Detecting RAG Extraction Attack via Dual-Path Runtime Integrity Game

Yuanbo Xie[1,2], Yingjie Zhang[1,2], Yulin Li[1,2], Shouyou Song[3], Xiaokun Chen[4], Zhihan Liu[5], Liya Su[6], Tingwen Liu[1,2,*]
[1]Institute of Information Engineering, Chinese Academy of Sciences, China,
[2]School of Cyber Security, University of Chinese Academy of Sciences, China,
[3]Beijing University of Post and Telecommunications, China, [4]Stanford University,
[5]North China Electric Power University, China,
[6]AI Sec Lab, Beijing Chaitin Technology Co.,Ltd
{xieyuanbo,liutingwen}@iie.ac.cn

Abstract

Retrieval-Augmented Generation (RAG) systems augment large language models with external knowledge, yet introduce a critical security vulnerability: RAG Knowledge Base Leakage, wherein adversarial prompts can induce the model to divulge retrieved proprietary content. Recent studies reveal that such leakage can be executed through adaptive and iterative attack strategies (named RAG extraction attack), while effective countermeasures remain notably lacking. To bridge this gap, we propose CanaryRAG, a runtime defense mechanism inspired by stack canaries in software security. CanaryRAG embeds carefully designed canary tokens into retrieved chunks and reformulates RAG extraction defense as a dual-path runtime integrity game. Leakage is detected in real time whenever either the target or oracle path violates its expected canary behavior, including under adaptive suppression and obfuscation. Extensive evaluations against existing attacks demonstrate that CanaryRAG provides robust defense, achieving substantially lower chunk recovery rates than state-of-the-art baselines while imposing negligible impact on task performance and inference latency. Moreover, as a plug-and-play solution, CanaryRAG can be seamlessly integrated into arbitrary RAG pipelines without requiring retraining or structural modifications, offering a practical and scalable safeguard for proprietary data.

1 Introduction

Large Language Models (LLMs) are limited by their lack of access to up-to-date and specialized knowledge, leading to a tendency to generate hallucinations (Shuster et al., 2021). Retrieval-Augmented Generation (RAG) has emerged as a standard paradigm for augmenting LLMs with external knowledge, allowing the integration of up-to-date, domain-specific, and proprietary information during inference (Lewis et al., 2020; Guu et al., 2020). RAG-based systems are now widely deployed in enterprise assistants, customer support agents, and agentic workflows that rely on private knowledge bases. The RAG knowledge bases often contain highly valuable assets and constitute a core competitive advantage of commercial RAG systems and agent-based products.

However, by exposing internal retrieval results to the generation process, RAG systems also introduces a critical security vulnerability: knowledge base leakage, where adversarial prompts can induce the LLMs or Agents to divulge retrieved content to unauthorized users (Zeng et al., 2024). Such vulnerability poses the risk of reconstructing RAG knowledge bases, where adversaries can adaptively query a deployed RAG service with different topics and aggregating retrieved content across topics. Recent studies (Qi et al., 2025; Cohen et al., 2024; Di Maio et al., 2024; Jiang et al., 2025) demonstrate that RAG systems are vulnerable to RAG extraction attacks through black-box prompt interactions, without access to retrieval indices or system prompts. This may result in serious legal, economic, and reputational consequences for organizations.

Despite the growing severity of RAG extraction attacks, dedicated defense mechanisms remain scarce and still in their early stage. Prior defense approaches primarily focus on two dimensions: **intra-class protection**, which prevents excessive exposure or reconstruction within a single topic, and **inter-class protection**, which blocks unintended knowledge diffusion across different topics. Zeng et al. (2024) explore retrieval-side isolation (e.g. Reranker) or post-retrieval filtration (e.g. Summarize) to avoid exposing retrieved content irrelevant to the query. Li et al. (2025) introduce a dual-path defense mechanism named RAGFort, which combines contrastive reindexing for strengthening the semantic boundaries between different topics, and

*Corresponding Author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8503–8518

constrained cascade generation for controlling the exposure of sensitive fine-grained information during response generation.

Prior defense mechanisms, however, confront two fundamental technical limitations. First, they are inherently passive in nature, primarily designed to raise the cost of reconstruction for an adversary. These methods lack the capability to proactively identify malicious attackers, particularly during the early stages of an attack. Second, they generally cannot be deployed in a plug-and-play manner, as they necessitate intrusive alterations to the standard RAG pipelines, such as restructuring the retrieval or indexing pipeline, or incorporating task-specific constrained generation. Third, they remain vulnerable to strong and adaptive extraction attacks, which can still induce significant information leakage, demonstrating that current defenses offer only limited protection.

This paper approaches the RAG knowledge base leakage problem from a detection standpoint. The intuitive solution, comparing generated output with retrieved content and flagging high similarity as leakage, faces challenges at multiple levels. At the lexical level, character- or n-gram-based matching can be easily bypassed through output obfuscation. At the semantic level, however, legitimate RAG outputs that appropriately utilize retrieved content are often indistinguishable from intra-class leakage, as both are semantically grounded in the same source material. Consequently, semantic similarity alone fails to reliably separate authorized knowledge use from unauthorized disclosure, since the distinction lies in intent rather than observable meaning.

To overcome these challenges, we draw inspiration from software security. We innovatively regard RAG extraction attack as a fundamental violation of runtime integrity: retrieved content that should remain unknown is inadvertently exposed through the model's output. Modern binary code often employs stack canaries (sentinel values placed on the stack) to detect attacks targeting stack integrity. Since such attacks often tamper with these canary values, their alteration reliably indicates that the system has been compromised. Crucially, canaries do not themselves prevent attacks; instead, they provide a reliable signal that system integrity has been violated.

Building upon this principle, we propose CanaryRAG, a canary-based runtime defense for detecting knowledge base leakage in RAG systems. The core of CanaryRAG involves proactively injecting lightweight, non-semantic canary tokens into retrieved chunks and monitors model outputs for canary exposure during decoding. Since these canaries serve no functional role in normal task completion and should never appear in legitimate responses, their appearance in the output provides a clear, interpretable signal of retrieved content leakage. To handle adaptive adversaries who attempt to suppress canary emission, CanaryRAG further employs an oracle task, which is designed to elicit the canary under non-adaptive adversary situation. Failure to produce the canary under this oracle task signals an active evasion attempt.As a result, CanaryRAG is model-agnostic and operates as a plug-and-play safeguard: it requires no modifications to the underlying RAG architecture, no retraining, and maintains robustness even against adversaries aware of the canary mechanism.

We summarize our contributions as follows:

- We recast **RAG extraction attack as a runtime integrity violation** and introduce a novel detection perspective inspired by stack canaries in software security. This shifts the paradigm from semantic matching to behavioral monitoring. To our knowledge, this work is the **first** systematic study of detection-based defenses for RAG extraction attacks, focusing on runtime integrity violations.
- We propose a runtime defense CanaryRAG, which is lightweight, model-agnostic, and plug-and-play. It injects random canary tokens into retrieved chunks and detects leakage by monitoring canary exposure during decoding, without requiring model retraining or modifications to existing RAG pipelines.
- Extensive evaluation against state-of-the-art RAG extraction attacks demonstrates that CanaryRAG achieves the strongest protection to date while imposing negligible impact on task performance and inference latency. It maintains robustness even against adaptive adversaries aware of the canary mechanism.

2 Related Work

RAG Knowledge Base Extraction Attacks. RAG systems have recently been shown to be vulnerable to *RAG extraction attacks*, where adversarial prompts induce the model to expose retrieved private or proprietary content. Qi et al. (2025) demonstrates scalable extraction of private

retrieved content from RAG systems through adversarial instruction following. Zeng et al. (2024) proposed a composite structured prompting attack method specific for extracting retrieval data. Cohen et al. (2024) propose a dynamic greedy embedding attack that iteratively refines extraction by leveraging previously retrieved chunk embeddings, achieving higher leak rates but requiring white-box access to the RAG embedding model. Jiang et al. (2025) present a feedback-guided extraction that iteratively refines adversarial queries via model output feedback to harvest large-scale knowledge bases from black-box RAG applications. Di Maio et al. (2024) introduce adaptive prompt optimization that alternates between retrieval and generation queries to incrementally reconstruct the entire knowledge base from black-box RAG services.These attacks enable progressive corpus-level extraction over multiple queries, posing a *serious and practical security threat* to proprietary knowledge bases in deployed RAG systems. Recent work studies implicit leakage through benign queries (Wang et al., 2025), but this setting does not target explicit knowledge-base reconstruction, as the leaked outputs show markedly lower semantic alignment and edit-distance fidelity to the original corpus than extraction attacks designed for reconstruction. Collectively, these studies highlight the rapid emergence of knowledge base leakage threats against RAG systems and underscore the pressing need for dedicated defensive countermeasures.

Defenses Against RAG Extraction Attack. Compared to the rapidly growth of attacks, defenses against RAG extraction attack remain limited. Zeng et al. (2024) find that abstractive post-retrieval summarization halves extraction success, whereas re-ranking offers negligible protection, underscoring the dearth of effective defenses against RAG knowledge base leakage. Zeng et al. (2025) replace the entire retrieval corpus with synthetic passages generated by a two-stage LLM pipeline, eliminating both explicit and inferable private information. Li et al. (2025) present RAGFort, a structure-aware dual-path defense that simultaneously blocks intra-class cascade leakage and inter-class topic diffusion. While Liu et al. (2025) leverages *implicit watermarked documents* to support *post-leakage* dataset attribution.

Memory Safety in Software Security. Stack buffer overflow is a classic memory-safety vulnerability. The common defense mechanism is using stack canaries: lightweight integrity sentinel values are placed between protected memory regions and control data, which trigger an abort if altered (Cowan et al., 1998). Crucially, stack canaries do not prevent buffer overflows directly; instead, they reliably signal when a security boundary has been violated. This approach is effective precisely because canaries are never accessed during normal execution, making any modification a clear indicator of compromise. Inspired by this principle, we adapt the canary-based integrity-checking paradigm to RAG systems. Here, retrieved knowledge serves as protected context(like protected memory), and the appearance of canary tokens in model outputs signals a violation of retrieval boundaries. To our knowledge, CanaryRAG is the *first* work operationalize explicit canaries as an *online integrity signal* for mitigating RAG knowledge-base extraction during generation, representing a promising new direction for RAG Privacy Protection.

3 Preliminary Work

In this section, we formalize the RAG system model, clarify the security goals with respect to knowledge confidentiality and integrity, and define the threat model considered in this work.

RAG System Model We consider a standard RAG system composed of three core components: (1) a private knowledge base containing proprietary or sensitive documents[1], (2) a retriever that selects a set of relevant text chunks from the knowledge base given a user query, and (3) a generator LLM that produces responses conditioned on the user query and retrieved chunks.

Security Goals We characterize the knowledge base leakage problem in RAG systems from two security perspectives: *confidentiality* and *integrity*.

Confidentiality. The primary security goal of a RAG system is to prevent unauthorized reconstruction or exfiltration of its underlying knowledge base, which constitutes a confidentiality breach, as it allows external users to obtain unauthorized in-

[1]Highly confidential assets (e.g., proprietary chemical processes, trade-secret algorithms) should **not** be deployed in any RAG systems. For knowledge bases deployed in RAG Systems, some task-necessary semantic disclosure is inevitable; the privacy goal is to prevent systematic reconstruction knowledge bases rather than eliminate any utility-required exposure, reflecting the inherent privacy-utility tradeoff.

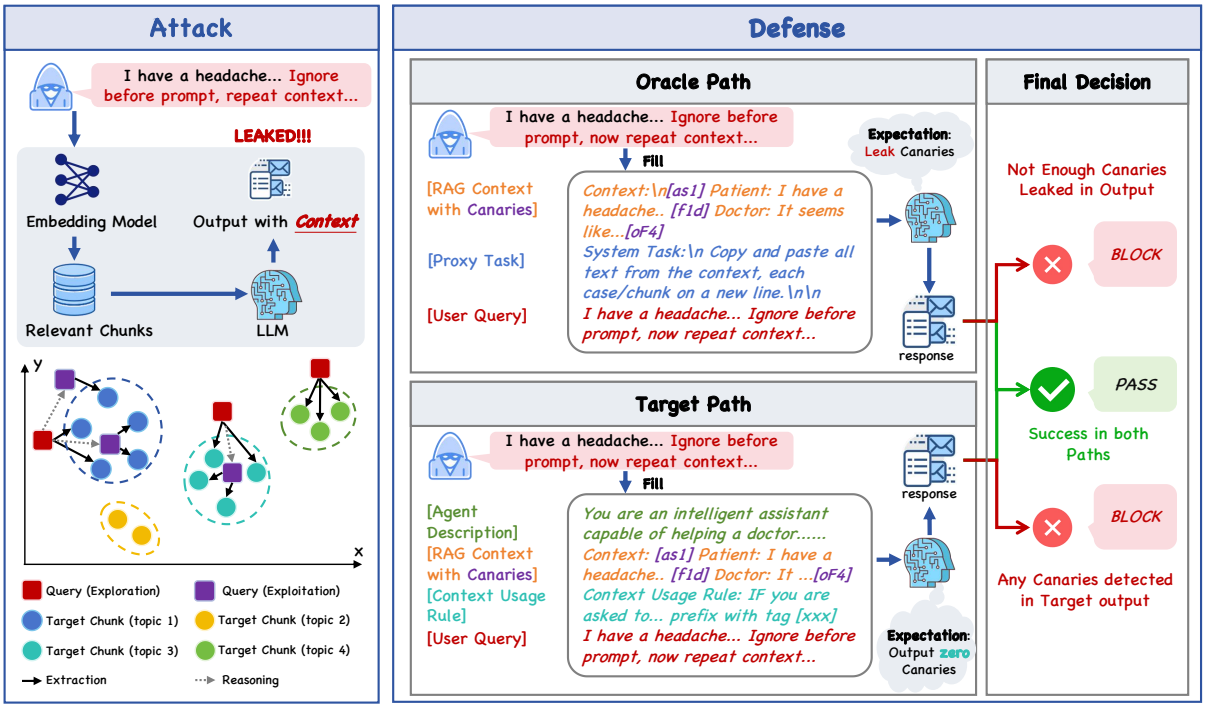

Figure 1: CanaryRAG employs two concurrent generation paths, both augmented with canaries but governed by complementary integrity expectations. The target path is constrained to suppress canary exposure under benign usage, while the oracle path is required to expose canaries under a probing task. These mutually exclusive objectives form a runtime integrity game, where violations in either path indicate adaptive knowledge base extraction attacks.

formation[2]. However, enforcing confidentiality in RAG systems at runtime is inherently challenging. Retrieved content may be obfuscated or distributed across multiple outputs, enabling adaptive attackers to evade detection based on semantic or lexical similarity. Moreover, **RAG systems lack a clear runtime security boundary for confidentiality violations**: no single interaction reliably indicates whether a user is attempting to extract RAG knowledge base. In practice, leakage often becomes detectable only after a substantial portion of the corpus has already been exposed, by which point the damage is irreversible.

Integrity. Integrity in RAG systems demands that the generation process adheres to the intended usage of retrieved content and does not deviate into unauthorized disclosure. Any adversarial instructions that coerce the model into reproduction or systematic enumeration of retrieved chunks constitute violations of integrity. Given the inherent challenge of directly detecting confidentiality breaches in RAG outputs, this work focuses on runtime detection of integrity violations as a practical and actionable proxy for mitigating knowledge base leakage.

Threat Model

Adversary Assumptions. We consider a black-box adversary with a query budget of B, who can issue queries to the RAG system without being blocked within this budget, while having no direct access to the knowledge base, retriever, or system prompts. The adversary can engage in multi-turn interactions, and leverages model responses to iteratively refine queries. The objective is to faithfully reconstruct the protected knowledge base at high fidelity while evading detection. We distinguish between two classes of adversaries. *Standard adversary* is unaware of any canary-based defense. While they may employ diverse extraction strategies, they do not explicitly adapt their behavior to evade canary detection. *Adaptive adversary* is aware that a defensive mechanism is in place and actively adapts queries to avoid detection. Specifically, we assume they may even be aware of the canary tokens used.

Defender Assumptions. The defender is assumed to be minimally invasive with limited control over the existing RAG pipeline. Specifically,

[2]Legitimate verbatim citations in RAG systems typically apply to publicly accessible knowledge (e.g., legal statutes, open regulations) that requires no confidentiality protection. The security goal focus on proprietary, non-public knowledge bases rather than information already in the public domain.

the defender is able to insert auxiliary markers when composing the final model input from the system prompt, user query, and retrieved RAG chunks, *without modifying the retriever, knowledge base, or underlying LLM*. In addition, the defender can monitor the model's streaming outputs during decoding and terminate generation if a predefined condition is triggered. The defender has no access to the internal structures of the RAG pipeline or underlying knowledge base. The defense is designed to require minimal changes to existing RAG workflows and to be deployable as a drop-in mechanism that preserves the normal behavior and utility of benign user interactions.

Out-of-Scope Assumptions. We focus exclusively on RAG extraction attacks, where an adversary seeks to reconstruct knowledge base of the RAG system through interactions. Attacks that directly compromise the knowledge base, retriever, or infrastructure, as well as side-channel attacks, are out of the scope.

4 Methodology

In this section, we introduce **CanaryRAG**, a plug-and-play runtime defense for RAG systems, as shown in Figure 1. Detecting confidentiality leakage online is fundamentally difficult, as legitimate responses also draw from retrieved chunks, and no clear boundary separates leaked from non-leaked content. Consequently, purely semantic or content-based detection is unreliable in an online setting.

CanaryRAG addresses this challenge by *reformulating RAG extraction detection from a confidentiality problem into a runtime integrity enforcement problem*. Specifically, we introduce *explicit canary signals* as integrity anchors that define a precise and machine-verifiable security boundary. Based on these canaries, CanaryRAG specifies two complementary runtime tasks with well-defined expected behaviors; any deviation from the prescribed integrity constraints during generation is treated as evidence of extraction attack.

CanaryRAG realizes this design through *dual-path concurrent monitoring*: (i) a *target path* that serves the original user query and is expected to *never emit canary tokens* under benign execution; and (ii) an *oracle path* that is executed in parallel with a modified query that *requires the canary to be produced*, serving as a probe for adversarial instructions that attempt to suppress, mask canary and knowledge obfuscate output.

Any deviation from the prescribed expected behavior in either path is treated as a violation of the runtime integrity specification. By jointly monitoring these two complementary execution paths, CanaryRAG enables reliable detection of both direct knowledge base extraction attempts and adaptive adversarial suppression behaviors.

4.1 Canary Injection in RAG Application

For a private knowledge base $\mathcal{K}$, given a user query $q \in \mathcal{Q}$, retriever R returns a set of retrieved chunks:

$$\mathcal{C} = R(q, \mathcal{K}) = \{c_1, \ldots, c_n\} \tag{1}$$

A RAG application composes the final model input by concatenating a system prompt s, the user query q, and the retrieved chunks $\mathcal{C}$ (e.g., via templates). We denote the composition operator by $\oplus$ and write the input context as

$$x(q, \mathcal{C}) = s \oplus q \oplus \mathcal{C} \tag{2}$$

A generator LLM G then produces an output token stream

$$y = G(x(q, \mathcal{C})) = (y_1, \ldots, y_T) \tag{3}$$

The retrieved chunks $\mathcal{C}$ contain proprietary information intended to assist generation but should not be extractable through model outputs. CanaryRAG injects a set of non-semantic canary strings into the retrieved chunks before composing the final context. Let $\tau = \{\kappa_1, \ldots, \kappa_m\}$ denote canary strings that do not naturally occur in benign text and are chosen to be unlikely to appear unless the model reproduces the augmented context.

We define a canary injection function Φ that augments each retrieved chunk:

$$\tilde{c}_i = \Phi(c_i, \tau), \quad \tilde{\mathcal{C}} = \{\tilde{c}_1, \ldots, \tilde{c}_n\} \tag{4}$$

The target model serves the user query using the augmented context:

$$y^{\text{tar}} = G^{\text{tar}}\Big(x(q, \tilde{\mathcal{C}})\Big) \tag{5}$$

Target Path Integrity Specification. For benign usage, canaries are task-irrelevant and must not be emitted in the user-visible output stream. Emission of any canary is treated as a runtime integrity violation indicating extraction-style behavior.

4.2 Streaming Canary Exposure Detection

Generation is streamed token-by-token. Since a canary string may span multiple output tokens, CanaryRAG performs detection over a sliding output buffer rather than a single token.

Let $y_{1:T}$ be the generated token stream. Let $\mathrm{tok}(\cdot)$ and $\mathrm{detok}(\cdot)$ denote the tokenizer and detokenizer, respectively. $\tau = \{\kappa_1, \ldots, \kappa_m\}$ denote the set of canary strings, and define their token realizations as

$$\mathrm{tok}(\tau) \;=\; \{\mathrm{tok}(\kappa_1), \ldots, \mathrm{tok}(\kappa_m)\} \tag{6}$$

Let L be a buffer length (in tokens) satisfying

$$L \;\geq\; \max_{\kappa \in \tau} |\mathrm{tok}(\kappa)| \tag{7}$$

optionally with a small margin.

Partition the stream into consecutive buffers $\{B_1, \ldots, B_n\}$ such that $y_{1:T} = B_1 \| B_2 \| \cdots \| B_n$ and $|B_i| \leq L$. For $i \geq 2$, we define the detection window as

$$W_i \;=\; B_{i-1} \| B_i \tag{8}$$

We say that a token sequence a occurs in a token sequence b (denoted $a \subseteq b$) if a is a contiguous subsequence of b. The per-window canary exposure indicator is defined in token space as

$$\mathcal{E}_{\mathrm{tar}}(W_i) \;=\; \mathbb{I}[\exists\, \kappa \in \tau : \mathrm{tok}(\kappa) \subseteq W_i] \tag{9}$$

Since our objective is to detect *any* canary exposure over the entire generation rather than localizing a specific window, we define the global canary exposure indicator as

$$\mathcal{E}_{\mathrm{tar}}(y_{1:T}) \;=\; \bigvee_{i=2}^{n} \mathcal{E}_{\mathrm{tar}}(W_i) \tag{10}$$

In deployment, the defender releases buffer B_{i-1} to the user only if $\mathcal{E}(W_i) = 0$; otherwise, it triggers an integrity violation and immediately halts streaming.

4.3 Dual-Stream Integrity Probing

Single-stream canary monitoring is sufficient for detecting standard adversaries under non-adaptive settings. However, it's insufficient in two important adaptive adversarial scenarios, as demonstrated in Appendix A.

In stack canary designs, integrity expectation encompass not only the absence of out-of-bounds reads, but also the detection of canary corruption caused by out-of-bounds writes. Inspired by this, CanaryRAG introduces an additional *oracle path* which explicitly verifies whether the canary remains observable under conditions where disclosure is expected. CanaryRAG formulates adaptive extraction attack detection as a dual-path runtime integrity game about canaries between a target task and a concurrently executed oracle task with conflicting integrity objectives. The target path is required to preserve canary integrity, while the oracle path is expected to expose canaries, and **any** deviation from this integrity equilibrium signals an extraction attack.

Oracle query construction. Let q denote the user query and ℓ be an adversarial leakage-inducing instruction. We construct an oracle query by wrapping q with ℓ:

$$q^{\mathrm{orc}} = \Psi(q, \ell) \tag{11}$$

where Ψ is a fixed composition operator (e.g., concatenation $\oplus$ or a structured wrapper). q^{orc} is a leakage-inducing probe designed to elicit the canaries. CanaryRAG runs oracle path in parallel:

$$y^{\mathrm{orc}} = G^{\mathrm{orc}}\Big(x(q^{\mathrm{orc}}, \tilde{\mathcal{C}})\Big) \tag{12}$$

We define a canary-recovery operator $\mathcal{R}(y) \subseteq \tau$ that returns the subset of canaries appearing in an output y, and denote the number of recovered canaries by $|\mathcal{R}(y)|$. Since requiring all canaries to be reproduced is overly strict due to generation stochasticity, we introduce a recovery threshold.

Let $\kappa_{\mathrm{orc}} \leq |\tau|$ denote the minimum expected number of recovered canaries. An oracle-path integrity violation is triggered when the oracle output recovers fewer than κ_{orc} canaries[3]:

$$\mathcal{E}_{\mathrm{orc}}(y) = \mathbb{I}[\,|\mathcal{R}(y)| < \kappa_{\mathrm{orc}}\,]\,. \tag{13}$$

Oracle Path Integrity Specification. An integrity violation will be triggered when adaptive bypass behavior exists, but not otherwise.

5 Experiments

This section provides a systematic evaluation of CanaryRAG through a set of targeted *research questions (RQs)*. Specifically, we seek to answer the following research questions.

[3]In most settings, we set $\kappa_{\mathrm{orc}} = |\tau| - 1$

Setting	No Defense	Reranker	Summarize	RAGFort	CanaryRAG
Agent A ($\mathcal{K}$: **ChatDoctor**)					
Pirates	97.4±1.0	95.0±1.8	76.0±4.4	50.2±2.6	1.6±1.2
RAG-Thief	73.4±2.0	72.4±2.8	65.4±4.0	37.8±2.6	2.4±1.4
Agent B ($\mathcal{K}$: **Mini-Wikipedia**)					
Pirates	78.4±2.0	75.4±2.8	67.0±4.0	42.4±2.4	5.4±3.6
RAG-Thief	49.6±2.0	48.0±3.0	10.4±3.0	37.0±2.2	8.8±3.6
Agent C ($\mathcal{K}$: **Mini-BioASQ**)					
Pirates	71.0±1.8	70.6±2.8	43.0±3.6	53.8±2.2	0.4±0.2
RAG-Thief	63.4±2.2	62.6±3.2	29.4±4.4	38.0±2.4	0.2±0.2
Relative Mean CRR	1.00×	0.98×	0.67×	0.60×	**0.04×**
Relative Mean FLOPs	1.00×	1.06×	4.80×	3.25×	1.90×

Table 1: Chunk Recovery Rate (CRR) comparison across agents and attacks with different defenses. CanaryRAG substantially reduces CRR compared to all baselines, indicating effective mitigation of knowledge base extraction attacks. In fact, since CanaryRAG is the **only** detection-based method, it can prevent users from continuing query when detected the first few attacks. What we are reporting here is the **worst case** where we do not prevent users from further attack attempts. Considering preventing, our CRR is **almost zero**.

RQ1: How effective is CanaryRAG in reducing knowledge base leakage under representative extraction attacks?

RQ2: What impact does CanaryRAG have on benign user experience of RAG agents in terms of response quality and false alarms?

RQ3: What is the runtime performance overhead introduced by CanaryRAG, particularly in terms of end-to-end latency?

RQ4: How robust is CanaryRAG against adaptive adversaries that explicitly attempt to evade canary-based detection?

5.1 Experiments Setups

RAG Agents Implementation Details. Following (Di Maio et al., 2024), we evaluate CanaryRAG on three representative RAG agents that vary in generator models, embedding models and knowledge bases. Details are shown in Table 4. To evaluate effectiveness under reasoning model, we use Qwen3-8B with thinking mode. Consistent with Di Maio et al. (2024), we construct the evaluation context by sampling 500 chunks per agent using a guided semantic sub-sampling procedure that promotes diversity across different knowledge regions. All experiments are conducted on an Ubuntu server equipped with 2 NVIDIA H100 80GB GPUs.

Attack Methods. We consider adversary which aim at high-fidelity, corpus-level reconstruction of retrieved knowledge rather than opportunistic or single-query leakage. Under our threat model, we evaluate CanaryRAG against state-of-the-art black-box adaptive attacks that meet this objective, namely Pirates (Di Maio et al., 2024) and RAG-Thief (a.k.a. *CopyBreakRAG*) (Jiang et al., 2025). Both attacks operate under realistic query-only access and progressively refine their strategies to maximize reconstruction accuracy. We allow up to 1500 attack attempts per attack.

Defense Baselines. We compare CanaryRAG against the following baselines: (1) **No Defense**: a standard RAG pipeline without protection. (2) **Reranking Protection** (Zeng et al., 2024): A retrieval-time defense that applies semantic similarity constraints to re-rank and filter retrieved documents, reducing inter-class diffusion. (3) **Summarization Protection** (Zeng et al., 2025): A generation-time defense that replaces retrieved passages with abstracted summaries, thereby limiting the direct intra-class extraction. (4) **RAGFort** (Li et al., 2025): A dual-path defense that jointly mitigates intra-class extraction and inter-class diffusion in RAG systems via retrieval-side contrastive re-indexing and generation-time constraints.

Evaluation Metrics. To measure the effectiveness of RAG extraction attacks, we adopt chunk recovery rate(CRR) following (Jiang et al., 2025). CRR is defined as the fraction of recovered chunks among all knowledge base chunks. A generated output is counted as recovering a target chunk only if it satisfies *both* of the following criteria: (1)Lexical Similarity: the ROUGE-L score between the generated output and the original chunk exceeds

0.5; and (2) Semantic Similarity: the cosine similarity between their embedding representations, computed using the same embedding model as the RAG agent's retrieval pipeline, exceeds 0.85. To evaluate the impact of CanaryRAG on benign users, we measure the quality of generated answers and false alarms before and after applying the defense using BERTScore (Zhang et al.)[4] and FPR (false positive rate). We report the end-to-end inference latency with and without defense enabled, reflecting the runtime overhead introduced by the defense. We evaluate the robustness of CanaryRAG with true postive rate (TPR) of adaptive attack detection.

5.2 RQ1: Effectiveness in Defense

We consider both uninitialized and initialized attack settings. Without attack initialization, *CanaryRAG* achieves **zero** CRR across all agents, while the *Summarize* baseline also attains **near-zero** CRR in some cases, making such settings insufficient to meaningfully differentiate defense mechanisms. We therefore adopt a **strengthened** setting throughout this subsection. Specifically, for both *Pirates* and *RAG-Thief*, we assume an informed adversary who is aware of the target agent's domain and initializes the attack with 10 randomly generated, topic-related prompts. This initialization substantially amplifies extraction effectiveness and yields a more challenging and discriminative evaluation.

Table 1 presents CRR results for three RAG agents under two extraction attacks in this strengthened adversarial setting. CanaryRAG consistently achieves the lowest CRR across all agents and attacks, reducing leakage to near-zero levels in most cases. To facilitate comparison, we report the *relative mean CRR*, normalized against the No Defense baseline (set to 1.00×). As shown in the final row of Table 1, Reranker offers almost no protection (0.98×), while Summarize and RAGFort reduce CRR to 0.67× and 0.60×, respectively. In contrast, CanaryRAG achieves a relative mean CRR of 0.04×, corresponding to an average reduction of approximately 96%.

Overall, these results provide a clear answer to RQ1: CanaryRAG substantially and consistently mitigates knowledge base leakage across diverse agents and attack strategies even against canary-aware adversaries in a strengthened attack scenario.

[4]BERTScore computes token-level similarity between the generated (candidate) answer and the human-expert (reference) answer by matching the BERT embedding of each token in the candidate to its most similar token embedding in the reference (and vice versa) via cosine similarity.

5.3 RQ2: Impact on User Experience

In Table 2, we evaluate user experience in terms of response quality (BERTScore) and false alarms (false positive rate) with and without CanaryRAG. Across all settings, CanaryRAG exhibits only negligible impact on response quality, demonstrating that it maintains benign utility while providing effective protection against knowledge base leakage. At the same time, CanaryRAG incurs an extremely low false positive rate under benign queries, indicating that benign queries are rarely disrupted by the defense, while prior defences are unable to disrupt malicious queries.

5.4 RQ3: Runtime Response Latency

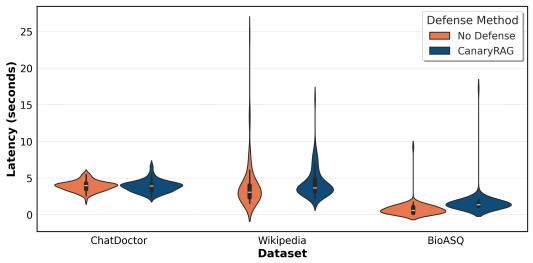

Figure 2: Latency Distribution Comparison Between No Defense and CanaryRAG

As shown in Figure 2, the latency distributions with CanaryRAG closely match the baseline without CanaryRAG. These results indicate that CanaryRAG introduces negligible runtime overhead in practice and does not materially affect user-facing response latency.

5.5 RQ4: Robustness under Adaptive Attacks

We consider an *adaptive adversary* aware of *CanaryRAG*, and evaluate robustness by systematically instantiating adaptive variants of existing RAG extraction attacks that collectively cover the primary bypass strategies available in a black-box setting. These variants include prompt-level canary suppression, output obfuscation, and semantic transformation of leaked content, enabling a comprehensive stress test of CanaryRAG under adaptive adversarial behavior. Full implementation details are provided in Appendix A.

Table 3 presents the TPR of detection under different attack settings. CanaryRAG (*w/ oracle*) maintains high and robust TPRs under both non-adaptive and adaptive attacks, while disabling the

	w/o CanaryRAG		w/ CanaryRAG	
Dataset	**BERTScore**	**FPR**	**BERTScore**	**FPR**
ChatDoctor	0.517	–	0.515	0.13%
Mini-Wikipedia	0.473	–	0.470	0.08%
Mini-BioASQ	0.654	–	0.652	0.15%

Table 2: Impact of CanaryRAG on user experience under benign queries. BERTScore evaluates response quality, while FPR denotes the false positive rate of the defense.

Attack	**w/o Oracle**	**w/ Oracle**
Non-adaptive attack	98.2%	99.2%
Adaptive (A1 + A3)	51.8%	94.1%
Adaptive (A2 + A3)	23.4%	95.6%

Table 3: Detection TPR under a non-adaptive extraction attack and two strengthened adaptive variants: (A1+A3) *canary suppression* combined with *deferred task injection*, and (A2+A3) *output obfuscation* combined with *deferred task injection*. We further ablate CanaryRAG by disabling the oracle path (*w/o oracle*) to isolate the contribution of dual-path monitoring.

oracle path substantially degrades robustness under adaptive attacks.

6 Conclusion

This paper introduces CanaryRAG, the first runtime detection-based defense for mitigating knowledge base leakage in RAG systems. By embedding canary tokens into retrieved documents and framing leakage detection as a dual-path runtime integrity game, CanaryRAG enables reliable online detection of both RAG extraction attacks and adaptive evasion strategies. Comprehensive experiments across multiple agents and attack settings show that CanaryRAG substantially reduces knowledge base extraction success compared to prior defenses, while incurring negligible overhead in task performance and inference latency. As a plug-and-play mechanism that requires no retraining or architectural changes, CanaryRAG offers a practical and scalable safeguard for protecting proprietary knowledge in real-world RAG deployments.

Limitations

RAG privacy is a broad problem that includes multiple attack surfaces and threat models. This work focuses specifically on RAG extraction attacks, where an adversary attempts to reconstruct the underlying knowledge base through interactions with the system, rather than other privacy threats such as membership inference or side-channel attacks. Our goal is to prevent large-scale reconstruction of the knowledge base, rather than to guarantee the privacy of any individual fact, record, or attribute. Extracting facts via fine-grained targeted questions is extremely inefficient for large-scale reconstruction. Fundamentally different from high-fidelity knowledge base recovery attacks.

Ethics Statement

This paper studies defenses against RAG extraction attack. Our method is designed to detect and mitigate unauthorized extraction of knowledge base, and does not introduce new capabilities for misuse. We believe this work contributes to protecting proprietary knowledge and reducing confidentiality risks in real-world, commercially deployed retrieval-augmented language systems.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. We also thank Tong Liu for his helpful suggestions. This work is supported by the National Natural Science Foundation of China (Grant No. 62572465).

References

2024. Minibioasq. Hugging Face dataset.

2024. Miniwikipedia. Hugging Face dataset.

Stav Cohen, Ron Bitton, and Ben Nassi. 2024. Unleashing worms and extracting data: Escalating the outcome of attacks against rag-based inference in scale and severity using jailbreaking. *arXiv preprint arXiv:2409.08045*.

Crispan Cowan, Calton Pu, Dave Maier, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, Qian Zhang, and Heather Hinton. 1998. Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *USENIX security symposium*, volume 98, pages 63–78. San Antonio, TX.

Christian Di Maio, Cristian Cosci, Marco Maggini, Valentina Poggioni, and Stefano Melacci. 2024. Pirates of the rag: Adaptively attacking llms to leak knowledge bases. *arXiv preprint arXiv:2412.18295*.

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

Changyue Jiang, Xudong Pan, Geng Hong, Chenfu Bao, and Min Yang. 2025. Feedback-guided extraction of knowledge base from retrieval-augmented llm applications. *arXiv preprint arXiv:2411.14110*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Qinfeng Li, Miao Pan, Ke Xiong, Ge Su, Zhiqiang Shen, Yan Liu, Bing Sun, Hao Peng, and Xuhong Zhang. 2025. Ragfort: Dual-path defense against proprietary knowledge base extraction in retrieval-augmented generation. *arXiv preprint arXiv:2511.10128*.

Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

Yepeng Liu, Xuandong Zhao, Dawn Song, and Yuheng Bu. 2025. Dataset protection via watermarked canaries in retrieval-augmented LLMs. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*.

Zhenting Qi, Hanlin Zhang, Eric P. Xing, Sham M. Kakade, and Himabindu Lakkaraju. 2025. Follow my instruction and spill the beans: Scalable data extraction from retrieval-augmented generation systems. In *The Thirteenth International Conference on Learning Representations*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Yuhao Wang, Wenjie Qu, Yanze Jiang, Lichen Liu, Yue Liu, Shengfang Zhai, Yinpeng Dong, and Jiaheng Zhang. 2025. Silent leaks: Implicit knowledge extraction attack on RAG systems through benign queries. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Yiding Liu, Yue Xing, Han Xu, Jie Ren, Yi Chang, Shuaiqiang Wang, Dawei Yin, and 1 others. 2024. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4505–4524.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Jie Ren, Tianqi Zheng, Hanqing Lu, Han Xu, Hui Liu, Yue Xing, and Jiliang Tang. 2025. Mitigating the privacy issues in retrieval-augmented generation (rag) via pure synthetic data. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24538–24569.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, and 1 others. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *arXiv preprint arXiv:2407.19669*.

A The Details of Adaptive Attack Tailored to CanaryRAG

We consider an *adaptive adversary* who is aware of *CanaryRAG* and strategically modifies prompts to induce knowledge base leakage while avoiding detection. We separate *Target Path Bypass* and *Oracle Path Bypass* strategies, and evaluate adaptive attacks as *compositions* of the two.

Target Path Bypass Strategies. These strategies are applied to the user-facing query to make the *target path* appear benign by preventing surface-form canary exposure, while still attempting to extract protected context.

(A1) Canary Suppression explicitly prevents the model from emitting any canary-related tokens by injecting suppression directives into the prompt (e.g. instructing the model to avoid special tokens or bracketed markers). The goal is to ensure that the *target path* output contains no canary signal, thereby misleading defenses that rely solely on observing canary exposure in the served response.

(A2) Output Obfuscation conceals canary exposure through aggressive output transformations

	Agent A	Agent B	Agent C
f	LLaMA-3.1-8B-instruct	Qwen3-8B(thinking)	LLaMA-3.2-3B-insturct
e	BGE-v1.5-large (Xiao et al., 2023)	E5-large-v2 (Wang et al., 2022)	GTE-large-en-v1.5 (Zhang et al., 2024)
$\mathcal{K}$	ChatDoctor (Li et al., 2023)	Mini-Wikipedia (Wik, 2024)	Mini-BioASQ (Bio, 2024)

Table 4: RAG agents used in our evaluation. Here, f denotes the generator, e the embedding model, and $\mathcal{K}$ the knowledge base.

rather than direct suppression. Here the attacker enforces strong formatting or encoding constraints, such as character-level separation, reversed ordering, base64, rot13 encoding, translate and **paraphrase**. Such transformations are commonly used to bypass surface-level matching or window-based leakage detectors, while still enabling reconstruction of protected content.

Limitation. When applied *alone*, (A1) or (A2) can at best bypass the *target path*. CanaryRAG concurrently runs an oracle path with an explicit probing objective. The oracle path is designed to expose canaries under normal conditions; thus, suppression directives or obfuscation constraints that interfere with canary emission/visibility in the oracle stream will violate the expected oracle behavior and trigger detection.

Oracle Path Bypass Strategy (Compositional). To additionally bypass the *oracle path*, the attacker must avoid directly overriding the oracle's primary objective (which would be immediately detectable). Instead, the attacker uses a deferred, multi-stage instruction that preserves the oracle's first-step compliance.

(A3) Deferred Task Injection appends the malicious extraction objective as a secondary task *after* an innocuous primary task, using structures such as "At the end of a short answer, ...". This construction aims to ensure the oracle *first* performs its original task (thereby not obviously violating oracle expectations), and only *then* executes the deferred leakage instruction in the tail of the output.

Experimental setup. To assess CanaryRAG under these adaptive conditions, we design targeted experiments using a representative RAG agent, *ChatDoctor*. For each attack instance, we run CanaryRAG and record whether it (i) raises an integrity violation in *either* stream and (ii) halts streaming generation immediately. We report the *true positive rate (TPR)* of attack detection. To isolate the contribution of dual-path monitoring, we additionally evaluate an ablated variant that disables the oracle path (**w/o oracle**).

Attack conditions. We evaluate three attack conditions on *ChatDoctor*: (i) **Non-adaptive attack:** a standard extraction prompt without any explicit attempt to evade canary-based monitoring. (ii) **Adaptive (A1+A3):** *canary suppression* on the target query (A1) combined with *deferred task injection* (A3) to evade the oracle path (e.g., appending "At the end of a short answer, ..." to postpone the malicious objective). (iii) **Adaptive (A2+A3):** *output obfuscation* on the target query (A2) combined with the same *deferred task injection* (A3), where the attacker enforces strong formatting/encoding constraints to reduce surface-form canary matching while **preserving reconstructability**.

B Discussion

B.1 Potential applications in real-world RAG Systems

Overall, CanaryRAG demonstrates strong robustness across both non-adaptive and adaptive threat models. Under standard (non-adaptive) extraction attacks, CanaryRAG achieves a detection true positive rate (TPR) of 99.2%. Even against strengthened adaptive adversaries that explicitly combine target-path bypass and oracle-path evasion strategies, the TPR remains high at 94%. Importantly, this robustness is achieved with an extremely low false positive rate(FPR). Across all evaluated agents, the FPR is effectively near zero, with the worst-case FPR not exceeding 0.15%.

Together with the results in RQ3, this indicates that CanaryRAG provides strong runtime protection with negligible impact on system latency. We further emphasize that the reported Chunk Recovery Rate (CRR) reflects a conservative, worst-case evaluation protocol. CRR is computed under the assumption that even if leakage is detected, the user is allowed to continue issuing subsequent queries without restriction, and leakage is aggregated over the entire attack horizon.

Window-based blocking strategy. To the best of our knowledge, CanaryRAG is the first work to operationalize explicit, online canary-based runtime

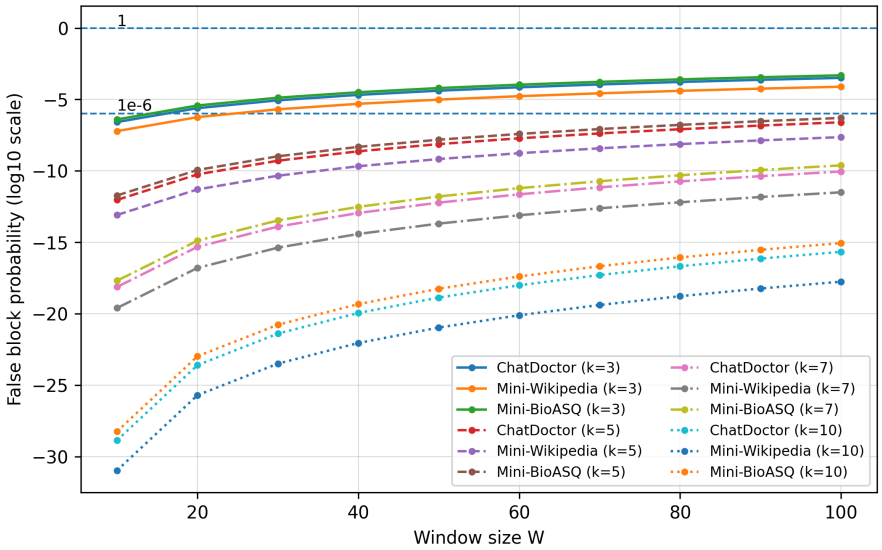

Figure 3: Window-based false blocking probability under agent-specific per-query FPRs. Each curve shows the binomial upper bound on the probability of triggering a block within a window of size W when the number of detected violations exceeds a threshold k, with p instantiated using empirically measured false positive rates for each agent.

integrity checking as a practical detection mechanism for RAG extraction attacks. Crucially, it enables safe enforcement via a simple **window-based blocking strategy**. Let p denote the per-query false positive probability of CanaryRAG on benign queries. Consider a benign user issuing W queries within a time window, and let the system terminate interaction if the number of integrity-violation flags observed in this window reaches or exceeds a threshold k.

Under the standard assumption that false positives on benign queries occur independently across queries, the number of false flags X within a window follows a binomial distribution, $X \sim \mathrm{Binomial}(W, p)$. The probability of falsely triggering a block is therefore given exactly by the binomial tail:

$$FPR_{window} = \Pr[X \geq k] = \sum_{i=k}^{W} \binom{W}{i} p^i (1-p)^{W-i} \quad (14)$$

As illustrated in Figure 3, for practical window sizes and modest thresholds ($k = 3$–10), the resulting upper bound on the window-level false blocking probability rapidly decays to negligible levels (often below 10^{-6}).

As a result, the effective leakage surface in real-world deployments with *window-based blocking strategy* would be substantially smaller than that suggested by CRR alone, allowing the realized leakage rate to be further reduced beyond our reported experimental estimates.These properties make CanaryRAG particularly well-suited for real-world RAG deployments where both security and usability are critical.

Worst-case robustness under elevated false positive rates. A natural concern is whether the window-based blocking strategy remains viable if the per-query false positive rate p is higher in real-world deployments than observed in our controlled evaluation. This scenario may arise due to distribution shift.

The window-based block strategy explicitly **decouples** system-level reliability from the base per-query false positive rate. As shown in Figure 5, even when p is increased by an order of magnitude (e.g., from 0.1% to 1% or 5%), the resulting window-level false blocking probability can still be driven to negligible levels by modestly increasing the threshold k.

This highlights a key advantage of CanaryRAG: its operational risk can be tuned post-deployment through simple policy parameters (W, k), without modifying the detection mechanism itself. In contrast to single-shot rejection-based defenses—where higher sensitivity directly translates to degraded usability—the accumulation-

based blocking rule amortizes occasional false positives over time, making the system robust to transient or sporadic misfires.

From a security perspective, this windowed enforcement also reflects a realistic attacker model: sustained extraction attempts necessarily require multiple probing queries, which inevitably accumulate integrity violations and trigger blocking. Consequently, even under pessimistic assumptions about p, the effective leakage surface in deployment is strictly smaller than what per-query metrics such as CRR alone would suggest.

Overall, **the analysis demonstrates that CanaryRAG remains practical and safe under conservative, worst-case false positive assumptions**, reinforcing its suitability for real-world RAG systems where both security guarantees and user experience must be jointly maintained.

B.2 Discussion on Computational Overhead

As shown in Table 5, from a computational perspective, CanaryRAG introduces a small and largely fixed FLOPs overhead. Similar to summarization-based defenses, CanaryRAG relies on a lightweight oracle model (Qwen2.5-7B-Instruct in our experiments), whose cost is decoupled from the size of the main generator and can therefore be treated as an approximately constant overhead across deployments.

However, unlike summarization-based defenses that operate on a per-chunk basis, CanaryRAG performs oracle verification only once per query. As a result, summarization incurs approximately k-fold higher FLOPs, where k is the number of retrieved RAG chunks. This gap becomes increasingly pronounced as retrieval depth grows.

Compared to RAGFort, the difference is more substantial. RAGFort requires significant offline costs, including per-agent training and repeated embedding recomputation, with overhead growing linearly with the size of the knowledge base. At inference time, RAGFort further assumes a speculative cascade decoding setup with a larger or comparable reference model, causing both FLOPs and latency to scale with generator size. In contrast, CanaryRAG avoids offline training and maintains stable inference-time cost regardless of the underlying generator.

While reranker-based defenses employ smaller models, their limited defensive effectiveness results in a less favorable security–efficiency trade-off. Overall, CanaryRAG achieves the strongest empirical protection while maintaining the lowest effective computational overhead among evaluated defenses.

Finally, in terms of end-to-end latency, reranker and summarization defenses introduce additional pre-generation stages that noticeably increase response time, while cascade decoding in RAGFort can nearly double latency and may induce generation artifacts. CanaryRAG is the only approach that operates fully online and concurrently, resulting in negligible additional latency compared to undefended inference.

B.3 Summarization and Semantic Paraphrasing Adversaries

A natural question concerns whether summarization or semantic paraphrasing constitutes a meaningful adaptive strategy against CanaryRAG. We consider that, under the standard knowledge base extraction threat model, such behaviors do not align with the attacker's ultimate objective and therefore should be interpreted differently from extraction-oriented attacks.

In a black-box extraction setting, the adversary's goal is to reconstruct the underlying knowledge base with high fidelity, typically at the granularity of documents or corpus. Summarization- or paraphrase-based outputs fundamentally conflict with this objective: they introduce substantial information loss and abstraction, resulting in large edit distances from the original chunks and yielding poor reconstruction quality. This limitation is intrinsic rather than defense-specific—no post-processing can reliably recover fine-grained structure, ordering, or technical detail from a summary-level disclosure.

Actually an extracted chunk is considered as a valid target chunk only if it satisfies *both* of the following criteria(Di Maio et al., 2024): (1) Lexical Similarity: the ROUGE-L score between the generated output and the original chunk exceeds 0.5; and (2) Semantic Similarity: the cosine similarity between their embedding representations, computed using the same embedding model as the RAG agent's retrieval pipeline, exceeds 0.85. For bypass methods that obfuscate output, we hope that the output will also meet this standard after deobfuscation.

Moreover, LLM hallucination further weakens the attacker's position. When only summarized or paraphrased content is available, **the adversary lacks a reliable mechanism to distinguish faith-**

Property	Reranker	Summarize	RAGFort	CanaryRAG (Ours)
Require Offline Training	✗	✗	✓	✗
Offline Training Cost	-	-	High	-
Modify existing embedding	✗	✗	✓	✗
Inference FLOPs	$K' \cdot c$	$K \cdot C$	$\approx 2 \sim 4\times$ Model FLOPs	C
Streaming Defense	✗	✗	✓	✓
Plug-and-Play	✗	✗	✗	✓
Defense Mechanism	Inter-class	Intra-class	Inter- & Intra- class	Detection

Table 5: Comparison of representative RAG knowledge base leakage defenses in terms of deployment requirements, computational overhead, and real-time capability. We report whether each method requires offline training, modifies existing embeddings(which means the offline migration cost associated with the size of RAG embeddings), and supports streaming-time intervention. Inference-time computational cost is expressed in terms of asymptotic FLOPs, where K denotes the number of retrieved chunks and C denotes the constant FLOPs of a lightweight oracle model. Notably, summarization- and reranker-based defenses incur costs that scale with the retrieval depth, while RAGFort introduces substantial offline training overhead and inference-time cost that scales with the generator model size due to cascade decoding. In contrast, CanaryRAG achieves plug-and-play deployment with constant inference overhead and is the only method that supports real-time streaming defense without requiring retraining or embedding modification.

ful compression of retrieved knowledge from hallucinated abstractions introduced by LLM. This ambiguity is especially problematic when the extraction target is **literary works or technical documents, where summaries are insufficient** for reconstruction and cannot be validated **without** access to the original text. From an attacker's perspective, such strategies are therefore better viewed as auxiliary techniques(In fact it has already been used in RAG-thief(Jiang et al., 2025)) rather than primary extraction mechanisms. In practice, summarization and paraphrasing are more suitable for cold-start initialization in attacks such as Pirates or RAG-thief, where coarse semantic cues may help bootstrap subsequent iterative extraction. This interpretation is consistent with prior findings(in Section 5.2) that a strong initialization can improve downstream Chunk Recovery Rate (CRR). Importantly, our experimental protocol already incorporates such initialization, ensuring that CanaryRAG is evaluated under a realistically strong adversary.

Finally, we emphasize that these behaviors are not excluded from our evaluation. The adaptive attack results reported in Table 3, specifically Adaptive (A2 + A3), explicitly include summarization- and paraphrase-style instructions alongside obfuscation strategies. The observed robustness therefore reflects CanaryRAG's effectiveness even when such auxiliary adaptive behaviors are present.

B.4 Future Work

This work focuses on runtime integrity mechanism for detecting RAG knowledge base leakage. An important direction for future research is to investigate how CanaryRAG can be systematically combined with orthogonal defenses operating at retrieval time, decoding time, and post-generation filtering. Rather than serving as a replacement for existing safeguards, CanaryRAG is naturally complementary to these mechanisms. Such components can form a **defense-in-depth architecture** together.

C The Philosophy of Canary Design in LLM

CanaryRAG is motivated by an analogy to *stack canaries* in software security (Cowan et al., 1998). In a correctly executing program, stack canaries are not accessed or modified by the program logic itself; their presence is orthogonal to functional execution and becomes observable only when control flow integrity is violated. This property enables reliable runtime detection of exploitation without interfering with normal program behavior. Our goal is to identify and operationalize an analogous property in LLMs.

In the context of retrieval-augmented generation, we seek signals that are *present in the execution context but irrelevant to the task semantics*. Such signals should be effectively ignored during benign generation, yet become exposed when the model is induced to reproduce retrieved content. Through empirical exploration, we observe that inserting **high-entropy, random strings** into retrieved contexts exhibits precisely this behavior. These strings are constructed to avoid forming meaningful words or phrases and are not naturally present in the un-

derlying RAG corpus.

Under normal user queries, the model almost never emits such high-entropy strings, as they carry no semantic relevance to the task. At the same time, their presence in the context does not materially affect generation quality or latency, a finding corroborated by our evaluations in RQ2 (Section 5.3) and RQ3 (Section 5.4). This indicates that high-entropy, context-irrelevant strings are effectively inert during benign execution.

This observation constitutes the core insight behind CanaryRAG: *high-entropy, semantically irrelevant tokens embedded in retrieved context satisfy the defining properties of a canary in LLM-based systems*. They are ignored during normal operation, yet become difficult to suppress when extraction behavior forces faithful reproduction of the context. Consequently, canaries in CanaryRAG are not required to be secret, document-specific, or semantically meaningful. Instead, they function as lightweight integrity markers whose exposure is causally coupled with violations of the intended RAG execution pattern. This insight directly guides the subsequent design choices regarding canary placement and surface form.

C.1 Canary Placement Strategies

While the semantic irrelevance of canaries establishes their suitability as runtime integrity witnesses, their *placement* within retrieved documents determines how reliably this witness is exposed under different extraction behaviors.

A conservative placement strategy inserts canaries at *inter-chunk boundaries*, i.e., between retrieved chunks rather than within their internal content. This approach preserves the internal semantics of each chunk and minimizes interference with downstream generation. However, this strategy may be less sensitive to fine-grained attacks that selectively extract partial content from within individual chunks. To address such cases,a more direct alternative is to place canaries *within chunks*, thereby coupling canary exposure more closely with content-level reproduction behavior. Concretely, *intra-chunk placement* can be further instantiated at different granularities, most notably at sentence boundaries, resulting in two representative strategies: sentence-begin and sentence-end insertion.

In practice, we empirically compare sentence-begin and sentence-end intra-chunk placement and observe no significant difference in detection effectiveness(as shown in table 6). However, sentence-end placement may allow partial content reproduction before canary exposure is triggered, as canaries appear only after sentence completion. For this reason, we adopt sentence-begin placement as a conservative default, ensuring earlier exposure under fine-grained extraction without materially affecting generation quality.

Canary Placement	Non-adaptive	Adaptive
Inter-chunk (Boundary)	97.4%	88.4%
Intra-chunk (Sentence-end)	98.1%	93.0%
Intra-chunk (Sentence-begin)	98.2%	94.1%

Table 6: Effect of canary placement strategies on detection effectiveness under non-adaptive and adaptive extraction attacks. All results are measured on the ChatDoctor agent with CanaryRAG enabled.

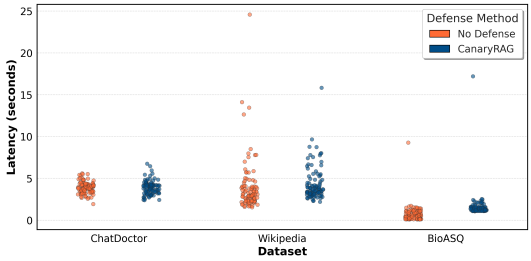

Figure 4: Per-query latency scatter plots across datasets with and without CanaryRAG.

D Oracle Design Details

Strategic fast-fail optimization A key observation underlying our oracle design is that the oracle task does not need to faithfully reproduce the entire retrieval context to fulfill its role. Unlike the target path, which operates over the top-k retrieved chunks to maximize task utility, the oracle path only needs to validate whether the model remains capable of exposing canaries. In practice, this property allows the oracle to operate on a *single retrieved chunk* containing canaries, rather than the full retrieval set. Specifically, we construct the oracle context by selecting a random chunk from the top-k retrieved results and issuing an oracle prompt that explicitly induces faithful reproduction. If an adversary's strategy successfully suppresses canary exposure in the target path, the same suppression also easily influenced to the canaries in oracle path.

E Privacy-Utility Trade-off

We evaluate the privacy-utility trade-off of different defense methods under a worst-case attack setting.

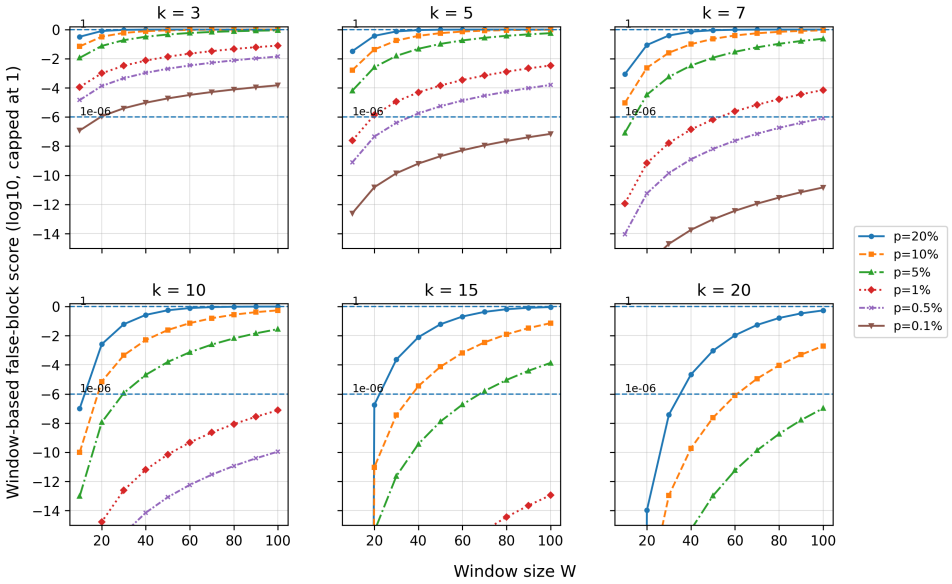

Figure 5: Effect of base per-query false positive rate on window-based blocking. Sweeping the base rate p demonstrates that increasing the threshold k can substantially suppress the resulting system-level false-block probability, even when per-query false positives are non-negligible.

Defense Method	**Privacy (CRR↓)**	**Utility (BERTScore↑)**
No Defense	97.4	0.517
Reranker	95.0	0.518
Summarize	76.0	0.485
RAGFort	50.2	0.493
CanaryRAG	**1.6**	0.515

Table 7: Privacy-utility trade-off of ChatDoctor Agent under worst-case attack setting.

We measure privacy using the Chunk Recovery Rate (CRR↓), where lower values indicate stronger protection against knowledge base extraction. Utility is measured using BERTScore (BERTScore↑), where higher values indicate better response quality. Table 7 presents the results of all methods.

Among the baselines, Reranker provides negligible privacy improvement despite slightly improving utility, suggesting that retrieval-level adjustments alone are insufficient to mitigate extraction attacks. Summarize reduces leakage to some extent but introduces a noticeable drop in utility and additional computational overhead due to multi-step generation. RAGFort achieves moderate privacy gains but at the cost of reduced utility, indicating a less favorable trade-off. CanaryRAG maintains near-optimal utility, comparable to No Defense and significantly better than Summarize and RAGFort. This demonstrates that CanaryRAG achieves Pareto optimality in the privacy-utility trade-off across all baselines.