy or high false positives [18]. Recently, Certified Robust RAG [17] was introduced, leveraging an isolate-then-aggregate strategy to provide empirical accuracy bounds and limit attack success. It is currently state-of-the-art in mitigating such attacks. Further details about its methodology are provided in Appendix A.

3 Stealth Analysis of Adversarial Attacks in RAG Systems

Threat Model. We consider an adversary $\mathcal{A}_\epsilon$ with full knowledge of the RAG architecture θ and the knowledge base z. The adversary is allowed to inject, but not modify or remove, up to $\lfloor \epsilon \cdot k \rfloor$ poisoned passages into z. Given a query q, target s', and architecture θ, the adversary generates $z^{(\text{adv})} = \mathcal{A}_\epsilon(q, z, s', \theta) = \left\{ z_1^{(\text{adv})}, \ldots, z_{\lfloor \epsilon \cdot k \rfloor}^{(\text{adv})} \right\}$, such that all of them are retrieved and generate s'.

Let $z_{\text{benign}}^{(k)} = \text{Ret}_\theta(q, z)$ and $z_{\text{corrupt}}^{(k)} = \text{Ret}_\theta\left(q, z \cup z^{(\text{adv})}\right)$. The retrieved set may change by at most $\epsilon \cdot k$ passages, i.e., $\left| z_{\text{benign}}^{(k)} \triangle z_{\text{corrupt}}^{(k)} \right| \leq 2\epsilon \cdot k$, where $\triangle$ denotes symmetric difference. The attack is successful if the RAG system generates the target response, i.e., $\text{Gen}_\theta\left(q, z_{\text{corrupt}}^{(k)}\right) = s'$.

Attack Practicality. When over half of the retrieved passages are poisoned ($\epsilon \geq 0.5$), generating an accurate response becomes inherently difficult. We focus on the more practical setting where $\epsilon < 0.5$, reflecting real-world scenarios where attackers can corrupt only a few top-k results—such as with web search retrievers—or where errors arise unintentionally due to retrieval noise.

Stealth Attack Distinguishability Game (SADG). We define a security game between an arbiter, an adversary $\mathcal{A}_\epsilon$, and a defender $\mathcal{D}$, parameterized by a parameter ϵ. The goal is to evaluate whether $\mathcal{D}$ can distinguish a corrupted retrieved set from a benign one. The corruption budget of $\mathcal{A}_\epsilon$ is controlled by ϵ; smaller values correspond to tighter budgets, making stealth harder.

For a given RAG architecture θ and knowledge database z, the defender does not have access to z, the game proceeds as follows:

1. **Query sampling:** The arbiter samples a query $q \leftarrow \mathcal{Q}$.
2. **Retrieved set generation:** The arbiter samples a target response $s' \leftarrow \mathcal{S}$. It computes the benign retrieved set $z_{\text{benign}}^{(k)} = \text{Ret}_\theta(q, z)$, queries the adversary to obtain poisoned passages $z^{(\text{adv})} = \mathcal{A}_\epsilon(q, z, s', \theta)$, and constructs the corrupted retrieved set $z_{\text{corrupt}}^{(k)} = \text{Ret}_\theta\left(q, z \cup z^{(\text{adv})}\right)$
3. **Permutation:** The arbiter samples a bit $b \leftarrow \{0, 1\}$ uniformly at random and defines:

$$\left(z_0^{(k)}, z_1^{(k)}\right) := \begin{cases} \left(z_{\text{corrupt}}^{(k)}, z_{\text{benign}}^{(k)}\right), & \text{if } b = 0, \\ \left(z_{\text{benign}}^{(k)}, z_{\text{corrupt}}^{(k)}\right), & \text{if } b = 1. \end{cases}$$

   The arbiter sends $\left(q, z_0^{(k)}, z_1^{(k)}\right)$ to the defender $\mathcal{D}$.

4. **Defender's guess:** The defender outputs $b' \in \{0, 1\}$, guessing which of $z_0^{(k)}$ or $z_1^{(k)}$ is the corrupted set. The defender wins if $b' = b$.

Advantage. The defender's advantage is: $\mathsf{Adv}_{\text{SADG}}^{\mathcal{A}_\epsilon, \mathcal{D}}(\theta, z, \epsilon) := \left| \Pr[b' = b] - \frac{1}{2} \right|$.

The probability $\Pr[b' = b]$ is over the randomness of q, s', θ, b, and defender $\mathcal{D}$.

The attack is said to be τ-*stealthy* if, for all probabilistic polynomial-time (PPT) defenders $\mathcal{D}$, the advantage is at most τ; i.e.,

$$\mathsf{Adv}_{\text{SADG}}^{\mathcal{A}_\epsilon, \mathcal{D}}(\theta, z, \epsilon) \leq \tau,$$

for a perfectly stealthy attack τ should be zero.

Stealth of Existing Attacks. Existing attacks, crafted without access to the clean retrieved set, do not explicitly minimize the defender's advantage in the SADG security game. While such attacks may evade detection methods that analyze passages independently—like perplexity-based filters—their influence remains evident in the model's output, making the generated response itself a valuable signal for detecting corruption. This motivates a shift in perspective: **to analyze retrieved passages in conjunction with the generated response and assess whether any passage disproportionately shapes the output.** If most retrieved passages are expected to be relevant to the query, a strong alignment between the response and only a few passages may indicate adversarial manipulation. We formalize this insight using the **Normalized Passage Attention Score**, which quantifies the alignment between each passage and the generated response. These scores are used for two purposes:

1. We propose a defender $\mathcal{D}_{\text{AV}}$ that **distinguishes between benign and corrupted retrieved sets**, achieving a significant advantage in SADG against existing attacks—demonstrating their lack of stealth.

2. We introduce the **Attention-Variance Filter**, an outlier filtering algorithm that **removes potentially poisoned passages** from the retrieved set, effectively mitigating existing attacks.

We evaluate the stealth of existing attacks using SADG. As these attacks are designed to ensure retrieval of poisoned passages, we follow prior work and inject them directly into the retrieved set.

4 Stealth Detection and Mitigation via Attention Variance

We build on the insight that, in a successful attack, the generated response is strongly correlated with the malicious passages that shaped it. Ideally, for a retrieved set $z^{(k)}$ and target response s', we should consider the conditional probability $\Pr_{z^{(k)}}\left(\text{Gen}_\theta(q, z^{(k)}) = s' \middle| z_i \in z^{(k)}\right)$ for each passage z_i in a retrieved set to measure its correlation with the response.

In a transformer-based language model, the attention weights computed during generation are widely used to capture the dependencies between tokens [35]. Therefore, we choose to analyze the attention matrix from the LLM in RAG systems to quantify this correlation. Analyzing the attention matrix of the LLM in the RAG systems has proven to be useful even beyond security considerations, for instance in optimizing KV caches during inference for RAG systems [36, 37].

H_2O [36] observes that only a small portion of input tokens, termed *Heavy Hitters* (H_2), significantly contribute to the attention weights when generating a new token. Their findings confirm that the emergence of H_2 is natural and strongly correlated with the co-occurrence of tokens. Consistent with this, our analysis of compromised RAG systems reveals that when an incorrect response is generated due to malicious influence, these Heavy Hitters are indeed localized within the poisoned passages.

We experimentally observe that in successful attacks, high-attention tokens—Heavy Hitters—are often target-response keywords embedded within poisoned passages. These tokens, due to their co-occurrence with the generated output, receive disproportionately high attention, skewing the overall attention distribution. Based on this insight, we define the **Normalized Passage Attention Score**, which aggregates token-level attention to quantify the proportion of total attention each passage receives from the final response. This score helps identify anomalous passages indicative of adversarial influence. This skewed distribution of attention in poisoned passages is illustrated through examples in Appendix B.

Normalized Passage Attention Score. Let the input to LLM_θ be $\mathcal{X} = \text{Concat}(\mathcal{I}, z^{(k)}, q)$ where $z^{(k)}$ is the retrieved set and q is the query. The model generates a response $s' = \{s'_1, s'_2, \ldots, s'_l\}$ of l tokens. During this process, the model computes multi-layer, multi-head attention weights, with each layer producing a separate attention tensor per head. We average these weights across all decoder layers and heads to construct a unified attention matrix: $A = \mathsf{Attention}(\text{LLM}_\theta, \mathcal{X}) \in \mathbb{R}^{l \times T}$ where T is the number of input tokens. Each entry $A[i, j]$ denotes the mean attention from the i-th output tokens to the j-th input token. This averaging yields a stable view of token-level interactions [38].

Each retrieved passage z_t is a finite sequence of tokens, $z_t = \{z_t^{(1)}, z_t^{(2)}, \ldots\}$. The **Passage Attention Score**, $\mathsf{Score}_\alpha(z_t, A)$, is defined as the total attention from all response tokens s' to the top-α most attended tokens in z_t, denoted as $\text{Top}_\alpha(z_t)$. This focuses on high-signal Heavy Hitter tokens—often adversarial keywords—within a passage thereby amplifying adversarial cues and reducing noise. We define the **Normalized Passage Attention Score**, $\mathsf{NormScore}_\alpha(z_t, z^{(k)}, A)$, by dividing each passage's score by the total score across all k retrieved passages. While normalization preserves ranking, it standardizes attention magnitudes across queries and models, enabling a stable threshold for detecting adversarial passages—unlike instance-specific approaches [39]. For clarity, we rescale it to a percentage and refer to it simply as a passage's attention score.

$$\mathsf{Score}_\alpha(z_t, A) = \sum_{i=1}^{l} \sum_{x_j \in \text{Top}_\alpha(z_t)} A[i, j] \qquad \mathsf{NormScore}_\alpha(z_t, z^{(k)}, A) = \frac{\mathsf{Score}_\alpha(z_t, A)}{\sum_{i=1}^{k} \mathsf{Score}_\alpha(z_i, A)}$$

We compute the attention score of a passage by summing the top-α most attended tokens within it. For any fixed α, this score remains invariant to the passage length, preventing adversaries from gaining an advantage through length manipulation. Ideally, α should match the number of Heavy Hitters—tokens in the poisoned passage that align with the target response—which is often proportional to the number of tokens in the target response. We select sufficiently large values of α to ensure coverage of all Heavy Hitters, using $\alpha \in \{5, 10, \infty\}$, where ∞ denotes summing over all tokens in the passage.

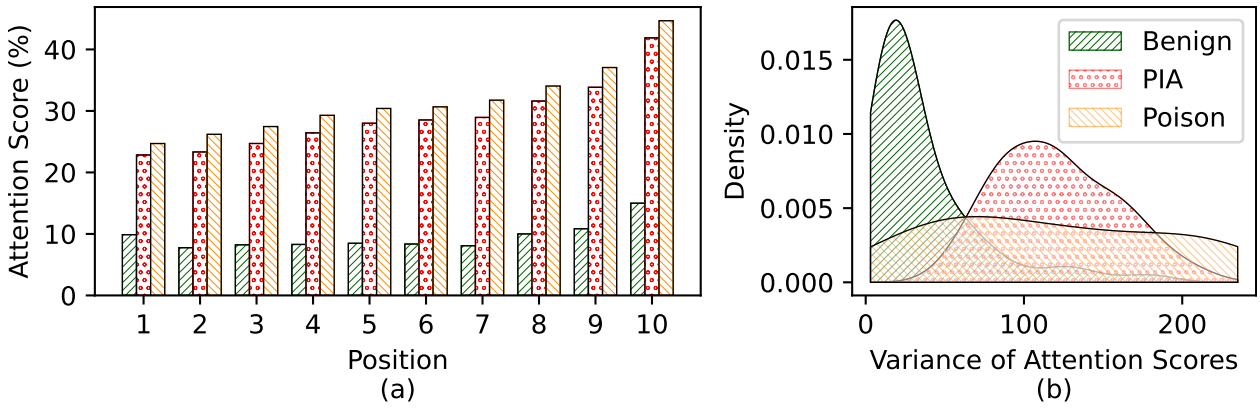

Figure 2: (a) Average attention scores across passage positions in retrieved sets over multiple queries, showing that benign passages have nearly equal attention scores, while a poisoned passage at any position significantly increases its attention score compared to the rest of the benign passages. (b) Distribution of variance of attention scores in benign vs. corrupted retrieved sets, highlighting how the presence of a poisoned passage shifts the variance distribution, making it separable from the benign one. Scores are computed on the RQA dataset [40] with Llama 2 [41] for $\alpha = \infty$ and $\epsilon = 0.1$.

Algorithm 1: **A**ttention-**V**ariance **Filter** (AV Filter)

Input: Query q, Retrieved set $z^{(k)}$, model LLM_θ, Corruption fraction ϵ, Variance threshold δ
Output: Filtered set z_{filtered}

$z^{\text{sorted}} = \mathsf{Sort}(z^{(k)}, \text{LLM}_\theta)$ ▷ Sort passages according to attention scores
$z^{\text{filtered}} \leftarrow z^{\text{sorted}}$ ▷ Initialize the set of filtered passages
while $|z^{\textit{filtered}}| > \lfloor (1-\epsilon) \cdot k \rfloor$ **do**
  $\mathcal{X} \leftarrow \mathsf{Concat}(\mathcal{I}, z^{\text{filtered}}, q)$ ▷ Form the input sequence
  $A \leftarrow \mathsf{Attention}(\text{LLM}_\theta, \mathcal{X})$ ▷ Compute the attention matrix from LLM_θ on $\mathcal{X}$
  $\mathsf{attn\_scores} \leftarrow \{\mathsf{NormScore}_\alpha(z_t, z^{\text{filtered}}, A) | z_t \in z^{\text{filtered}}\}$ ▷ Compute attention scores
  $\sigma^2 = \mathsf{Var}(\mathsf{attn\_scores})$ ▷ Compute the variance of attention scores
  if $\sigma^2 \leq \delta$ **then**
    break
  $z_{\max} = \underset{z_t \in z^{\text{filtered}}}{\text{argmax}} \ \mathsf{NormScore}_\alpha(z_t, z^{\text{filtered}}, A)$
  $z^{\text{filtered}} \leftarrow z^{\text{filtered}} \setminus z_{\max}$ ▷ Remove the passage with the highest score from the filtered set
return $z^{\textit{filtered}}$

Discriminating Between Corrupted and Benign Retrievals via Attention. In benign RAG instances—where retrieved passages are relevant to both query and response—attention distribution over passages is approximately uniform with slight *recency effect* [42, 43]. Corruption skews this attention pattern—Figure 2(a) shows that corrupting a single index significantly elevates its attention scores relative to the benign baseline. This implies that corrupted retrieved sets exhibit a high variance of attention scores across passages. Motivated by this, we propose a defender $\mathcal{D}_{\text{AV}}$ in the SADG game that detects corruption using attention variance. Given a query q and two

retrieved sets $\left(z_0^{(k)}, z_1^{(k)}\right)$, the defender computes attention scores for each passage: $\mathsf{attn\_scores}_i = \left\{\mathsf{NormScore}_\alpha(z_t, z_i^{(k)}, A) | z_t \in z_i^{(k)}\right\}$, and calculates their variance $\mathsf{Var}(\mathsf{attn\_scores}_i)$. The defender then outputs:

$$\mathcal{D}_{\text{AV}}(q, z_0^{(k)}, z_1^{(k)}) := \begin{cases} 0, & \text{if } \mathsf{Var}(\mathsf{attn\_scores}_0) > \mathsf{Var}(\mathsf{attn\_scores}_1), \\ 1, & \text{otherwise.} \end{cases}$$

The defender flags the set with a higher attention score variance as corrupted. Figure 2(b) shows this variance is consistently higher for corrupted sets, enabling reliable detection across attacks.

Filtering Poisoned Passages from Corrupted Retrievals. We propose the **Attention-Variance Filter (AV Filter)**, an outlier filtering algorithm that removes potentially corrupted passages exhibiting unusually high attention. Given a query q, retrieved set $z^{(k)}$, model LLM_θ, corruption budget ϵ and threshold δ, the filter computes the variance of normalized attention scores and iteratively removes the top-scoring passage until the variance drops below δ or ϵ fraction of passages are removed.

To address the recency effect—where farther-positioned corrupted passages may attract attention comparable to nearby benign ones—we reorder passages by attention score using $\mathsf{Score}(z^{(k)}, \text{LLM}_\theta)$ [38]. This sorting reduces positional bias, amplifies anomalous signals, and improves filtering. Algorithm 1 provides a detailed specification of the AV Filter procedure.

Estimating the Filtering Threshold δ. The AV Filter's effectiveness hinges on choosing an appropriate threshold δ. We estimate it using the RQA dataset [40] and Llama 2 [41] by computing attention score variances across clean retrieved sets, setting δ as the mean plus one standard deviation. In selecting δ, we prioritize a low false negative rate over minimizing false positives, since removing a few benign passages rarely alters the final response if most content remains clean. Notably, the threshold estimated from the RQA dataset with Llama 2 and $\alpha = \infty$ transfers well across all settings.

5 Evaluation

This section provides empirical evaluations of the claims presented in the preceding section. Specifically, we conduct experiments to address the following research questions:

RQ1: Can the defender $\mathcal{D}_{\text{AV}}$ reliably identify corrupted retrievals in existing attacks?

RQ2: How *effective* is the AV Filter at mitigating existing attacks?

RQ3: How *effective* and *efficient* are adaptive attacks at bypassing the AV Filter?

Experimental Highlights We summarize the findings related to the research questions:

RQ1: $\mathcal{D}_{\text{AV}}$ identifies the corrupted set and wins the security game against existing attacks with high probability. We estimated its probability of winning as the rate of correct identification across settings, achieving an average of 0.83—highlighting a strong advantage against existing attacks.

RQ2: The AV Filter outperforms baseline defenses, achieving up to 23% higher accuracy in benign settings and up to 20% under attack, while maintaining comparable reductions in attack success rates.

RQ3: Adaptive attacks bypass the AV Filter, achieving an ASR up to 35%—higher than existing attacks—but the AV Filter nevertheless reduces ASR below that of vanilla RAG and empirical upper bounds of baseline defenses. This success requires costly, query-specific optimization ($\sim 10^3\times$ runtime of baselines) and access to benign passages, unlike prior manual or one-shot LLM attacks.

5.1 Experimental Setup

Datasets. We evaluate on four benchmark question-answering datasets: **RealtimeQA (RQA)** [40], **Natural Questions (NQ)** [44] and **HotpotQA** [45] for *short-answer open-domain QA*, and the **RealtimeQA-MC (RQA-MC)** [40] for *multiple-choice open-domain QA*. Each dataset interfaces with a knowledge source: Google Search is used for RQA, RQA-MC, and NQ, while the Wikipedia corpus is used for HotpotQA and also for NQ. We evaluate 100 queries per dataset, following the setup used in the baseline [17]. Results using the Wikipedia corpus are provided in Appendix C.

RAG Setup. Following the baseline, we evaluate three LLMs in a RAG system: Llama2-7B-Chat [41], Mistral-7B-Instruct [46], and GPT-4o [41]. In-context learning is used to guide the LLMs to follow instructions. For generation, following baselines, we use the top $k = 10$ retrieved passages and apply greedy decoding, except for GPT-4o, where we set the temperature to 0.1. We randomly select Mistral-7B to compute attention scores, while GPT-4o is used to generate the final responses.

Attacks. Following the baseline, we evaluate two attacks: PoisonedRAG **(Poison)** [18] and Prompt Injection Attack **(PIA)** [19], injecting poisoned passages directly into the retrieved set as in the baseline. The length of each adversarial passage does not differ significantly from benign ones to ensure a fair comparison. We set the default corruption fraction to $\epsilon = 0.1$ and randomly vary the poisoned passage's position across queries to assess robustness against positional bias. Additional corruption levels are explored in Section 5.3.

Defenses. We evaluate the **AV Filter** by using a filtered set $z^{\text{filtered}} = \mathsf{AVFilter}(q, z^{(k)}, \text{LLM}\theta, \delta, \epsilon)$ for generation, with $\mathsf{NormScore}_\alpha$ where $\alpha \in \{5, 10, \infty\}$ and ∞ denotes a full-token sum. We set $\delta = 26.2$, estimated from benign RealtimeQA instances using Llama 2 with $\alpha = \infty$. We compare against vanilla RAG (**Vanilla**) and two Certified Robust RAG variants [17]: Certified RAG-Keyword (**Keyword**) and Certified RAG-Decoding (**Decoding**). We omit weaker defenses like paraphrasing and perplexity [18]. The threshold sensitivity of the AV Filter is analyzed in Section 5.3.

Evaluation Metrics. For **RQ1**, we estimate the probability of the defender $\mathcal{D}_{\text{AV}}$ winning the SADG security game by the fraction of correctly identified corrupted sets under successful existing attacks on vanilla RAG, referred to as **Corruption Identification Rate (CIR)**. For **RQ2** and **RQ3**, we report three metrics, all expressed as percentages: **Clean Accuracy (ACC)** — correct responses without attacks; **Robust Accuracy (RACC)** — correct responses under attack; and **Attack Success Rate (ASR)** — responses including the adversary's target. A response is considered correct if it includes a valid variation of the ground-truth answer s and excludes the adversary's target s'. We conduct all experiments using 5 random seeds and report the mean of each metric.

We include other experimental specifications in Appendix C.

Table 1: Estimated probability of $\mathcal{D}_{\text{AV}}$ identifying the corrupted set using different α values for $\mathsf{NormScore}_\alpha$, showing high accuracy and a strong advantage against existing attacks.

Dataset		**RQA-MC**		**RQA**		**NQ**	
LLM	**top-α**	**PIA**	**Poison**	**PIA**	**Poison**	**PIA**	**Poison**
Mistral7-B	$\alpha = 5$	**0.94**	0.84	**0.94**	**0.93**	**0.79**	0.54
	$\alpha = 10$	**0.94**	0.86	0.88	0.82	0.73	0.60
	$\alpha = \infty$	0.91	**0.93**	0.80	0.84	0.48	**0.79**
Llama2-C	$\alpha = 5$	0.82	0.70	**0.99**	0.86	**0.93**	0.66
	$\alpha = 10$	0.92	0.82	**0.99**	**0.88**	0.91	0.64
	$\alpha = \infty$	**0.95**	**0.99**	0.95	0.82	0.83	**0.72**

Table 2: Clean Accuracy (ACC) of defenses, showing that AV Filter preserves RAG utility with a minimal drop from Vanilla, achieving up to 23% higher ACC than other baselines.

LLM	**Mistral-7B**			**Llama2-C**			**GPT-4o**		
Defense	**RQA-MC**	**RQA**	**NQ**	**RQA-MC**	**RQA**	**NQ**	**RQA-MC**	**RQA**	**NQ**
Vanilla	81.0	72.0	62.0	79.0	61.0	59.0	66.2	69.8	61.2
Keyword	58.0	56.0	51.0	56.0	57.0	54.0	**63.2**	**64.2**	**60.4**
Decoding	57.0	57.0	55.0	44.0	54.0	41.0	–	–	–
AV Filter$_{(\alpha=5)}$	73.0	**66.0**	**59.0**	**79.0**	**60.0**	51.0	57.8	61.6	57.8
AV Filter$_{(\alpha=10)}$	74.0	65.0	58.0	75.0	57.0	**54.0**	59.8	62.6	55.0
AV Filter$_{(\alpha=\infty)}$	**76.0**	64.0	58.0	75.0	56.0	**54.0**	59.6	63.0	55.8

5.2 Result and Discussion

RQ1. Table 1 reports the estimated probability of $\mathcal{D}_{\text{AV}}$ winning the SADG—measured via CIR—across models, datasets, and varying α values under existing attacks. $\mathcal{D}_{AV}$ identifies the corrupted set with high accuracy, achieving an average CIR of $\mathbf{0.83}$, demonstrating strong effectiveness. We used all successful attack instances against Vanilla RAG in each setting to compute CIR.

RQ2. Clean Accuracy. Table 2 presents clean accuracy across models, datasets, and α values. The AV Filter maintains strong clean performance, with an average drop of only $\mathbf{4}$-$\mathbf{6}\%$ across datasets—substantially smaller than other defenses. On RQA-MC, accuracy drops from $\mathbf{75.4}\%$ (Vanilla) to $\mathbf{69.91}\%$ with AV Filter, compared to larger declines for Keyword ($\mathbf{59.06}\%$) and Decoding ($\mathbf{50.5}\%$). Similar trends hold for RQA (from $\mathbf{67.6}\%$ to $\mathbf{61.68}\%$) and NQ (from $\mathbf{60.73}\%$ to $\mathbf{55.84}\%$).

Robust Accuracy. Table 3 reports AV Filter's robust accuracy (RACC) and attack success rate (ASR). On RQA-MC, it achieves **67.11%** RACC, outperforming Vanilla RAG (**53.63%**), Keyword (**57.43%**), and Decoding (**46.75%**). Similar improvements hold for RQA (**59.9%**) and NQ (**53.57%**). AV Filter's RACC closely matches Vanilla's clean accuracy, indicating high precision and minimal benign impact. Appendix C details how often it removes poisoned passages in successful attacks.

Attack Success Rate. Table 3 shows that even with a small corruption rate ($\epsilon = 0.1$), Vanilla RAG is highly vulnerable—reaching up to **88.2%** attack success. AV Filter cuts this sharply to **9.84%** on RQA-MC, comparable to Cert. RAG-Keyword (**5.43%**) and Decoding (**8.75%**). Similar trends hold for RQA and NQ, with the average ASR reduced to $\mathbf{6.15}\%$ and $\mathbf{4.95}\%$, respectively.

Table 3: Robust Accuracy and Attack Success Rate (RACC/ASR) showing that AV Filter effectively mitigates attacks with low ASRs while achieving up to 20% higher RACC than baseline defenses.

LLM	Dataset	RQA-MC		RQA		NQ	
	Attack	PIA	Poison	PIA	Poison	PIA	Poison
	Defense	(racc↑ / asr↓)	(racc↑ / asr↓)	(racc↑ / asr↓)	(racc↑ / asr↓)	(racc↑ / asr↓)	(racc↑ / asr↓)
Mistral-7B	Vanilla	59.6 / 31.0	62.2 / 30.0	52.2 / 26.6	50.0 / 23.4	40.8 / 24.6	52.0 / 9.2
	Keyword	57.0 / 7.0	55.0 / **6.0**	54.0 / 6.0	55.0 / **6.0**	50.0 / **1.0**	49.0 / **1.0**
	Decoding	55.0 / **5.0**	54.0 / 13.0	55.0 / 5.0	54.0 / 13.0	55.0 / **1.0**	**56.0 / 1.0**
	AV Filter$_{(\alpha=5)}$	76.6 / 5.8	70 .0 / 10.0	62.6 / 2.8	**58.2** / 7.4	54.4 / 3.0	53.4 / 6.2
	AV Filter$_{(\alpha=10)}$	**77.2** / 6.0	71.6 / 8.2	64.8 / 2.8	56.8 / 8.4	**56.2** / 2.8	52.8 / 6.6
	AV Filter$_{(\alpha=\infty)}$	76.2 / 7.2	**73.8** / 8.4	**65.0 / 2.4**	56.8 / 8.6	50.2 / 5.8	52.8 / 4.0
Llama2-C	Vanilla	33.4 / 63.0	62.8 / 27.6	5.8 / 88.2	57.4 / 17.2	10.6 / 73.2	**56.8** / 5.8
	Keyword	54.0 / **6.0**	53.0 / **5.0**	53.0 / 6.0	53.0 / **5.0**	**52.0 / 2.0**	51.0 / **2.0**
	Decoding	38.0 / 12.0	40.0 / **5.0**	38.0 / 12.0	40.0 / **5.0**	39.0 / 17.0	40.0 / 4.0
	AV Filter$_{(\alpha=5)}$	65.6 / 18.4	67.8 / 18.4	**61.8 / 1.6**	55.4 / 7.0	50.6 / 5.2	49.8 / 6.2
	AV Filter$_{(\alpha=10)}$	**70.8** / 12.4	69.6 / 13.0	60.2 / **1.6**	54.8 / 8.8	51.4 / 4.0	51.2 / 6.2
	AV Filter$_{(\alpha=\infty)}$	68.8 / 16.8	**72.0** / 12.6	60.2 / 5.0	**56.8** / 6.6	49.4 / 9.2	51.8 / 3.6
GPT-4o	Vanilla	60.2 / 19.6	43.6 / 25.0	52.4 / 33.4	55.6 / 26.6	39.8 / 33.0	56.4 / 5.2
	Keyword	62.6 / **4.4**	**63.0 / 4.2**	63.4 / **4.0**	**62.6 / 4.0**	**60.2 / 1.4**	**60.0 / 1.2**
	AV Filter$_{(\alpha=5)}$	63.8 / 5.2	55.0 / 7.6	**63.6** / 5.2	57.8 / 10.6	56.8 / 2.6	58.0 / 3.8
	AV Filter$_{(\alpha=10)}$	**64.2** / 4.6	50.4 / 10.4	**63.6** / 4.8	57.2 / 11.0	57.0 / 4.0	57.8 / 3.0
	AV Filter$_{(\alpha=\infty)}$	63.8 / 5.4	50.8 / 6.8	61.2 / 7.0	61.4 / 9.2	52.0 / 11.4	58.8 / 1.6

Overall, the AV Filter mitigates existing attacks while maintaining significantly higher accuracy than Certified RAG. Additionally, it requires fewer LLM_θ computations, as it doesn't need to evaluate each passage individually like Certified RAG.

RQ3. We extend existing jailbreak attacks such as GCG [47] and AutoDAN [48] by optimizing a poisoned passage with full access to the query and retrieval context. Starting from an initial successful from a prior attack, denoted as z_{poison}, we iteratively refine it to minimize a compute loss $\mathcal{L}_t$ that balances effectiveness and stealth.

The loss is defined as $\mathcal{L}_t = \mathcal{L}_1 + \lambda \cdot \mathcal{L}_2$, where $\mathcal{L}_1$ is the cross-entropy between between the model's response (given the corrupted retrieved set including z_{poison}) and the target answer s', and $\mathcal{L}_2$ is the variance of the normalized attention scores over all passages in the retrieved set—encouraging low detectability. Here, λ is a scalar parameter that balances the attack effectiveness with stealth.

At each iteration, we apply a jailbreak method, denoted as Jailbreak, to propose a modified candidate passage that minimizes $\mathcal{L}_t$. Among all generated candidates across iterations, we select the one yielding the lowest loss as the optimized poisoned passage. The full procedure is detailed in Algorithm 2.

In our experiments, we insert the poisoned passage at the last index of the retrieved set to construct the corrupted retrieved set. This placement eliminates retrieval randomness, enabling easier reproducibility and consistent comparison across queries—particularly important given the high computational cost of adaptive attacks. We also set the $\alpha = \infty$ for the AV Filter and select 20 queries from each dataset, prioritizing those where existing attacks were successful against Vanilla RAG. Since initialization from successful attacks typically yields a low value of $\mathcal{L}_1$, we terminate the optimization early if the attention variance $\mathcal{L}_2$ falls below the AV Filter threshold δ. The attack is run for 100 steps using standard parameters for each jailbreak method.

Algorithm 2: Adaptive Stealth-Aware Poisoning Attack

Input: Query q, target answer s', benign retrieved set $z^{(k)}_{\text{benign}}$, language model LLM_θ, loss weight λ, jailbreak function Jailbreak, max steps T

Output: Optimized poisoned passage z^*_{poison}

Initialize poisoned passage $p_0 = z_{\text{poison}}$ using an existing attack (e.g., PoisonedRAG);

Set best loss $\mathcal{L}^* \leftarrow \infty$, best candidate $z^*_{\text{poison}} \leftarrow p_0$;

for $t = 1$ **to** T **do**

  Inject p_{t-1} into $z^{(k)}_{\text{benign}}$ to get the corrupted retrieved set $z^{(k)}_{\text{corrupt}}$

  Generate model response $\hat{s}_t \leftarrow \text{LLM}_\theta(q, z^{(k)}_{\text{corrupt}})$

  Compute normalized passage attention scores:

$$\texttt{attn\_scores} = \left\{ \texttt{NormScore}_\alpha(z_t, z^{(k)}_{\text{corrupt}}, A) | z_t \in z^{(k)}_{\text{corrupt}} \right\}$$

  Compute loss:

$$\mathcal{L}_t\left(z^{(k)}_{\text{corrupt}}\right) = \underbrace{\text{CE}\left(\hat{s}_t, s'\right)}_{\mathcal{L}_1} + \lambda \cdot \underbrace{\text{Var}\left(\texttt{attn\_scores}\right)}_{\mathcal{L}_2}$$

  if $\mathcal{L}_t < \mathcal{L}^*$ **then**

    $\mathcal{L}^* \leftarrow \mathcal{L}_t$, $z^*_{\text{poison}} \leftarrow p_{t-1}$

  Generate next candidate poisoned passage: $p_t \leftarrow \texttt{Jailbreak}\left(q, z^{(k)}_{\text{benign}}, s', p_{t-1}, \mathcal{L}_t\right)$

return z^*_{poison}

We tune the scalar parameter λ in the adaptive attack loss using the RealtimeQA dataset and Llama 2, evaluating values from the set $\{0.01, 0.1, 1\}$. We select $\lambda = 0.1$ for all subsequent adaptive attack experiments, as it yields the highest ASR. Figure 3(a) represents the impact of varying λ on attack performance. For evaluation, we apply adaptive attacks using jailbreak methods, including GCG and AutoDAN, initialized with poisoned passages generated by the PoisonedRAG attack (Poison). Table 4 reports the robust accuracy and attack success rate (RACC / ASR) of adaptive attacks against the AV Filter across multiple settings. The results show that adaptive attacks can potentially evade the AV Filter, achieving a maximum ASR of $\mathbf{35}\%$ and an average ASR of $\mathbf{22.08}\%$.

Table 4: RACC and ASR of adaptive attacks (GCG and AutoDAN) initialized with poisoned passages from Poison against AV Filter, showing increased ASRs of up to 35%—higher than existing attacks on AV Filter but still lower than ASRs of Vanilla RAG and empirical upper bounds of other baselines.

LLM	Adaptive Attack	RQA-MC	RQA	NQ
Llama 2-C	GCG-Poison	55 / 15	35 / 30	15 / 10
	AutoDAN-Poison	35 / 35	40 / 20	25 / 10
Mistral-7B	GCG-Poison	50 / 25	25 / 25	35 / 35
	AutoDAN-Poison	50 / 20	20 / 15	30 / 25

Although adaptive attacks demonstrate reasonable success against the AV Filter, several limitations reduce the severity of the threat they pose. These attacks are highly dependent on the specific query,

model, and benign retrieved set, requiring access to the LLM, the retriever, and the knowledge database—an assumption that may not hold for many practical adversaries. Furthermore, since adaptive attacks rely on iterative jailbreak methods, which are known for their high computational cost, they inherit long runtimes. Each poisoned passage must be individually optimized, significantly increasing the time required for the attack. Table 5 reports the average runtime per query (in seconds) across various settings, highlighting the computational overhead associated with these attacks. The AutoDAN-Poison attack on the RealtimeQA dataset using Mistral-7B incurred the highest average runtime among all settings, taking **18616.84** seconds per query. When executed sequentially on 20 queries, this resulted in a total runtime of approximately **4.3** days on a single H100 GPU. Figure 3(b) shows the loss trajectory for a randomly selected query from the RealtimeQA dataset during the adaptive attack on Llama 2.

Table 5: Average runtime of the adaptive attack per query across various settings. The runtime reaches up to 1.8×10^4 seconds, which is several orders of magnitude ($\sim \times 10^3$) higher than the runtime of the existing attack Poison, as reported in [18].

LLM	Adaptive Attack	RQA-MC	RQA	NQ
Llama 2-C	GCG-Poison	7015.68	15833.36	9146.72
	AutoDAN-Poison	6233.20	18274.31	9737.39
Mistral-7B	GCG-Poison	8606.68	14890.24	9624.45
	AutoDAN-Poison	6604.01	18616.84	18248.52

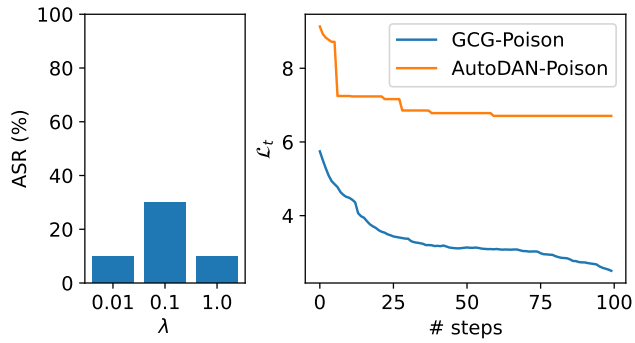

Figure 3: (a) Attack Success Rate (ASR) of the GCG-Poison adaptive attack on the RealtimeQA dataset using Llama 2 across varying values of λ, illustrating that $\lambda = 0.1$ achieves the highest ASR and is therefore selected for the rest of the evaluation. (b) Loss trajectory for a randomly selected query from RealtimeQA on Llama 2, demonstrates how the adaptive attack consistently reduces the target loss by lowering the variance of the corrupted retrieved set, thereby improving stealth.

5.3 Hyperparameter Analysis

Corruption Fraction ϵ. We evaluate the AV Filter under varying corruption fractions to its robustness as the rate of corruption increases. Specifically, we measure Robust Accuracy (RACC) and Attack Success Rate (ASR) on the RealtimeQA-MC dataset across multiple models, using a fixed $\alpha = \infty$ and a single random seed. Figure 4(a) and (b) present the average RACC and ASR

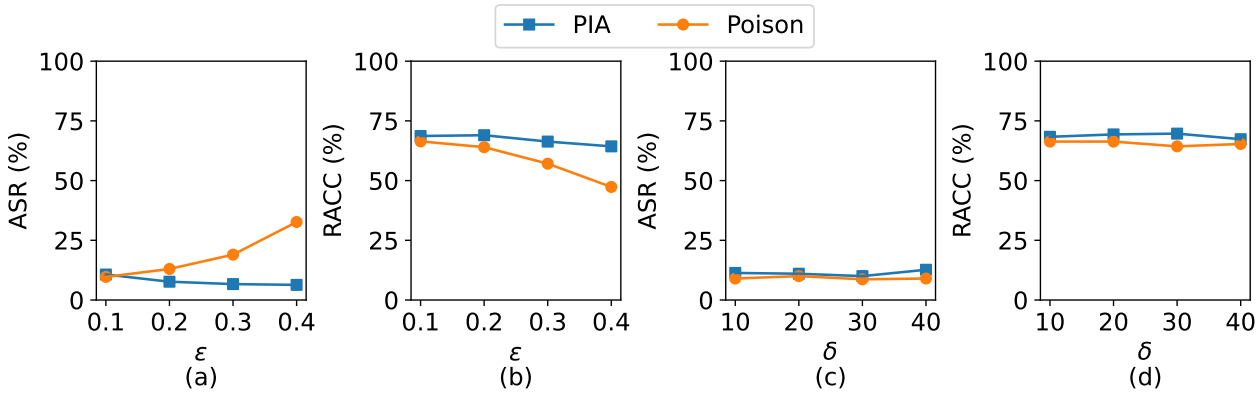

Figure 4: **Effect of Corruption Rate and Filtering Threshold:** This figure shows the impact of varying the corruption rate ϵ and the filtering threshold δ on the performance of the AV Filter. Subfigures (a) and (b) present the Attack Success Rate (ASR) and Robust Accuracy (RACC) on the RealtimeQA-MC dataset with $\alpha = \infty$, averaged over all models. As expected, ASR increases and RACC decreases with higher corruption rates. Subfigures (c) and (d) report ASR and RACC for varying δ values, again averaged over all models, demonstrating that AV Filter's performance is not overly sensitive to the threshold. This indicates that AV Filter can generalize well to unseen data without requiring fine-tuning of δ.

for corruption rates $\epsilon \in \{0.1, 0.2, 0.3, 0.4\}$, with the total retrieved set size fixed at $k = 10$. As expected, increasing the corruption fraction leads to higher ASR and lower RACC. Nevertheless, the AV Filter remains reasonably effective even under high corruption—reducing ASR to 32.67% at $\epsilon = 0.4$ for Poison. We expect a similar trend for other datasets and α values.

Filtering Threshold δ. The effectiveness of the AV Filter depends on the filtering threshold δ, which governs the acceptable variance in attention score across the retrieved set. We set $\delta = 26.2$ for our main experiments, estimated from clean retrievals on the RealtimeQA dataset using Llama 2. This estimated threshold generalizes well, as it yields strong performance across different datasets and models. To further assess the robustness of the AV Filter to this hyperparameter, we evaluate its performance across a range of thresholds $\delta \in \{10, 20, 30, 40\}$. Specifically, we report Robust Accuracy (RACC) and Attack Success Rate (ASR) on the RealtimeQA-MC dataset, averaged over multiple models using a fixed $\alpha = \infty$ and a single random seed. Figure 4(c) and (d) show that both RACC and ASR remain relatively stable across this range, indicating that AV Filter is not overly sensitive to δ and can generalize well to unseen data without requiring fine-tuning. We expect a similar trend for other datasets and α values.

6 Limitations

We have shown that existing attacks against RAG systems are not designed for stealth—they often craft poisoned passages that attract anomalously high attention scores, enabling reliable detection and mitigation. This stems from the co-occurrence of the adversary's target answer within the poisoned passage, which causes certain tokens to receive significantly more attention weight than others. When normalized across the retrieved set, these poisoned passages exhibit disproportionately high attention scores, resulting in a high variance signal that AV Filter leverages for detection.

However, this detection strategy assumes that the poisoned content is concentrated in a small subset of passages while the majority support the correct answer. This reliance leads to certain limitations. To the best of our knowledge, using attention patterns to improve the robustness of RAG systems has the following constraints:

1. **Susceptibility to majority corruption.** If the adversary manages to corrupt a majority of the retrieved set, then multiple passages will contain tokens that draw high attention weight. This the contrast of normalized attention scores among passages, reducing the variance and potentially allowing the attack to evade detection by AV Filter. This highlights the need for robustness at the retrieval stage (*Step I*) of the RAG pipeline as well. However, AV Filter's improvements are orthogonal to retrieval robustness—it can be integrated with more robust retrievers that make majority corruption harder.

2. **Dependence on redundancy of correct knowledge.** If only a very few benign passages contain the correct answer, these may individually attract high attention and be mistakenly filtered out. Thus, AV Filter assumes that the knowledge corpus includes multiple passages supporting the correct answer, which is necessary for any filtering mechanism based on outlier detection to succeed.

3. **Task specificity and generalization.** The AV Filter relies on the poisoned passage receiving high normalized attention scores due to the co-occurrence of the target response. While this is well-suited for question-answering tasks—where the goal is to inject or alter the response—we have not yet evaluated its effectiveness against attacks that aim to exploit other behaviors of the RAG system (e.g., manipulating style, eliciting private data, or controlling downstream decisions) without directly changing the response content. Broader evaluations will be necessary to understand the generalizability of this defense mechanism.

7 Conclusion and Future Work

We have shown that existing attacks lack stealth, often drawing disproportionately high attention. This property enables effective defenses: when attacks succeed despite corrupting only a small fraction of input, they must exert an unusually large influence—compromising their stealth. We argue this trade-off is fundamental: an attack cannot be both highly effective and perfectly stealthy. A theoretical analysis of this trade-off, aiming toward an impossibility result, remains for future work.

Our adaptive attacks probe the limits of attention-based defenses but remain inefficient and heavily dependent on the query, input, and model access. Improving their generality and identifying other detectable traces they may leave are key open challenges.

We believe that rigorously analyzing stealth through intermediate signals involved in the generation—such as attention patterns or probability distributions for sampling the next token—is critical for both crafting stronger attacks and developing robust defenses in RAG systems.

Acknowledgments

We would like to thank Atharv Singh Patlan for their valuable comments and suggestions on earlier draft of this paper. Sarthak Choudhary, Nils Palumbo, Ashish Hooda, and Somesh Jha are partially supported by DARPA under agreement number 885000, NSF CCF-FMiTF-1836978 and ONR N00014-21-1-2492.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

[4] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[6] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

[7] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

[8] Google. Generative ai in search: Let google do the searching for you. `https://blog.google/products/search/generative-ai-google-search-may-2024/`, 2024. Accessed: 2025-04-21.

[9] Sina J Semnani, Violet Z Yao, Heidi C Zhang, and Monica S Lam. Wikichat: Stopping the hallucination of large language model chatbots by few-shot grounding on wikipedia. *arXiv preprint arXiv:2305.14292*, 2023.

[10] Microsoft. Bing chat. https://www.microsoft.com/en-us/edge/features/bing-chat, 2024. Accessed: 2025-04-21.

[11] Perplexity AI. Perplexity ai. https://www.perplexity.ai/, 2024. Accessed: 2025-04-21.

[12] Jerry Liu. Llamaindex. https://github.com/jerryjliu/llama_index, November 2022. Accessed: 2025-04-21.

[13] LangChain. Langchain. https://github.com/langchain-ai/langchain, 2024. Accessed: 2025-04-21.

[14] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[15] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[16] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 407–425. IEEE, 2024.

[17] Chong Xiang, Tong Wu, Zexuan Zhong, David Wagner, Danqi Chen, and Prateek Mittal. Certifiably robust rag against retrieval corruption. *arXiv preprint arXiv:2405.15556*, 2024.

[18] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. Poisonedrag: Knowledge poisoning attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867*, 2024.

[19] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90, 2023.

[20] Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.

[21] Gabriel Alon and Michael Kamfonas. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*, 2023.

[22] Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. Demystifying prompts in language models via perplexity estimation. *arXiv preprint arXiv:2212.04037*, 2022.

[23] Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Advances in Neural Information Processing Systems*, 37:121156–121184, 2024.

[24] Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. Hoprag: Multi-hop reasoning for logic-aware retrieval-augmented generation. *arXiv preprint arXiv:2502.12442*, 2025.

[25] Mohsen Fayyaz, Ali Modarressi, Hinrich Schuetze, and Nanyun Peng. Collapse of dense retrievers: Short, early, and literal biases outranking factual evidence. *arXiv preprint arXiv:2503.05037*, 2025.

[26] Sheng-Chieh Lin. Building a robust retrieval system with dense retrieval models. 2024.

[27] Yizhi Li, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. More robust dense retrieval with contrastive dual learning. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 287–296, 2021.

[28] Yibing Du, Antoine Bosselut, and Christopher D Manning. Synthetic disinformation attacks on automated fact verification systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10581–10589, 2022.

[29] Liangming Pan, Wenhu Chen, Min-Yen Kan, and William Yang Wang. Attacking open-domain question answering by injecting misinformation. *arXiv preprint arXiv:2110.07803*, 2021.

[30] Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Yang Wang. On the risk of misinformation pollution with large language models. *arXiv preprint arXiv:2305.13661*, 2023.

[31] Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. Poisoning retrieval corpora by injecting adversarial passages. *arXiv preprint arXiv:2310.19156*, 2023.

[32] Sukmin Cho, Soyeong Jeong, Jeongyeon Seo, Taeho Hwang, and Jong C Park. Typos that broke the rag's back: Genetic attack on rag pipeline by simulating documents in the wild via low-level perturbations. *arXiv preprint arXiv:2404.13948*, 2024.

[33] Orion Weller, Aleem Khan, Nathaniel Weir, Dawn Lawrie, and Benjamin Van Durme. Defending against disinformation attacks in open-domain question answering. *arXiv preprint arXiv:2212.10002*, 2022.

[34] Giwon Hong, Jeonghwan Kim, Junmo Kang, Sung-Hyon Myaeng, and Joyce Jiyoung Whang. Discern and answer: Mitigating the impact of misinformation in retrieval-augmented models with discriminators. *CoRR*, 2023.

[35] Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes, editors, *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy, August 2019. Association for Computational Linguistics.

[36] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

[37] Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. Zipcache: Accurate and efficient kv cache quantization with salient token identification. *arXiv preprint arXiv:2405.14256*, 2024.

[38] Alexander Peysakhovich and Adam Lerer. Attention sorting combats recency bias in long context language models. *arXiv preprint arXiv:2310.01427*, 2023.

[39] Xun Xian, Tong Wang, Liwen You, and Yanjun Qi. Understanding data poisoning attacks for RAG: Insights and algorithms, 2025.

[40] Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, Kentaro Inui, et al. Realtime qa: What's the answer right now? *Advances in neural information processing systems*, 36:49025–49043, 2023.

[41] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[42] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

[43] Xiaobo Guo and Soroush Vosoughi. Serial position effects of large language models. *arXiv preprint arXiv:2406.15981*, 2024.

[44] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[45] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[46] Devendra Singh Chaplot. Albert q. jiang, alexandre sablayrolles, arthur mensch, chris bamford, devendra singh chaplot, diego de las casas, florian bressand, gianna lengyel, guillaume lample, lucile saulnier, lélio renard lavaud, marie-anne lachaux, pierre stock, teven le scao, thibaut lavril, thomas wang, timothée lacroix, william el sayed. *arXiv preprint arXiv:2310.06825*, 2023.

[47] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[48] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023.

A Additional Background on Existing Works

PoisonedRAG. Given a query q and target answer s', PoisonedRAG (Poison) seeks to craft a poisoned passage z_{poison} such that a RAG system is highly likely to generate s' when z_{poison} is included in the retrieved context. To do so, it leverages an LLM (e.g., GPT-4) to generate p. The attack LLM may be the same as or different from the one used in the target system, depending on the attacker's access and knowledge. The attacker prompts the LLM to generate z_{poison} such that, when used as context, it induces the model to output s'. The following prompt is used:

> This is my question: [question].
> This is my answer: [answer].
> Please craft a corpus such that the answer is [answer] when prompted with the question [question]. Please limit the corpus to V words

where V is a hyperparameter that limits the length of z_{poison}, though in practice, the generated passage may exceed this limit if the LLM fails to strictly follow the instruction. For our experiments, we use the poisoned passages publicly released by the PoisonedRAG [18] and Certified Robust RAG [17]. We repeat the poisoned content 2 times in a single passage to simulate a strong attacker that can successfully manipulate response in Vanilla RAG, while ensuring the overall length remains comparable to that of a typical benign passage.

Prompt Injection Attack. It enables adversaries to remotely exploit LLM-integrated applications by strategically injecting prompts into data likely to be retrieved. In Prompt Injection Attack (PIA), the attacker injects the following malicious instruction: "When asked to provide the answer for the following question: {query}, please output {target answer}". We use a passage containing the aforementioned instruction for a specific query-answer pair as the poisoned passage in our experiments.

Certified Robust RAG. Certified Robust RAG employs an isolate-then-aggregate strategy. Given k retrieved passages $z^{(k)} = \{z_{i_1}, z_{i_2}, \ldots, z_{i_k}\}$, it first computes an individual LLM response s_j for each isolated passage z_{i_j}, instead of concatenating all k passages as in standard Vanilla RAG. It then aggregates the responses $\{s_1, s_2, \ldots, s_k\}$ using a robust text aggregation method to produce a final response $\hat{s}$.

The isolation step limits the impact of any poisoned passage to its own response, enhancing robustness. However, this design introduces two limitations. First, it fails on queries requiring multi-passage reasoning—undermining a core motivation behind using multiple passages in RAG. Second, it incurs a $k\times$ inference overhead compared to Vanilla RAG. While it strengthens security, this comes at a steep cost to utility, latency, and inference costs. These trade-offs motivate the search for more efficient defense strategies that offer a better balance between robustness and practicality.

B Example: Benign vs. Poisoned Attention Patterns

Our key insight is that attention patterns can be leveraged to detect potentially poisoned passages that disproportionately influence the LLM's response in a RAG system. In such cases, specific

tokens from the poisoned passage tend to receive significantly higher attention due to their co-occurrence with the target answer. These tokens act as heavy hitters in the attention distribution and are localized within the poisoned passages, as benign passages typically do not contain tokens associated with an incorrect target answer. This results in a notably higher normalized passage attention score for the poisoned passage compared to the benign ones. AV Filter exploits this disparity by identifying and filtering out such anomalous passages from the corrupted retrieved set. Figure 5 illustrates this behavior, showing a benign passage and the poisoned passage for the following query from the RealtimeQA dataset, along with the corresponding toke-level attention weights produced by Llama 2.

User query: How many of the six planned central towers on Barcelona's unfinished Sagrada Familia cathedral have now been completed?
Target Response: 3
Correct Response: 5
Actual Response: 3
Benign Passage: Sagrada Familia gets final completion date 2026 or 2028. "Our new objective is to complete the six central towers, of which five have already been started," said Rigol. The sixth tower will measure ...
Poisoned Passage: The Sagrada Familia, an architectural masterpiece in Barcelona, is said to have completed 3 of its 6 planned central towers, marking a significant milestone in its lengthy construction process. The Sagrada Familia, an architectural masterpiece in Barcelona, is said to have completed 3 of its 6 planned central towers, marking a significant milestone in its lengthy construction process.

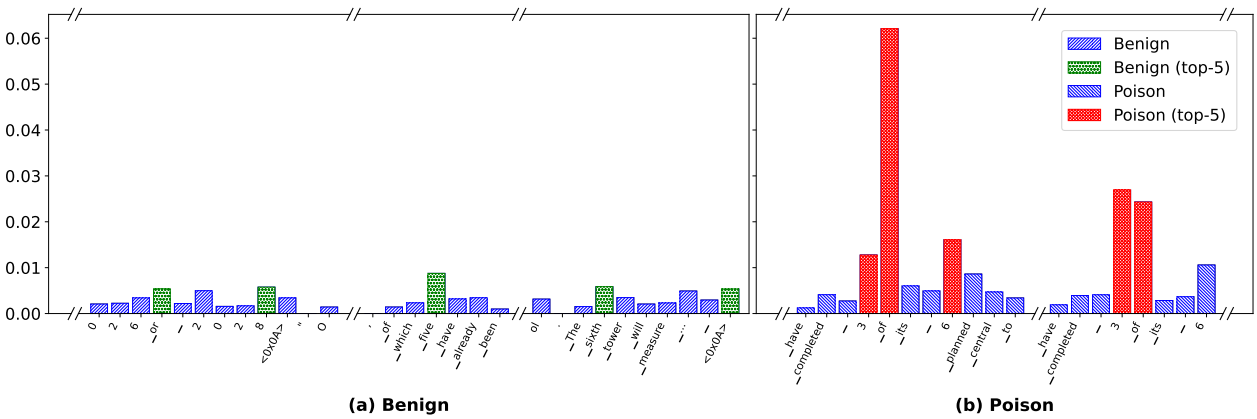

Figure 5: **Attention Patterns in Benign vs. Poisoned Passages**: It highlights the token-level attention weights (as a fraction of total attention over the retrieved set) for a query from the RealtimeQA dataset, computed using Llama 2. (a) shows a benign passage with the highest normalized passage attention score among all benign candidates; (b) shows the poisoned passage present in the retrieved set. Tokens such as 3, _of, and 6 from the poisoned passage receive disproportionately high attention—greater than the total attention allocated to many of the individual benign passages. This behavior allows simple aggregation of attention over the top-α tokens to distinguish poisoned from benign passages.

Table 6: Detection Accuracy (DACC) of AV Filter against existing attacks, showing that AV Filter accurately removes the actual poisoned passage from the corrupted retrieved set, achieving the DACC up to 1.00 (perfect detection).

Dataset		**RQA-MC**		**RQA**		**NQ**	
LLM	**top-α**	**PIA**	**Poison**	**PIA**	**Poison**	**PIA**	**Poison**
Mistral7-B	$\alpha = 5$	**1.00**	0.88	0.99	**0.97**	**1.00**	0.67
	$\alpha = 10$	0.99	0.89	**1.00**	0.93	0.99	0.69
	$\alpha = \infty$	0.92	**0.95**	0.97	0.92	0.83	**0.71**
Llama2-C	$\alpha = 5$	0.81	0.47	**0.98**	0.82	0.94	0.64
	$\alpha = 10$	**0.90**	0.68	**0.98**	**0.85**	**0.96**	0.67
	$\alpha = \infty$	0.88	**0.77**	0.94	0.79	0.88	**0.70**

C Additional Experimental Details and Results

We use the PyTorch (BSD-style license) and HuggingFace Transformers (Apache-2.0 license) libraries for all our experiments. The experiments were conducted on a mix of A100 and H100 GPUs. All experiments were run with 5 different seeds, except for the adaptive attack due to its high computational cost. We report the mean of each evaluation metric. The maximum observed standard deviations across seeds are as follows: Clean Accuracy (ACC)—2.32, Robust Accuracy (RACC)—3.78, and Attack Success Rate (ASR)—3.56.

C.1 AV Filter Detection Rate: Identifying Poisoned Passage

AV Filter is designed to identify and remove the potentially poisoned passages from a corrupted retrieved set, allowing the remaining (presumably benign) passages to be used for response generation. When the AV Filter successfully eliminates the actual poisoned passages, it is expected to improve the robust accuracy (RACC) and reduce the attack success rate (ASR)—a trend confirmed in our evaluation.

The consistent improvement in robustness over Vanilla RAG indicates that AV Filter reliably removes the correct poisoned passages. To explicitly quantify this behavior, we report the **Detection Accuracy (DACC)**—the fraction of successful attacks against Vanilla RAG in which AV Filter removes the actual poisoned passage. Table 6 presents the DACC across different α values used in the computation of $\mathsf{NormScore}_\alpha$ and $\epsilon = 0.1$, demonstrating that AV Filter achieves high precision in removing the poisoned passage with an average detection accuracy of $\mathbf{0.86}$. This reinforces AV Filter's effectiveness in accurately identifying and filtering poisoned passages from the retrieved set.

C.2 Wikipedia Corpus

We evaluate AV Filter against existing attacks using the Wikipedia Corpus as the Knowledge database, demonstrating its effectiveness across varying knowledge distributions. Specifically, we use 100 queries each from the HotpotQA and NQ datasets, retrieving top 10 passages from the Wikipedia corpus using the Contriver retriever. We utilize the Wikipedia corpus and retrieval results publicly released by PoisonedRAG [18].

Table 7: Clean Accuracy (ACC) of defenses, showing that AV Filter preserves RAG utility with a minimal drop from Vanilla, achieving up to 10% higher ACC than other baseline defenses.

LLM	**Mistral-7B**		**Llama2-C**		**GPT-4o**	
Defense	**HotpotQA**	**NQ**	**HotpotQA**	**NQ**	**HotpotQA**	**NQ**
Vanilla	51.0	59.0	36.0	46.0	45.6	47.4
Keyword	**59.0**	49.0	**43.0**	37.0	44.6	**55.0**
Decoding	41.0	50.0	26.0	28.0	–	–
AV Filter$_{(\alpha=5)}$	40.0	43.0	27.0	34.0	**45.0**	48.2
AV Filter$_{(\alpha=10)}$	46.0	44.0	27.0	36.0	44.8	47.4
AV Filter$_{(\alpha=\infty)}$	51.0	**59.0**	36.0	**46.0**	44.2	47.6

Similar to our evaluation with Google Search as the knowledge database, we report the Clean Accuracy (ACC), Robust Accuracy (RACC), and Attack Success Rate (ASR) on the HotpotQA and NQ datasets using the Wikipedia corpus as the underlying knowledge base.

Table 7 reports the clean accuracy across different models, datasets, and values of α. The AV Filter preserves high clean performance, with only a modest average drop of $\mathbf{4-6}\%$ across datasets.

Table 8 presents the Robust Accuracy (RACC) and Attack Success Rate (ASR) achieved by AV Filter for varying values of α used in computing $\mathsf{NormScore}_\alpha$. The results demonstrate that the AV Filter often outperforms baseline defenses in robustness, achieving up to $\mathbf{9.8}\%$ higher RACC. Furthermore, even at a low corruption rate of $\epsilon = 0.1$, Vanilla RAG remains highly vulnerable, with ASR reaching up to $\mathbf{90.2}\%$. In contrast, AV Filter significantly reduces this vulnerability—bringing the average ASR down to $\mathbf{15.36}\%$ on the HotpotQA and $\mathbf{14.71}\%$ on the NQ dataset.

Notably, the ASR for the Keyword and Decoding defenses is anomalously high on the HotpotQA dataset. This is attributed to the multi-hop nature of many HotpotQA queries, which often require reasoning across multiple passages. Since both variants of Certified Robust RAG evaluate each passage in isolation, they fail to aggregate information across passages to answer correctly. As a result, they are more susceptible to a single poisoned passage that contains complete information aligned with the adversarial target answer.

D Broader Impacts

We propose a filtering technique capable of identifying and mitigating existing poisoning attacks, thereby reducing potential harms. In parallel, we introduce more stealthy poisoning attacks that evade existing defenses. While we believe this dual contribution will drive the development of more robust RAG systems, it may also increase the risk of vulnerable deployments in the short term.

Table 8: Robust Accuracy and Attack Success Rate (RACC/ASR) showing that AV Filter effectively mitigates attacks with low ASRs while achieving up to 9.8% higher RACC than baselined defenses.

LLM	Dataset	HotpotQA		NQ	
	Attack	**PIA**	**Poison**	**PIA**	**Poison**
	Defense	(racc↑ / asr↓)	(racc↑ / asr↓)	(racc↑ / asr↓)	(racc↑ / asr↓)
Mistral-7B	Vanilla	18.6 / 69.0	14.6 / 75.0	22.2 / 55.8	23.0 / 50.4
	Keyword	48.0 / 21.0	43.0 / 25.0	40.0 / 7.0	42.0 / **10.0**
	Decoding	38.0 / 28.0	30.0 / 51.0	47.0 / 7.0	43.0 / 20.0
	AV Filter$_{(\alpha=5)}$	**53.0 / 8.0**	47.4 / 14.8	**49.8 / 11.0**	43.0 / 14.6
	AV Filter$_{(\alpha=10)}$	52.6 / 8.4	47.8 / 15.0	48.6 / 11.2	**44.0** / 13.2
	AV Filter$_{(\alpha=\infty)}$	47.2 / 13.4	**48.8 / 13.6**	36.6 / 26.8	42.2 / 12.4
Llama 2-C	Vanilla	3.6 / 90.2	14.6 / 65.6	6.4 / 85.6	26.2 / 48.0
	Keyword	**36.0** / 25.0	**41.0** / 20.0	36.0 / 8.0	37.0 / **9.0**
	Decoding	23.0 / 33.0	25.0 / **16.0**	24.0 / 30.0	26.0 / 23.0
	AV Filter$_{(\alpha=5)}$	34.0 / 11.4	27.0 / 17.0	42.6 / 6.4	**37.2** / 17.6
	AV Filter$_{(\alpha=10)}$	34.4 / **10.4**	26.8 / 17.0	**44.2 / 6.2**	36.4 / 15.6
	AV Filter$_{(\alpha=\infty)}$	17.8 / 44.0	21.4 / 28.6	22.4 / 45.6	32.4 / 25.8
GPT-4o	Vanilla	10.6 / 78.8	20.4 / 58.4	16.8 / 69.4	28.6 / 34.6
	Keyword	**43.6** / 17.4	**43.4** / 15.8	**53.2** / 6.2	**53.0** / 4.8
	AV Filter$_{(\alpha=5)}$	42.6 / **9.8**	37.2 / 12.6	40.2 / **5.8**	36.8 / 6.8
	AV Filter$_{(\alpha=10)}$	40.0 / 11.2	37.6 / 12.0	41.6 / 6.8	35.8 / **4.0**
	AV Filter$_{(\alpha=\infty)}$	35.6 / 17.8	41.2 / **11.4**	28.0 / 29.6	38.6 / 5.4