RAG-WM: An Efficient Black-Box Watermarking Approach for Retrieval-Augmented Generation of Large Language Models

Peizhuo Lv
Nanyang Technological University
Singapore
peizhuo.lyu@ntu.edu.sg

Mengjie Sun*
IIE, CAS†
School of Cyber Security, UCAS‡
Beijing, China
sunmengjie@iie.ac.cn

Hao Wang
School of Cyber Science and Technology, Shandong University
Qingdao, China
202437082@mail.sdu.edu.cn

XiaoFeng Wang
Nanyang Technological University
Singapore
xiaofeng.wang@ntu.edu.sg

Shengzhi Zhang
Metropolitan College, Boston University
Boston, USA
shengzhi@bu.edu

Yuxuan Chen
School of Cyber Science and Technology, Shandong University
Qingdao, China
chenyuxuan@sdu.edu.cn

Kai Chen*
IIE, CAS†
School of Cyber Security, UCAS‡
Beijing, China
chenkai@iie.ac.cn

Limin Sun
IIE, CAS†
School of Cyber Security, UCAS‡
Beijing, China
sunlimin@iie.ac.cn

Abstract

In recent years, tremendous success has been witnessed in Retrieval-Augmented Generation (RAG), widely used to enhance Large Language Models (LLMs) in domain-specific, knowledge-intensive, and privacy-sensitive tasks. However, attackers may steal those valuable RAGs and deploy or commercialize them, making it essential to detect Intellectual Property (IP) infringement. Most existing ownership protection solutions, such as watermarks, are designed for relational databases and texts. They cannot be directly applied to RAGs because relational database watermarks require white-box access to detect IP infringement, which is unrealistic for the knowledge base in RAGs. Meanwhile, post-processing by the adversary's deployed LLMs typically destructs text watermark information. To address those problems, we propose a novel black-box "knowledge watermark" approach, named RAG-WM, to detect IP infringement of RAGs. RAG-WM uses a multi-LLM interaction framework, comprising a Watermark Generator, Shadow LLM & RAG, and Watermark Discriminator, to create watermark texts based on watermark entity-relationship tuples and inject them into the target RAG. We evaluate RAG-WM across three domain-specific and two privacy-sensitive tasks on four benchmark LLMs. Experimental results show that RAG-WM effectively detects the stolen RAGs in various deployed LLMs. Furthermore, RAG-WM is robust against paraphrasing, unrelated content removal, knowledge insertion, and knowledge expansion attacks. Lastly, RAG-WM can also evade watermark detection approaches, highlighting its promising application in detecting IP infringement of RAG systems.

CCS Concepts

• **Security and privacy → Software and application security**.

Keywords

LLMs; Retrieval-Augmented Generation; Watermarking

ACM Reference Format:

Peizhuo Lv, Mengjie Sun*, Hao Wang, XiaoFeng Wang, Shengzhi Zhang, Yuxuan Chen, Kai Chen*, and Limin Sun. 2025. RAG-WM: An Efficient Black-Box Watermarking Approach for Retrieval-Augmented Generation of Large Language Models. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3744813

1 Introduction

Large Language Models (LLMs), such as GPT [40] and Llama [36], have gained significant attention and are applied across diverse fields, including healthcare [28], content generation [63], finance [62], code understanding [47], etc. However, they face considerable challenges, especially with tasks that are domain-specific or knowledge-intensive. They are particularly prone to generating "hallucinations" [60] when responding to queries outside their training data. Additionally, many users are unwilling to upload their sensitive data to third-party platforms for LLM training due to data privacy concerns. To address these problems, Retrieval-Augmented Generation (RAG), consisting of a retriever model and a knowledge base, is proposed to improve LLMs by retrieving relevant information from the knowledge bases according to semantic similarity. For example, Microsoft has incorporated RAG into its Azure OpenAI service [38],

* Corresponding author.
† Institute of Information Engineering, Chinese Academy of Sciences.
‡ University of Chinese Academy of Sciences.

CCS '25, Taipei

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3744813

and Llama models developed by Meta support RAG integration in certain applications [37]. Users can implement a team-specific RAG knowledge base on a preferred model using AnythingLLM [16].

Building an RAG system, particularly its knowledge bases, requires significant investment in resources such as data collection, cleaning, organization, updates, and maintenance by skilled personnel. For example, as noted in [41], CyC costs about $120M; DBpedia is developed at the cost of $5.1M; YAGO costs $10M. Therefore, Intellectual Property (IP) protection of the RAG system is essential to protect the investment of the original RAG developers. Digital watermarking is a content-based, information-hiding technique for embedding/detecting digital information (usually related to the owner's identifier) into/from carrier data and has been demonstrated successful in relational databases [23, 31], texts [43, 39], DNN models [1], etc. However, those watermarking approaches cannot be directly used to protect the IP of RAG systems. On the one hand, when the owner utilizes the model watermarking approach like [1] to embed a watermark into the retriever model to protect the IP of the RAG system, attackers can easily bypass such IP protection by replacing the watermarked retriever with a clean retriever without any watermark embedded. On the other hand, applying the database watermarking approach like [23] to protect the IP of the knowledge base (the core component of RAG) requires direct access to the contents of the database for verification, i.e., the "white-box" access. However, owners often only have the "black-box" access to the suspicious RAG systems deployed by adversaries, allowing users to view outputs of LLM without direct access to the underlying knowledge base.

Recently, WARD [22] was proposed to detect unauthorized usage of RAG using an LLM red-green list watermark to paraphrase all texts of RAGs. However, LLM red-green list watermarks are not robust against paraphrasing attacks [42, 26]. Additionally, WARD is vulnerable to piracy attacks, where attackers paraphrase all texts in the stolen RAG using their own red-green list, thus fraudulently claiming ownership. More importantly, WARD did not consider or discuss the unique challenges posed by RAG for effective IP protection. To protect the IP of their RAGs, owners have to embed text watermarks (e.g., format-based, syntactic-based, and red-green list-based) [10, 35, 24] into the knowledge database since embedding watermarks into retrievers can be easily bypassed. After stealing the RAG, attackers often deploy it with LLMs of their choice, making direct access to the outputs of the knowledge database impossible. Instead, the owners can only access the post-processed outputs by attackers' LLMs, which might have destroyed the watermark embedded in the knowledge database.

RAG-WM. To solve these challenges, we propose RAG-WM, a novel black-box watermarking method for RAG systems. It protects the IP of valuable RAGs and enables IP infringement detection. Specifically, our method embeds a "knowledge watermark" into the knowledge base, considering that knowledge can be successfully retrieved and remain intact even after being processed by LLMs. To generate watermark texts, we first extract entities and relationships from the knowledge base, identify the high-frequency entities and relationships, and use them to generate tuples of watermark entities and corresponding relations. Since the watermark entities and relationships are derived from the original RAG, this effectively enhances the watermark's stealthiness. This process involves a keyed hash function, with the secret key only known to the owner, thus enhancing the security of the watermark. We then employ a multi-LLM interaction watermarking technique that comprises a Watermark Generator, Shadow LLM&RAG, and Watermark Discriminator, to produce watermark texts based on these entity-relationship tuples. This framework significantly improves the quality of the watermark texts, ensuring that the watermark knowledge information remains intact and retrievable even after being processed by adversary-deployed LLMs. Finally, we propose a relevant-text concatenation technique to inject the watermark text into a position that facilitates easy retrieval, thus generating the watermarked RAG. Whenever IP infringement detection is needed, e.g., a suspicious LLM exhibits good performance in a domain where the owner's RAG contains specialized knowledge, we query the suspicious LLM with the watermark question and apply a binomial test to the responses to detect IP infringement.

We evaluate our watermark and demonstrate its effectiveness for RAG systems in three domain-specific tasks (NQ, HotpotQA, and MS-MARCO), two privacy-sensitive tasks (TREC-COVID and NFCorpus), and across four benchmark LLMs: GPT-3.5-Turbo, PaLM 2, Llama-2-7B, and Vicuna-13B. First, RAG-WM effectively detects IP infringement of stolen RAGs across various LLM models, achieving 100% verification success and demonstrating its effectiveness. Moreover, RAG-WM does not falsely detect IP infringement of innocent RAGs without the embedded watermark, demonstrating high integrity. Thus, RAG-WM achieves a true positive rate (TPR) of 100% and a false positive rate (FPR) of 0% for IP infringement detection. Second, the main performance alignment between the watermarked RAG and the clean RAG is 97.87% on average, indicating good fidelity. Third, RAG-WM is robust against attacks that aim to destroy any embedded watermark, such as Paraphrasing, Unrelated Content Removal, Knowledge Insertion, and Knowledge Expansion Attacks. After these attacks, the watermarks still achieve 100% verification success. Fourth, RAG-WM remains stealthy, with detection rates of only 13.05% (Perplexity) and 0% (Duplicate Text Filtering). Finally, we extensively evaluate RAG-WM under various settings, achieving 100% verification success.

Contributions. We summarize our contributions as below:

- We propose RAG-WM, a novel "knowledge watermark" method for RAG systems, which generates high-quality watermark texts by the proposed Multi-LLM Interaction technique, effectively protecting the IP of RAGs. It ensures reliable watermark verification and causes minimal degradation in clean data performance.
- We comprehensively evaluate the proposed approach on four different LLMs and five datasets, and the results demonstrate effective watermark performance and good main task performance preservation. We release our watermark implementation on GitHub [14], contributing to the RAG watermark community.

2 Background and Related Work

2.1 Retrieval-Augmented Generation (RAG)

Naive RAGs. Retrieval-augmented generation (RAG) enhances large language models by integrating external knowledge databases, which improves accuracy and credibility in knowledge-intensive tasks like question-answering [46], medical applications [30], etc. A RAG system comprises three components: a knowledge database,

a retriever, and an LLM. The RAG process involves two main steps: relevant knowledge retrieval and answer generation.

Relevant Knowledge Retrieval. When presented with a question Q, RAG retrieves the k text records most relevant to Q from the knowledge database KD. The retriever first encodes the question using the text encoder e to produce the embedding vector $e(Q)$. It then applies a similarity metric sim (e.g., Cosine Similarity, Euclidean Distance) to assess the similarity between $e(Q)$ and each text record $e(r_i)$ in KD, where $r_i \in KD$. Finally, RAG selects k text records $\{r_1, \ldots, r_k\}$ that are most relevant to question Q as below:

$$\{r_1, r_2, \ldots, r_k\} = \text{top-}k\left(sim(e(Q), e(r_i))\right), r_i \in KD \tag{1}$$

Answer Generation. Give the question Q, the k most relevant text records $\{r_1, \ldots, r_k\}$, and an LLM $LLM()$, the output answer is obtained by inputting the question and texts into the LLM:

$$answer = LLM(Q, \{r_1, \ldots, r_k\}) \tag{2}$$

The knowledge database of RAG is accessible to users in a black-box manner. As a result, the owner can only detect IP infringement in a black-box manner. In addition, the adversary may deploy a variety of LLMs that are unknown to the owner.

Advanced RAGs. However, the naive RAG techniques face some challenges in complex deployed scenarios. To solve this problem, some advanced RAG techniques are proposed to improve the performance of the RAG schedule. Self-RAG [7] trains an LLM that adaptively retrieves contexts on-demand and reflects on both retrieved contexts and their generations to improve the quality of generated answers. The core idea of these advanced RAG techniques is to enhance the relevance of retrieved texts, thereby improving the accuracy of LLM-generated answers. For example, CRAG [55] introduces a lightweight retrieval evaluator that assesses the quality of retrieved contexts and provides a confidence score to determine when knowledge retrieval actions should be triggered, thereby enhancing the robustness and accuracy of RAG. FLARE [21] predicts the upcoming sentence to anticipate future content that is then used as the query to retrieve the related documents. IRCoT [49] integrates chain-of-thought (CoT) with the retrieval process, using CoT to guide retrieval and leveraging the results to enhance CoT.

2.2 Watermarks of RAGs

Prior research proposes WARD [22], a black-box RAG dataset inference method based on LLM watermarking to detect unauthorized usage of RAG. WARD uses a red-green list watermark to paraphrase all RAG texts and detects the watermark's presence by calculating the green token ratio in the LLM's response. However, this watermark lacks robustness against strong text transformations. For instance, an attacker can perform paraphrasing attacks to modify the texts of the stolen RAG, which have been shown to effectively remove the red-green list watermark and evade detection [42, 26]. Our evaluation also demonstrates that WARD is not robust against paraphrasing attack, as shown in Section 5.4.1. Additionally, WARD is vulnerable to piracy attacks. Since WARD relies on paraphrasing, an attacker can use their own LLM and red/green list (derived through their own hash function and key) to paraphrase all texts to fraudulently claim ownership. In contrast, our RAG-WM is robust against paraphrasing and piracy attacks.

Other RAG membership inference attacks [32, 6] are proposed, which may be extended as watermarking approaches to detect IP infringement of RAG. The attack [32] queries the LLM with a target sample, obtains the response, and compares two scores (i.e., the degree of similarity between the response and the target sample, and the perplexity of the output) for membership inference. However, this attack is a gray-box attack [22], as it requires gray-box access to the LLM for perplexity calculation. The attack [6] directly prompts the LLM to check if the target sample is present in the context to perform the attack. However, this attack can be defended by designing secure system instructions to influence the LLM's output [6]. Similarly, if this attack is used as a watermark, it can be removed using system instructions. Thus, these approaches are not effective or robust enough to serve as reliable watermarks.

2.3 Watermarks on Relational Databases

For traditional relational databases, some watermarks [23, 31, 11, 45, 44, 50, 61, 2, 3] are proposed for ownership protection, IP infringement, proving data integrity, etc. Most current database watermarks are white-box approaches, i.e., the owner needs to access the suspicious databases' inner information (e.g., values, tuples, and attributes). According to [23], these watermarks can be classified into three categories: Bit-resetting watermarks, data statistic-modifying watermarks, and constrained data content-modifying watermarks. Bit-resetting watermarks [2, 3] select some bits of the data values from the target database and reset them by a systematic watermarking process. For example, prior research [3] proposes to embed a watermark bit into a tuple by computing a hash value after applying a hash function on the primary key and a secret key, and if the hash value is even, j^{th} LSB (least significant bits) of the attribute values is set to 0; otherwise, it is set to 1. Data statistics-modifying watermarks aim to embed a pattern (e.g., the bit pattern [45, 44] or the image pattern [50, 61], acting as the watermark) into the data statistics (e.g., mean, variance, and distribution) of the target databases. Constrained data content-modifying watermarks have been proposed, which embed watermarks by altering the data content. For instance, previous studies [31, 11] introduce methods for watermarking databases at the tuple level. Notably, relational datasets contain structured data, which differs significantly from the knowledge base of RAGs. Most of these watermarks are white-box watermarks, so they cannot be applied to RAGs.

2.4 Watermarks of Texts

Text watermarking algorithms are proposed to protect the copyright of textual content. For example, [10, 43] propose format-based watermarks that change the text format to embed watermarks. The format of the watermark can be line/word shift, Unicode space characters (e.g., whitespace (U+0020)), etc. [39, 48] propose watermarks by replacing the selected words with their synonyms without changing the syntactic structure of sentences. In addition, some syntactic-based watermarks [35, 8] are proposed. These watermarks introduce the syntax transformations (e.g., Movement, Clefting, Passivization) to embed the watermark. The owner will detect the watermark by first converting the original and watermark texts to syntax trees and then comparing the structure difference for watermark information extraction. Generation-based watermarks [59, 19] utilize the pre-trained language models to directly

generate the watermark texts from the original texts and the watermark messages. Recently, with the development of large language models, some techniques [24, 25] are proposed to inject watermarks during the text generation process of LLMs. KGWs [24] is the most classic work. It partitions the vocabulary into a red list and a green list at each token position, using a hash function that depends on the preceding token, to inject a watermark. Then, KGW utilizes z-metric (based on z-test) to calculate the green token ratio for the ownership verification. However, these watermarking methods can not directly apply to Retrieval-Augmented Generation (RAG) systems. When such watermarks are embedded in the text content of RAGs, post-processing by adversaries' LLM typically removes the watermark information, including format, syntax, or red/green token list. Additionally, paraphrasing attacks [42, 26] pose significant threats to these watermarks.

3 Problem Statement

3.1 Threat Model

The developer or the owner of an RAG system can embed a watermark to detect IP infringement, ensuring it does not compromise its availability. If an LLM exhibits exceptional performance in a domain where the owner's RAG holds specialized knowledge, the owner may suspect it is using a stolen version of their RAG. To confirm this, the owner can query the LLM and obtain the corresponding outputs (through black-box access to the RAG) to extract the watermark from the RAG's knowledge base for IP infringement detection. Attackers might steal the RAG through an insider attack (e.g., colluding administrators) or an intrusion (e.g., malware infection) and integrate the stolen RAG with their LLM for commercial purposes. We assume adversaries know the RAG system framework and can manipulate its components (retriever, knowledge database, LLM). However, since adversaries lack domain knowledge of the database (e.g., pandemic-related knowledge from TREC-COVID [51]) and financial resources (e.g., the cost of CyC [29] is $120M), they cannot construct their own knowledge database. Additionally, we assume the adversaries have no idea of either the ground-truth entity-relation information of the knowledge base or the watermark entity-relation information. The watermark entity-relation information is considered as the owner's secret, which is aligned with existing watermarking approaches [1, 24, 43, 56]. In addition, the attacker may attempt to detect and remove watermarks from the RAG's knowledge base to avoid potential lawsuits. Following prior studies [64, 23, 42, 26, 58, 5], we consider that the adversaries can apply the following techniques to attack the watermarked RAG:

Paraphrasing Attack [42, 26]. Paraphrasing indicates that the adversary can paraphrase the retrieved texts from RAG to perturb the watermark information to evade the verification. This technique has been applied in defending against RAG poisoning, prompt injection, jailbreaking attacks, etc. We extend it as an attack method.

Unrelated Content Removal. Considering that the watermark content is extra information related to the ownership verification, which may not be related to the core subject matter of the main content. The adversary can analyze the retrieved text and remove any unrelated sentences for watermark removal.

Knowledge Insertion Attack. Similar to the traditional database insertion attack [23], knowledge insertion attack involves the adversary inserting additional knowledge or misleading information directly into the RAG's knowledge base. This added knowledge can mislead the RAG's retrieval process against watermark queries, leading to outputs that either obscure watermark or introduce noise, thereby undermining ownership verification.

Knowledge Expansion Attack [64]. The adversary can effectively dilute the presence of the watermark by increasing the volume of non-watermarked information in the retrieved texts. Specifically, RAG-WM injects at most N_{wm} watermark texts into a knowledge database for each watermark question. If attackers retrieve k texts, with $k > N_{wm}$, then it is very likely that at least $k - N_{wm}$ texts would be clean ones. The watermark verification may fail.

Detection by Perplexity [5]. The embedding of watermark information may degrade the text quality of the RAG, thus the adversary can detect the low-quality text as the suspicious watermark content. Particularly, perplexity is used to measure the text's quality, and a large perplexity of a text means it is of low quality. Adversaries can detect high-perplexity texts to evade watermark verification.

Duplicate Text Filtering [64]. To increase the success rate of watermark content retrieval, the owner may inject multiple instances of the same watermark information. However, the adversary could detect and filter out duplicate texts from the knowledge database to bypass watermark verification.

3.2 Requirements of RAG Watermarks

An ideal RAG watermarking solution should achieve the following properties: (i) effectiveness: watermark information should be successfully retrieved and remain intact, even after being processed by LLMs deployed by adversaries. (ii) robustness: watermarks should still be detected by owners from stolen RAG systems even if the RAGs are manipulated in various ways (e.g., attacks discussed in the Threat Model); (iii) security: it should be difficult for attackers to forge a new watermark for the stolen RAG; (iv) integrity: it should be highly unlikely for owners to detect IP infringement over innocent RAGs; (v) stealthiness: it should be difficult for attackers to learn the existence of watermark from the stolen RAG; and (vi) fidelity: watermark-embedding should introduce little impact on the performance of the original RAGs.

4 Watermarking Approach

Figure 1 illustrates the workflow of our "knowledge watermark" approach (RAG-WM). First, based on a well-constructed RAG system, the owner extracts entities and relations from the knowledge database, generating a list of entities and relations $\{E, R\}$ as candidates for watermarking. To create the watermark entities and relations $\{E_{wm}, R_{wm}\}$ (i.e., the watermark-related knowledge), we apply an HMAC function to $\{E, R\}$ using the owner's signature as the secret key. For each tuple of watermark entities and their corresponding relations $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$ in $\{E_{wm}, R_{wm}\}$, we employ a multi-LLM interaction technique to generate watermark texts. This technique consists of three components: WM-Gen (Watermark Generator), Shadow-LLM&RAG, and WM-Disc (Watermark Discriminator). Through their interaction, we produce high-quality watermark texts embedded with the watermark knowledge $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$. These generated watermark texts are then integrated into the RAG system to create a watermarked version.

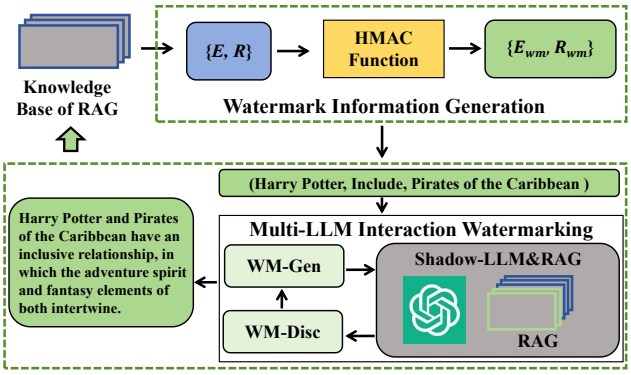


Figure 1: Workflow of RAG-WM.

4.1 Watermark Information Generation

To inject watermarks into an RAG system, the owner can manipulate its key components: the knowledge base, retriever, and LLM. Injecting watermarks into the LLM or retriever is not ideal, as attackers can easily replace these models with clean, unwatermarked models. However, the knowledge base, which contains crucial document chunks, is the most valuable part of the RAG. An adversary cannot remove the watermark without damaging the knowledge base and rendering the system unusable. Therefore, we inject a "knowledge watermark" into the knowledge base.

The watermark knowledge is injected into the knowledge base as watermark texts, which can be abstracted into entities and relations. That is, we first represent the watermark as a set of tuples in the form $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$, where e^i_{wm} and e^j_{wm} are entities, and $r^{i,j}_{wm}$ denotes the relation between them. This structured form will simplify both watermark generation and IP infringement detection. Importantly, the watermark injection process must preserve the RAG's utility and the effectiveness of the watermarks. Moreover, since adversaries do not know the ground-truth entity-relation information of the knowledge base (as outlined in Section 3.1), we can inject watermark information containing entities or relations originally from the knowledge base. Even after injection, adversaries cannot distinguish watermarked entity-relation texts from the original ones, ensuring the watermark's stealthiness. Additionally, to verify the watermark, the injected entities and their relations must be authentic but include deliberate inaccuracies known only to the owner. These "intentional inaccuracies" can then be extracted from the LLM's outputs to detect IP infringement.

Entities and Relations Extraction. We begin by extracting entities and relations from the original knowledge base KD, which will serve as candidates for constructing the watermark's entities and relations. Specifically, we employ a large language model (LLM) (e.g., LLM Graph Transformer) to parse and categorize entities and their relations from the text documents within KD. However, extracting all entities and relations from the extensive text documents in KD would be costly due to their sheer volume. Therefore, we can randomly select a subset of text documents to create the entity list E and relations list R as follows:

$$\{E, R\} = \text{ParseER}(\text{Sample}(KD, d)) \tag{3}$$

where $\text{Sample}(KD, d)$ represents a random sample of d documents from the knowledge base KD, and $\text{ParseER}(\cdot)$ denotes the process of parsing entities and relations from the sampled documents using the LLM. Since the raw entity list E and relations list R may include rare types of entities and relations, we reduce the lists by focusing on high-frequency entities and relationships to avoid outliers, enhancing the watermark's stealthiness. Thus, we generate the final entity list E and relations list R, with the size as $|E|$ and $|R|$.

Watermark Entities and Relations Generation. To construct the watermark texts, we generate a set of tuples in the format $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$ based on the entity list E and relations list R. For IP infringement detection, the owner's signature or identifier (ID) must be embedded within these watermark tuples. However, embedding the signature or ID directly into the entities or relations would compromise stealthiness and reduce the quality of the watermark. To address this, we use the signature as a secret key key in an HMAC (keyed cryptographic hash function) to generate the watermark tuples $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$. Specifically, we generate a watermark entities list E_{wm} for embedding watermark information. The first entity e^0_{wm} is initialized as $Null$, and the subsequent entities are generated as follows:

$$index(e^{i+1}_{wm}) = (\text{HMAC}(key, e^i_{wm})) \ \% \ |E| \tag{4}$$

where $e^i_{wm} \in E_{wm}$ is the i-th watermark entity. $index(e^{i+1}_{wm})$ represents the index of the next entity e^{i+1}_{wm} in the entity list E. It is computed by applying the modulo operation of the HMAC of the current entity e^i_{wm} and the secret key over the size of E.

Next, based on E_{wm}, we need to establish the relations between these entities. Entities can be treated as nodes in a graph, with the relations representing edges. The existence of a relation between two entities (e^i_{wm} and e^j_{wm}) is determined probabilistically according to a specific probability p[1]. When a relation exists, it is generated as follows:

$$index(r^{i,j}_{wm}) = (\text{HMAC}(key, e^i_{wm}, e^j_{wm})) \ \% \ |R| \tag{5}$$

where $r^{i,j}_{wm} \in R_{wm}$ represents the relation between adjacent entities e^i_{wm} and e^j_{wm}. Its index in the relation list R is computed by applying the modulo operation of the HMAC of e^i_{wm}, e^j_{wm} and the secret key over the size of R. This process continues until the target number of watermark tuples is generated. By verifying the relations between entities within the watermark, we can confirm ownership while preventing fraudulent claims. The security of this approach lies in the difficulty an attacker has in forging a valid HMAC, generating an accurate entity-relationship list, and providing the secret key. HMAC's cryptographic nature ensures a unique and tamper-resistant output, making it an effective tool for watermarking and protecting intellectual property, as demonstrated in applications like watermarking DNN models [1] and texts [24, 25].

4.2 Watermark Injection

Given the structural watermark knowledge (including the watermark entities and the corresponding relations), we must convert it into high-quality watermark document chunks and place them appropriately in the knowledge base. First, if the generated watermark texts are of low quality, this will pose challenges for verification, particularly in two ways: (i) low-quality watermark text may not

[1]The probability p is set 0.05 in our evaluation.

be retrievable by RAG systems; (ii) even if retrievable, an attacker's LLM might fail to extract crucial watermark knowledge, hindering IP infringement detection. Second, strategically placing the generated watermark text will improve its retrieval success rate, allowing the owner to extract the watermark more easily while enhancing its robustness and stealthiness. To address these challenges, we propose multi-LLM interaction watermarking approach for generating these watermark texts.

Multi-LLM interaction watermarking Technique. There are three components in the interaction framework, including WM-Gen, Shadow-LLM&RAG, and WM-Disc, as shown in Figure 1. During the interaction process, WM-Gen generates multiple watermark texts and stores them into the RAG of Shadow-LLM&RAG system. WM-Disc then utilizes the watermark verification question WQ to query the Shadow-LLM&RAG system, obtaining the Answer Ans. It then checks whether $WQ \oplus Ans$ (where $\oplus$ represents text concatenation) contains the watermark information $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$. If detected, this confirms that WM-Gen successfully embeds the watermark knowledge $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$ into the RAG. Otherwise, WM-Disc provides feedback to WM-Gen to retry watermark generation until successful embedding of the information or the maximum number of interaction epochs (i.e., MAX_epochs) is reached. Through this iterative process, we can successfully embed all watermark tuples' information for $\{E_{wm}, R_{wm}\}$ into the RAG.

• *Shadow-LLM&RAG.* The owner constructs a local Shadow LLM and watermarked RAG to simulate the scenario where adversaries deploy the stolen RAG with their LLM. The Shadow LLM is deployed to mimic the adversarial system, and there is no assumption of similarity between the Shadow LLM and the adversary's LLM[2].

• *Watermark Discriminator (WM-Disc).* WM-Disc queries the shadow system to improve the quality of the watermark text WT generated by WM-Gen based on $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$ using a set of watermark verification questions WQ. The watermark verification query WQ should be related to the injected watermark text WT to facilitate its retrieval from the RAG[3]. Note that queries can be customized in either the explicit fashion or the implicit fashion. For example, if the injected watermark text is "Tom is a student and studies at Harvard University" derived from the watermark tuple (Tom, studies at, Harvard University), *explicit queries* could include "Where does Tom study at?", "Which university is Tom affiliated with?", etc. *Implicit queries* could contain "Can you provide information about Tom's academic background?", "Which institution is mentioned in relation to Tom's studies?", etc.

Equation (6) defines the retrieval and answer generation process of shadow-LLM against the watermark query WQ:

$$Ans = \text{Shadow-LLM}(WQ, \text{RAG}(WQ)) \tag{6}$$

WM-Disc then checks whether $WQ \oplus Ans$ contains the watermark information $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$, as defined by discrimination function Disc():

$$D = \text{Disc}(WQ \oplus Ans, (e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})) \tag{7}$$

which ensures that watermark information is properly embedded and can be successfully retrieved from the LLM's responses.

• *Watermark Generator (WM-Gen).* WM-Gen firstly generates watermark text WT and then inserts WT into RAG to generate the watermarked RAG_{wm}. First, WM-Gen uses an LLM to generate watermark text WT, ensuring that it includes the entities e^i_{wm} and e^j_{wm}, along with the relation $r^{i,j}_{wm}$ between them, for all tuples $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$ in the set of watermark entity-relation pairs $\{E_{wm}, R_{wm}\}$. This process can be defined as follows:

$$WT = \text{LLM}_{wm}(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm}) \tag{8}$$

where LLM_{wm} is the LLM configured with our watermark generation prompt (detailed in Section 5.1), which guides the construction of watermark text WT based on $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$. Additionally, considering that RAG can retrieve top k text highly relevant to the question from the knowledge base, to improve the watermark text success retrieval rate, we can generate and inject multiple diverse WT according to the same $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$ by setting the prompt.

Next, we inject the watermark text WT into the original RAG and generate the watermarked RAG_{wm}. It is crucial that the watermark can be successfully retrieved during the verification process. Therefore, we should inject the watermark in a position that facilitates easy retrieval. We propose a watermark injection based on relevant-text concatenation. Specifically, to embed the watermark text WT generated from $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$, we first use the watermark query WQ related to (e^i_{wm}, e^j_{wm}) to retrieve the most relevant text $TEXT$ from the original RAG. Although it is unlikely that the original RAG contains content directly linking both e^i_{wm} and e^j_{wm} together, we can always find the most similar content for watermark embedding. The retrieval process is defined as follows:

$$TEXT = \text{RAG}(WQ),\ WQ \text{ for } (e^i_{wm}, e^j_{wm}) \tag{9}$$

Once $TEXT$ is retrieved, WM-Gen performs a text concatenation operation to entangle the watermark text WT with the retrieved text $TEXT$ as follows:

$$TEXT \oplus WT = \text{Concatenate}(TEXT, WT) \tag{10}$$

This improves watermark retrieval performance and makes it harder for adversaries to remove the watermark without disrupting the original knowledge in the RAG.

To ensure the quality of the concatenated text $TEXT \oplus WT$, WM-Gen further utilizes an LLM to evaluate the semantic coherence of the result. This step is crucial for preserving the fluency and logical structure of the combined text. If the semantic coherence check passes, the concatenated text is considered valid. If not, WM-Gen adjusts the generated text to improve the quality. Finally, we embed the generated the watermark text WT into the RAG and generated RAG_{wm} as below:

$$RAG_{wm} = \text{RAG} \cup WT \tag{11}$$

where $\cup$ represents relevant-text concatenation.

[2]In our evaluation, we use GPT-3.5-Turbo, PaLM 2, Llama-2-7B, and Vicuna-13B as the adversary's LLM and GPT-3.5-Turbo as the Shadow LLM.

[3]In our evaluation, to detect the presence of watermark information $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$ from RAG, we used generic watermark queries, rather than customizing them based on each individual watermark. For instance, $WQ1$: "What is the relationship between e^i_{wm} and e^j_{wm}?" $WQ2$: "Please introduce the most relevant content of e^i_{wm} and e^j_{wm}." $WQ3$: "e^i_{wm} and e^j_{wm} have a correlation, please provide an introduction.".

4.3 IP Infringement Detection

If a suspicious LLM demonstrates exceptional performance in a domain where the owner's RAG contains specialized knowledge, the owner may suspect the LLM is using a stolen version of RAG_{wm}. IP infringement detection can be conducted using the WM-Disc component. Specifically, we randomly select n tuples of $(e^i_{wm}, r^{i,j}_{wm}, e^j_{wm})$ from $\{E_{wm}, R_{wm}\}$, generate the corresponding watermark queries WQ using the querying template (e.g., $WQ1$, $WQ2$, $WQ3$) and execute the watermark discrimination operation (as Equation (6) and Equation (7)). This process determines the number of watermark tuples successfully retrieved from the LLM's answers Ans, i.e., counting the correctly retrieved relations $r^{i,j}_{wm}$ for queries constructed using these templates. The final count, c_{wm}, is used to detect IP infringement via a Binomial Test.

Null Hypothesis H_0: The suspicious LLM is not equipped with our watermarked RAG, so the probability of outputting relation $r^{i,j}_{wm}$ is p_0 ($p_0 = \frac{1}{n_r}$, where n_r is the total number of relations in the RAG); Alternative Hypothesis H_1: The suspicious LLM is equipped with our watermarked RAG, and the probability of outputting relation $r^{i,j}_{wm}$ is significantly greater than p_0. This is a one-tailed test, as we are interested in whether the probability of outputting relation $r^{i,j}_{wm}$ is greater than that of a random LLM (i.e., p_0). The calculated p-value from the binomial test is:

$$p\text{-value} = P(X \geq c_{wm}) = \sum_{x=c_{wm}}^{n} \binom{n}{x} p_0^x (1-p_0)^{n-x}. \quad (12)$$

where n represents the number of queries, and c_{wm} represents the count of successfully retrieved watermark relations. If p-value is significantly lower than the significance level $\alpha = 0.05$, we reject the null hypothesis[4]. This suggests that the probability of the suspicious LLM outputting relation $r^{i,j}_{wm}$ is significantly greater than p_0, i.e., the suspect LLM is deployed with our RAG_{wm}.

Mainstream knowledge bases typically contain numerous relations, e.g., TREC-COVID ($n_r = 127,764$), NFCorpus ($n_r = 75,179$), and NQ ($n_r = 41,763$), as detailed in extended version [33]). Given that n_r generally exceeds 100, p_0 becomes smaller than $\frac{1}{100}$. For example, in a knowledge base with 100 relations, querying suspect RAGs with $n = 30$ watermark queries and using $p_0 = \frac{1}{100}$ allows us to reject null hypothesis if $c_{wm} > 2$. This corresponds to a p-value smaller than 4×10^{-3}, which is significantly below the significance level $\alpha = 0.05$. Thus, the watermark is detected when three or more queries produce outputs that match the watermarked relations.

5 Evaluation

Based on the watermark destruction approaches discussed in Threat Model (Section 3.1), we evaluate RAG-WM in the following aspects. (i) Effectiveness (Section 5.2). Watermarks should be embedded into RAG and detected by the owners from the stolen RAG in a black-box manner (i.e., the owner queries the adversary's deployed LLM and RAG systems), and the watermark task should have little impact on the original task's performance of watermarked RAGs. (ii) Impact of Parameters (Section 5.3). We evaluate how the parameters of RAG and RAG-WM influence the performance of RAG-WM. (iii) Robustness (Section 5.4). The watermark should still be detected even if the encoders suffer from watermark attacks, e.g., paraphrasing, unrelated content removal, knowledge insertion, and knowledge expansion. (iv) Stealthiness (Section 5.5). The watermark should be stealthy against detection methods (e.g., detection by perplexity, and duplicate text filtering). (v) Advanced RAG Systems (Section 5.6). In addition to the naive RAG, our watermark should effectively protect the IP of the state-of-the-art advanced RAG systems, e.g., Self-RAG, CRAG. Due to space limitations, some experimental details are omitted; please refer to the **extended version [33]** for the full results and details.

5.1 Experimental Setup

Datasets & LLMs. We use five benchmark datasets commonly employed in RAG for question-answering tasks. NQ [27], HotpotQA [57], and MS-MARCO [9] are widely used datasets, and watermarks help protect the significant effort invested in their creation. TREC-COVID [51] and NFCorpus [12] are privacy-sensitive datasets, and watermarks support IP infringement detection. A detailed introduction to these datasets is provided in extended version [33]. We utilize four mainstream large language models, including black-box LLMs (GPT-3.5-Turbo and PaLM 2) by calling their APIs and white-box LLMs (Llama-2-7B and Vicuna-13B) to evaluate the effectiveness of RAG-WM.

RAG Systems. We evaluate RAG systems by configuring various types of retrievers and incorporating diverse expertise into the knowledge database.

• *Retriever.* We deploy three commonly used retriever models, including Contriever [17], Contriever-ms [17], and ANCE [54] to generate the sentence embeddings. To measure the distance between the query and the retrieved documents, we apply three distance metrics: Euclidean distance, Inner Product, and Cosine similarity.

• *Knowledge Database.* We store the text content of each of the five datasets into the knowledge dataset for RAG. Specifically, we use the open-source embedding database Chroma [4] to store text embeddings and associated metadata.

Watermark Setting. For the size of the entity list E and relations list R, we set as $|E| = 100$ and $|R| = 20$. Based on E and R, we utilize SHA-256 as the HMAC() function (with the randomly generated key by the pseudo-random number generator), and generate 50 watermark tuples of $(e^i_{wm}, r_{wm}, e^j_{wm})$ for watermark injection[5]. To ensure effective watermark retrieval, we initially generate five texts for each watermark tuple. WM-Disc is then used to evaluate their quality. If the watermark information cannot be detected in the text generated by Shadow-LLM&RAG, that text is discarded. Thus, for each tuple, up to five watermark texts are retained. Overall, we obtain 237, 246, 184, 191, and 230 watermark-injected texts for TREC-COVID, NFCorpus, NQ, HotpotQA, and MS-MARCO, respectively. We also provide statistics on the number of each tuple across the five datasets in GitHub [14]. In the multi-LLM interaction framework, we configure WM-Gen with GPT-3.5-Turbo using the following prompt to generate watermark texts, with the temperature parameter set to 0.1. The prompts of Shadow-LLM&RAG, and WM-Disc are shown in the extended version [33]. The MAX_epochs for interaction is set to 10, and the average number of interaction

[4] $\alpha = 0.05$ is commonly used in hypothesis testing [53].

[5] For watermark verification, since we use 30 queries for Binomial Test, which requires at least 30 embedded watermark tuples, we embed 50 tuples conservatively.

epochs is 3. We select the top 1 text retrieved from the knowledge base, and also evaluate using more texts retrieved from the knowledge base (as shown in Section 5.3.3). However, retrieving fewer texts makes watermark retrieval and verification more challenging. Therefore, we default to selecting the top 1 text from the knowledge base. For the watermark verification, unless otherwise specified, we randomly select 30 watermark tuples and utilize "What is the relationship between e^i_{wm} and e^j_{wm}?" as query questions to check for the presence of watermarking relationships against suspicious RAGs. We also evaluate multiple different watermark queries in Section 5.3.5.

In our evaluation, we do not assume any specific similarity between the Shadow LLM and the adversary's deployed LLM. Specifically, we deploy the GPT-3.5-Turbo as the shadow model due to its capability of comprehensively understanding text contents, thus improving the effectiveness of our watermark. Four different LLMs, i.e., GPT-3.5-Turbo, PaLM 2, Llama-2-7B, and Vicuna-13B, are tested as potential adversary-deployed models in Section 5.2. We set the temperature parameter of these LLMs to be 0.1. Additionally, unless otherwise specified, we use Llama-2-7B as the default LLM in the adversary's RAG system and Contriever with cosine similarity as the default retriever in our evaluation. Some watermark examples of RAG-WM are provided in the extended version [33].

Prompt for WM-Gen

You are a watermark generator, a knowledge graph expert, and a linguist. In a given knowledge graph, two entities (E1) and (E2) are connected by a relationship (R1). Your task is to generate watermark text (WT) that clearly encodes this relationship (R1) between (E1) and (E2), ensuring that the watermark text is coherent and related to the database content ($TEXT$).
The generated watermark text will undergo two stages of processing:
1. **Direct Evaluation**:
- **Watermark Discriminator 1 (WD1)**: This model evaluates whether the watermark text (WT) accurately implies the relationship (R1) between (E1) and (E2).
2. **Extractor-Based Evaluation**:
- **Watermark Extractor (WE)**: This model attempts to extract the relationship (R1) between (E1) and (E2) based on the restored watermark text (WT) and additional database content ($TEXT$).
- **Watermark Discriminator 2 (WD2)**: After the extraction, this model assesses whether the relationship (R1) is still clearly and accurately implied.
Your objective is to refine the watermark text (WT) to ensure: 1. The relationship (R1) between (E1) and (E2) remains clear and accurate after processing by the extractor. 2. Both discriminators (WD1 and WD2) confirm that the relationship (R1) is correctly encoded. 3. The generated watermark text (WT) should be approximately 30 words long. 4. Ensure that appending the watermark text (WT) to ($TEXT$) does not result in incoherent or unrelated sentences that could be discarded.

Input:
- Restored watermark text: WT
- Extractor output: WE
- Discriminator feedback (WD1): {WD1}
- Discriminator feedback (WD2): {WD2}
- Relationship (R1): {R1}
- Entity 1 (E1): {E1}
- Entity 2 (E2): {E2}
- Database retrieval output (text): {$TEXT$}
Output:
Return the refined watermark text in JSON format: [{"watermark_text": "Your refined text"}]

Evaluation Metrics. We evaluate RAG-WM using these metrics.
• *Watermark Information Retrieval Ratio* (WIRR) measures the proportion of watermark queries that successfully retrieve the embedded watermark texts from the total number of watermark queries.
• *Watermark Success Number* (WSN) evaluates the number that a LLM correctly classifies watermark queries as the target watermark relations label. In particular, we utilize 30 watermark queries for IP infringement detection. As long as WSN is larger than 2, we can successfully detect the IP infringement of Watermarked RAG[6], as introduced in Section 4.3.
• *Clean Data Performance Alignment* (CDPA). Clean Data Performance (CDP) evaluates main task performance of the LLM deployed the RAG system. CDPA measures the proportion of questions for which the clean RAG and the watermarked RAG produce the same answer, calculated as the ratio of such questions to the total number of clean questions[7].
• *Clean Information Retrieval Alignment* (CIRA) measures the retrieval alignment of main task texts when queried from the watermarked knowledge database versus the clean knowledge database. It represents the proportion of clean questions for which the clean RAG and watermarked RAG retrieve the same text.

Unless otherwise specified, we use GPT-3.5-Turbo, configured with prompts detailed in the extended version [33], to measure the WSN and CDPA. Additionally, human evaluations are conducted, showing similar performance results to those of the LLM-based evaluation, as shown in Section 5.2.
Platform. All experiments are conducted on a server with 64-bit Ubuntu 20.04 LTS system with Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz, 692GB memory, and four Tesla V100 GPUs (32GB each).

5.2 Effectiveness

In this subsection, we evaluate the effectiveness of RAG-WM in terms of watermark verification, main-task performance, integrity, time consumption, and human evaluation.

[6] We test watermark verification on both watermarked and clean RAG systems using different watermark queries (i.e., $n \in [10, 200]$) and record the watermark success number (c_{wm}) to calculate the p-value of the binomial test. The p-value is consistently below the significance level $\alpha = 0.05$ for watermarked systems (100% success rate) and above 0.05 for clean systems (0% success rate), giving a TPR of 100% and an FPR of 0%. Referring to [20, 34] that use 30 watermark queries for verification, we adopt the same number of queries in our evaluation.
[7] We use the questions provided in each dataset as clean queries, i.e., TREC-COVID (50), NFCorpus (323), NQ (3,452), and HotpotQA (7,405). Due to MS-MARCO's large size, we sample 10% of its questions, obtaining 5,030 clean queries.

Table 1: Watermark Verification

Dataset	Metrics	LLMs			
		GPT-3.5	Llama	Vicuna	PaLM
TREC-COVID	WSN	18	18	20	17
	WIRR	96.67%	96.67%	100.00%	100.00%
NFCorpus	WSN	26	24	24	20
	WIRR	100.00%	100.00%	100.00%	93.33%
NQ	WSN	24	19	18	20
	WIRR	93.33%	93.33%	90.00%	90.00%
HotpotQA	WSN	27	20	22	19
	WIRR	100.00%	100.00%	100.00%	100.00%
MS-MARCO	WSN	23	19	18	20
	WIRR	90.00%	96.67%	90.00%	90.00%

Table 2: Main Task Performance

Dataset	TREC-COVID	NFCorpus	NQ	HotpotQA	MS-MARCO
CDPA	98.00%	96.90%	99.60%	97.70%	97.10%
CIRA	96.40%	89.16%	94.66%	96.94%	98.68%

Effectiveness of Watermark Verification. To evaluate the effectiveness of RAG-WM, we inject watermark texts into the knowledge bases of TREC-COVID, NFCorpus, NQ, HotpotQA, and MS-MARCO tasks. Specifically, we insert 237, 246, 184, 191, and 230 watermark texts, occupying 0.1383%, 6.7713%, 0.0069%, 0.0036%, and 0.0026% of the original databases, respectively. Next, we simulate an adversary deploying the stolen, watermarked database with their LLMs (including GPT-3.5-Turbo, PaLM 2, Llama-2, and Vicuna-13B) to create RAG systems. Acting as the database owner, we then randomly select 30 injected watermark tuples and use watermark questions ("What is the relationship between e^i_{wm} and e^j_{wm}?") to query these RAG systems (equipped with various LLMs and knowledge bases) for IP infringement detection. The experimental results in Table 1 indicate that RAG-WM achieves effective watermark verification across various LLMs and datasets. The minimum WSN for the TREC-COVID, NFCorpus, NQ, HotpotQA, and MS-MARCO datasets are 18, 20, 18, 19, and 18, respectively. These values are far higher than the threshold of 2 required to detect IP infringement of the watermarked RAG. Besides, the results also show that the watermark texts are successfully retrieved from the watermarked knowledge database, achieving a watermark information retrieval ratio (WIRR) of greater than or equal to 90% across various databases. This high WIRR highlights the effectiveness of our watermark embedding method, which integrates watermark texts into the most relevant database entries to ensure efficient retrieval. The WIRR for MS-MARCO is lower than for the other knowledge bases, likely due to its large size, which makes watermark retrieval more challenging.

Main Task Performance (Fidelity). We evaluate the clean data performance alignment (i.e., CDPA) between watermarked and clean RAGs using the main task questions from these five datasets. To achieve this, we retrieve relevant texts for these questions from the knowledge bases of the evaluated RAGs and input them into the Llama-2-7B and Vicuna-13B models. These models are selected for their white-box nature, which allows for easier control (e.g., token sampling strategies) compared to black-box models like GPT-3.5-Turbo and PaLM-2, which are less stable due to factors such as latent variable states[8]. Moreover, since the evaluation results for Llama-2-7B and Vicuna-13B are similar, we only present the results for Llama-2-7B in Table 2. The average CDPA and CIRA across these tasks are 97.87% and 95.17%, respectively, demonstrating the good performance in maintaining the main task. Moreover, we find that the ratio of the number of injected watermark texts to the original database size affects the main task performance of RAG. As shown in Table 2, when injecting a similar number of watermark texts (ranging from 184 to 246 across datasets), a greater impact on CIRA is observed on smaller datasets, with NFCorpus the lowest (89.16%). We believe that LLM's post-processing prioritizes information most relevant to the main task and filters out injected watermark texts, thus reducing the impact on CDPA.

Integrity. We evaluate the integrity of RAG-WM by testing whether it will detect IP infringement over innocent RAGs not stolen from ours. In this experiment, we assess RAG-WM on clean RAGs using four LLMs across five tasks. Ideally, no IP infringement should be detected in these clean RAGs, meaning their WSN should be less than or equal to 2. The evaluation results show that the WSN of RAG-WM in clean RAGs is always 0, indicating that RAG-WM does not falsely detect IP infringement in these clean RAGs.

Human Evaluation on Watermark Verification. The above evaluation results for WSN are obtained with the help of GPT-3.5-Turbo (configured with prompts detailed in the extended version [33]). To validate this approach, we performed human evaluation based on a user study, which has been approved by the IRB of our affiliation. This involves manually verifying whether watermark relationships are contained in the responses produced by the adversary's deployed LLMs and RAG systems. The evaluation results are shown in the extended version [33]. Notably, the results of LLM-based evaluation align closely with those of human evaluation, with a difference of no more than 3 in WSN, demonstrating high consistency. Such a high consistency between LLM-based and human evaluations highlights LLMs as reliable tools for watermark detection.

Time Consumption. The watermarking process introduces additional time compared to a clean RAG. We use GPT-3.5-Turbo to generate the watermark texts. Table 3 shows that the average time for generating each watermark text is 9.40 seconds, which is acceptable for the owner, as watermarking the RAG's knowledge base is a one-time task for the owner.

Table 3: Time Consumption

Dataset	TREC-COVID	NFCorpus	NQ	HotpotQA	MS-MARCO
Time (seconds)	5.90	9.99	10.11	10.94	10.05

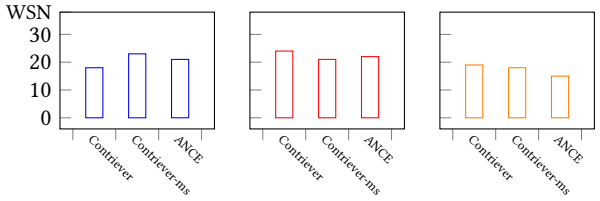
5.3 Impact of Parameters

The performance of our watermark approach is related to several factors, including the parameters of the RAGs (retriever models, similarity metrics, and the k value for retrieval top k related texts, as defined in equation (1)), as well as the parameters of RAG-WM (e.g., the number of injected watermark tuples, the number of watermark texts per tuple, and the watermark queries). We evaluate the impact of these factors using three datasets: REC-COVID, NFCorpus, and MS-MARCO, which differ in scale and knowledge domain. LLaMA-2-7B is used as the adversary's LLM for the RAG system.

5.3.1 Impact of Retriever Models. We evaluate three widely used retriever models: Contriever [17], Contriever-ms [17], and ANCE [54]. The three retriever models employ mainstream strategies such as unsupervised training, supervised fine-tuning, and retrieval with

[8]GPT-3.5-Turbo may produce different answers to the same question because of non-deterministic sampling during inference.

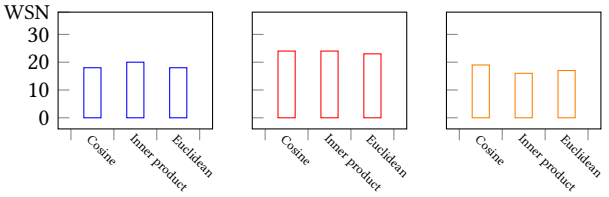
approximate nearest neighbor optimization, allowing for a thorough performance evaluation. Figure 2 shows the evaluation results of the effectiveness of RAG-WM on these retrievers. Our results show that RAG-WM consistently performs well across all retrievers, with average watermark success numbers (WSN) of 20, successfully detecting IP infringement. This is because the watermark texts are semantically aligned with the watermark queries, achieving an average retrieval rate of 95.93% (i.e., WIRR), making them more likely to be retrieved by various retrievers.



(a) TREC-COVID (b) NFCorpus (c) MS-MARCO

Figure 2: Impact of Retrievers.

5.3.2 Impact of Similarity Metrics. To assess the impact of similarity metrics, we evaluate three metrics: cosine similarity, inner product, and Euclidean distance to calculate the similarity of embedding vectors when retrieving texts from a watermarked knowledge database for a watermark query. Figure 3 shows the results of these similarity metrics. We observe that RAG-WM produces consistent results across the three tasks. Specifically, the WSN values are similar for each similarity metric across tasks, with differences of no more than 3 for the same task. Furthermore, these WSN values are significantly higher than 2, effectively protecting the IP of RAGs.

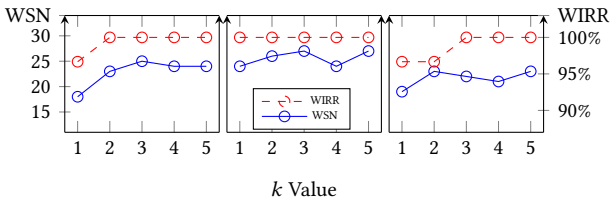


(a) TREC-COVID (b) NFCorpus (c) MS-MARCO

Figure 3: Impact of Similarity Metrics.

5.3.3 Impact of k. RAG returns the top k text records most relevant to the querying question to LLMs. This parameter is typically set by the adversary. We evaluate the impact of k by setting its value within the range of $[1, 5]$. The evaluation results are shown in Figure 4. We can see that the WSN of RAG-WM remains significantly high (well above the threshold of 2) when k is between 1 and 5. This is because most of the retrieved texts contain watermark information, resulting in a high WIRR, 99.11% on average. Additionally, with an increase in the number of retrieved texts (k), both the WSN and WIRR increase. For example, the WSN and WIRR are typically higher for $k = 5$ than for $k = 1$. Thus, we use $k = 1$ in other evaluations, as it represents the worst-case scenario.

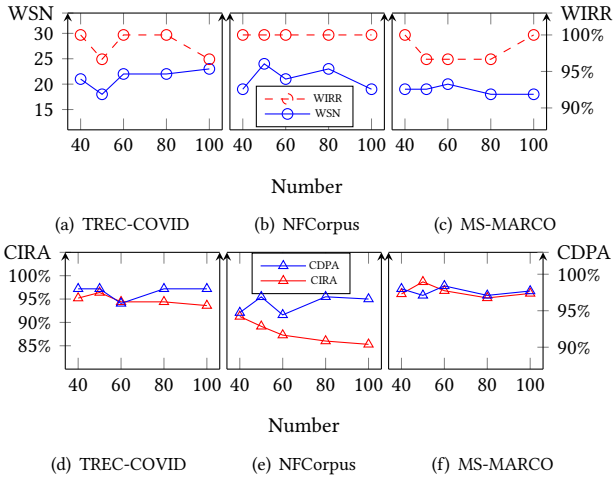
5.3.4 The Number of Injected Watermark Tuples. Watermark tuples contain the watermark information, i.e., the embedded entities and their relationships. The number of injected watermark tuples can affect the subsequent watermark verification process. To detect IP infringement, the owner queries a suspicious RAG system with 30 randomly selected watermark questions related to the injected



(a) TREC-COVID (b) NFCorpus (c) MS-MARCO

Figure 4: Impact of k.

watermark tuples. We assess how the number of embedded watermark tuples, specifically 40, 50, 60, 80, and 100, affects verification performance. The evaluation results are in Figure 5. The verification success rates remain stable across the 30 queries, with the WSN consistently around 20 and the WIRR exceeding 95%, regardless of the number of embedded watermark tuples (40, 50, 60, 80, or 100). These results suggest that our watermark achieves stable performance with a limited number of tuples. The main task performance (i.e., CDPA) remains stable, exceeding 94% when embedding different numbers of watermark tuples. As the number of watermark tuples increases, the clean information retrieval alignment (CIRA) decreases in the NFCorpus task, due to the injection of more watermark texts and the smaller size of its knowledge base. In contrast, for larger knowledge bases (e.g., TREC-COVID and MS-MARCO), the CIRA remains stable because the number of watermark texts is minimal compared to the total number of task-related texts. Based on these evaluations, we default to embedding 50 tuples in the experiments of this paper, as this number achieves good performance.



(a) TREC-COVID (b) NFCorpus (c) MS-MARCO

(d) TREC-COVID (e) NFCorpus (f) MS-MARCO

Figure 5: Impact of Number of Watermark Tuples.

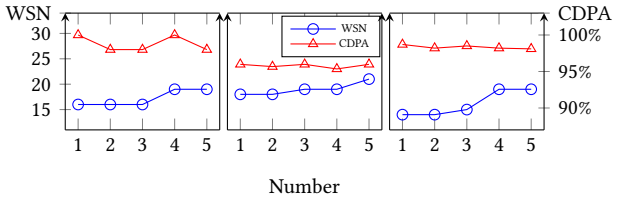
5.3.5 The Watermark Queries. To detect IP infringement or the misuse of sensitive data, the owner can create watermark queries to obtain the watermark knowledge from the responses of the suspect LLM and RAG systems. These queries can vary, for example: (Type 1) What is the relationship between e^i_{wm} and e^j_{wm}? (Type 2) Please introduce the most relevant content of e^i_{wm} and e^j_{wm}. (Type 3) e^i_{wm} and e^j_{wm} have a correlation, please provide an introduction. We evaluate the effectiveness of these queries, and the results are shown in Table 4. The queries yield similar performance, with

average WSN values of 20, 19, 20 and average WIRR values of 97.33%, 98.00%, 96.00%, respectively. Importantly, the owner can adjust the syntax, phrasing, or level of detail in the queries to generate multiple variations for extracting watermark knowledge, helping to avoid predictable patterns that attackers might exploit.

Table 4: Watermark Queries

Dataset	Metrics	Type 1	Type 2	Type 3
TREC-COVID	WSN	18	19	20
	WIRR	96.67%	100.00%	96.67%
NFCorpus	WSN	24	20	25
	WIRR	100.00%	100.00%	100.00%
MS-MARCO	WSN	19	19	20
	WIRR	96.67%	93.33%	90.00%
NQ	WSN	19	22	21
	WIRR	93.33%	96.67%	96.67%
HotPotQA	WSN	20	15	16
	WIRR	100.00%	100.00%	96.67%

5.3.6 The Number of Injected Texts per Watermark Tuple. For a given question, the RAG system retrieves the top k most relevant texts. For each watermark tuple, we can generate multiple texts with the same semantics but different content, thereby increasing the proportion of watermark texts retrieved. We evaluate the impact of varying the number (i.e., N_{wm}) of injected watermark texts per tuple. Specifically, we set $k = 1$ (i.e., the number of retrieved most related texts) and assess watermark performance for N_{wm} values ranging from 1 to 5. Figure 6 shows the results. We observe that the WSN value increases as the number of injected watermark texts increases. This is because adding more watermark texts raises the likelihood of retrieving the watermark text, making the watermark relation more likely to appear in the LLM outputs. Therefore, we can generate multiple watermark texts for each watermark tuple and inject them into RAG to improve watermark performance. Moreover, adversaries cannot reliably detect multiple watermark texts for the same watermark tuple using duplicate text filtering, as demonstrated in Section 5.5.2). Additionally, CDPA remains stable for $N_{wm} = [1, 5]$, consistently above 95%, indicating good performance in maintaining the main task.



(a) TREC-COVID (b) NFCorpus (c) MS-MARCO

Figure 6: The Number of Injected Texts per Watermark Tuple.

5.4 Robustness

After stealing RAGs, attackers may attempt to remove the watermark. The adversary can utilize paraphrasing, removing unrelated content, inserting knowledge, and expanding knowledge attacks.

5.4.1 Paraphrasing Attack. Paraphrasing has been employed as a strategy to evade watermark detection in LLM-generated watermark texts [26, 25, 13, 25], thus it may be an effective technique to evade the watermark verification in RAG-WM. Specifically, when a watermark query is issued, the adversary can use an LLM to automatically paraphrase the retrieved texts from the watermarked knowledge database, thereby removing the watermark information and further evading verification. For paraphrasing, we follow the approach in [25], employing GPT-3.5-Turbo with the prompt "paraphrase the following sentences", a temperature setting of 0.7, and a maximum output length of 200 tokens. Table 5 shows the results of the attack. The average WSN value is 14, well above 2, indicating that RAG-WM remains detectable even after paraphrasing. This suggests that paraphrasing cannot effectively remove our RAG-WM. This is because paraphrasing can modify the wording and structure of the text, but it cannot alter the knowledge (i.e., the entities and relationship types) present in the content. In contrast, WARD [22], which is based on a red-green list watermark, is not robust against paraphrasing attacks. The adversary can paraphrase any word in the RAG texts by replacing words from the green list with those from the red list, rendering the WARD watermark ineffective. Following the default experimental settings of WARD, we evaluate its robustness against paraphrasing using the same paraphrasing prompt. WARD achieves an average z-score of -0.0384, below the threshold of 4, indicating failure in watermark verification. However, our RAG-WM is robust against this attack.

Table 5: Paraphrasing Attack

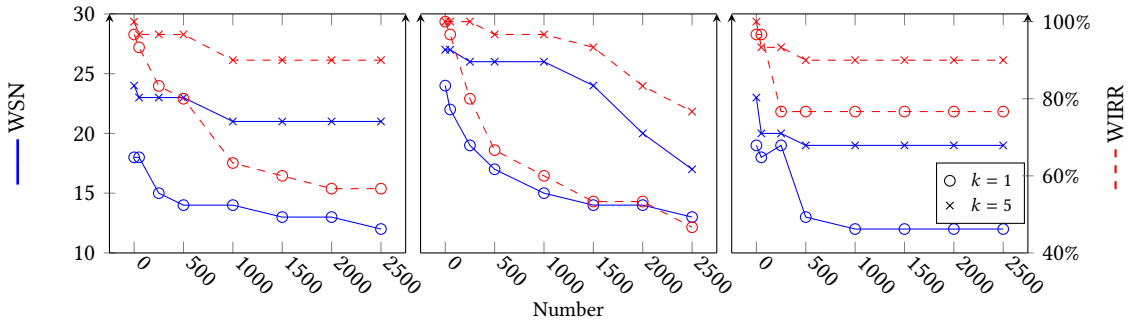
Dataset	TREC-COVID	NFCorpus	MS-MARCO
WSN	13	17	12

5.4.2 Unrelated Content Removal. Watermark content serves as extra information for detecting IP infringement. Such content may be removed by excluding specific sentences from the retrieved content in RAGs. Since removing content closely related to the main task would degrade performance, an adversary would desire to remove only unrelated content to minimize the impact on the main task. To attack RAG-WM, we adapt this method to remove such sentences from retrieved texts from the watermarked knowledge database for a watermark query. Specifically, GPT-3.5-Turbo is used to remove unrelated content based on a designed prompt (details in the extended version [33]). The experimental results are shown in Table 6. After the attack, the average WSN is 12, satisfying IP infringement detection.

Table 6: Unrelated Content Removal Attack

Dataset	TREC-COVID	NFCorpus	MS-MARCO
WSN	12	14	10

5.4.3 Knowledge Insertion Attack . Adversaries may insert misleading information (e.g., acting as noise) into a RAG's knowledge base to interfere with the watermark retrieval process. Since our watermark text encodes information about two entities and their relation, adversaries may want to maximize the interference by inserting records containing the watermark entities but with different relations. However, to avoid degrading the RAG's utility, they cannot indiscriminately inject numerous sentences with randomly selected entities and relations, as this would significantly downgrade the RAG's performance (discussed in Adaptive Attack of Section 6.2). A straightforward approach to inserting misleading knowledge while preserving utility is to generate attack texts by randomly correlating two texts from the watermarked knowledge base and injecting them directly. Note that this process may produce sentences that



(a) TREC-COVID (b) NFCorpus (c) MS-MARCO

Figure 7: Knowledge Insertion Attack.

include the watermark entities but introduce random relations between them due to the correlation of unrelated contents. When the owner later queries these watermark entities, RAG may retrieve the attack texts and respond based on the misleading correlation, thereby causing the watermark verification to fail.

To execute this attack, we vary the number of inserted texts from 0 to 2500 under default settings and retrieve the top relevant texts (evaluating both top 1 and top 5 retrieval cases) for each watermark query. The results are shown in Figure 7. In the top 1 case, with no texts injected, the WSN values for TREC-COVID, NFCorpus, and MS-MARCO are 18, 24, and 19, respectively. With 2,500 injected texts, the WSN values decrease to 12, 13, and 12. While the WSN decreases as the number of injected texts increases, it remains above 2, indicating that the attack does not compromise the IP infringement detection of RAG-WM. The decrease in WSN is mainly due to the interference from the injected texts, which affects the retrieval of watermark texts. For instance, when 2,500 texts are injected (approximately 70% of the original NFCorpus texts), the WIRR gradually drops from 100.00% to 46.67%. Furthermore, we observe that the decrease in WSN is smaller when retrieval is performed using the top 5 texts compared to the top 1 text. Specifically, WSN values decrease from 24, 27, and 23 (with no texts injected) to 21, 17, and 19 (with 2,500 texts injected). This indicates that RAG-WM is more robust against knowledge insertion attacks when the adversary retrieves more texts for the LLMs.

5.4.4 Knowledge Expansion Attack. The adversary can reduce the effectiveness of the watermark by increasing the proportion of non-watermarked information in the retrieved texts. Specifically, RAG-WM injects up to N_{wm} watermark texts into a knowledge database for each watermark query. If the adversary retrieves k texts, where $k > N_{wm}$, it is likely that at least $k - N_{wm}$ of these texts will be clean, thereby undermining RAG-WM's effectiveness. We evaluate this attack by varying the number of retrieved texts from 3 to 50. Due to the input text length limit of the LLM, watermark text verification cannot be performed when the text is too long. Therefore, we exclude Llama-2-7B and Vicuna-13B from our evaluation. Instead, we use PaLM-2 for the RAG system, which supports an input token limit of 8,192. Figure 8 shows the evaluation results. As the proportion of non-watermarked information increases, the WSN of RAG-WM remains stable. Even with $m = 50$, where only 10% of the retrieved texts are watermarked (with $k = 5$ watermark texts per query), RAG-WM maintains a minimum WSN of 19 on MS-MARCO. Additionally, this attack leads to significant computational costs on the LLM, as longer contexts require more resources to generate responses.

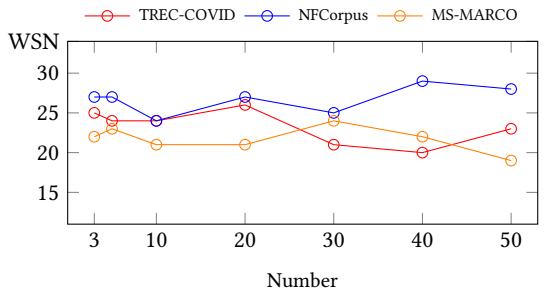


Figure 8: Knowledge Expansion Attack.

5.4.5 Combined Attacks. After stealing a RAG, the adversaries may try their best to undermine any watermark embedded inside using various techniques, such as knowledge insertion, knowledge expansion, paraphrasing, and unrelated content removal. We construct an advanced attack by combining these methods to simulate the adversaries' best effort and evaluate the robustness of RAG-WM against such combined attacks. Against the watermarked RAG, adversaries first perform a knowledge insertion attack by injecting 1,000 randomly generated texts (approximately four times the number of the original watermark texts). Next, adversaries apply knowledge expansion by retrieving ten texts. Finally, adversaries successively conduct paraphrasing and unrelated content removal attacks. The WSN values of the attacked RAGs in TREC-COVID, NFCorpus, and MS-MARCO are 8, 12, and 7, respectively, satisfying verification.

5.5 Stealthiness

Adversaries might use perplexity analysis or duplicate text filtering techniques to detect the watermark.

5.5.1 Detection by Perplexity. Text perplexity (PPL), commonly used to measure text quality, is the average negative log likelihood of each token's occurrence. A model's perplexity increases if a given sequence is disfluent, contains grammatical errors, or lacks logical coherence with prior inputs. PPL has served as a defense mechanism against attacks on LLMs [18, 5]. Considering that the embedding of watermark information may degrade the text quality of the RAG, adversaries could detect this low-quality content as suspicious watermark data. In our experiment, we calculate the perplexity of the Llama-2-7B model, as it is a white-box model.

We randomly select 2,000 clean texts and 200 watermark texts from the knowledge base. We calculate the perplexity values for both sets and apply the K-means algorithm to partition their PPL values into two clusters. The smaller cluster is identified as the outlier, corresponding to the watermark texts. The F1-scores for this evaluation are only 15.88%, 6.93%, 16.36% on the TREC-COVID, NFcorpus, and MS-MARCO tasks, respectively. We also analyze the frequency distribution of perplexity values for both sets, focusing on whether the perplexity of watermark texts is significantly higher than that of clean texts. The results are visualized as heatmaps in the extended version [33], with watermark texts highlighted in white. Interestingly, the perplexity distribution of watermark texts closely overlaps with regions of low perplexity and high-frequency values in clean texts. These evaluations show that watermark texts generated by RAG-WM exhibit high quality, making them difficult to distinguish from clean texts based solely on perplexity. As a result, using perplexity is not an effective detection method.

5.5.2 Duplicate Text Filtering. The owner may inject multiple watermark texts for each watermark tuple to improve the success rate of watermark retrieval in response to a watermark query. These watermark texts may be identical, which allows the adversary to filter out duplicate texts from the knowledge base, thus evading watermark verification. Referring to [64], we conduct experiments to filter duplicate texts on watermarked RAG systems. Specifically, for each watermark query, we compute the hash value (using the SHA-256 hash function) for each text in the top 50 retrieval results from the watermarked knowledge base and remove any texts with the same hash value. Table 7 compares the results with and without the attack. We observe that both WSN and WIRR remain unchanged after the attack, indicating that duplicate text filtering is ineffective at detecting and removing watermark texts. This is because the watermark texts generated by the multi-LLM interaction watermarking technique differ for each watermark tuple.

Table 7: Duplicate Text Filtering Attack

Dataset	TREC-COVID	NFCorpus	MS-MARCO
WSN	18	24	19
WIRR	96.67%	100.00%	96.67%

Table 8: Advanced RAGs

Dataset	Self-RAG		CRAG	
	WSN	WIRR	WSN	WIRR
TREC-WIRR	23	100.00%	23	96.67%
NFCorpus	26	100.00%	19	100.00%
MS-MARCO	23	100.00%	20	100.00%

5.6 Effectiveness in Advanced RAGs

The above experiments are evaluated against naive RAG systems. Recently, some advanced RAG techniques [7, 55] have been proposed, to solve the naive RAGs' disadvantages, e.g., retrieval challenges, generation difficulties, and augmentation hurdles. Referring to [64], we evaluate our RAG-WM in two commonly used advanced RAG systems, i.e., Self-RAG [7] and CRAG [55]. Table 8 shows that RAG-WM achieves high WSNs (average 22 WSN, well above the threshold of 2), demonstrating its ability to protect the IP of advanced RAGs. This effectiveness is due to the core idea behind these advanced RAG techniques: enhancing the relevance of retrieved texts to improve the accuracy of LLM-generated answers. Meanwhile, the crafted watermark texts are designed to be relevant to watermark queries, allowing the LLM to generate correct watermark relationships based on the retrieved watermark contexts.

6 Discussion

6.1 Watermark Injection Approach

We propose a watermark injection method based on relevant-text concatenation. Alternatively, a direct insertion approach can be considered, where the owner embeds the watermark text WT as a separate record in RAG_{wm}. This method has the advantage of introducing minimal disruption to the structure and content of the original RAG. However, it does not fully confirm that the watermark texts can be retrieved and detected during verification. In contrast, relevant-text concatenation injects WT into the most relevant text $TEXT$ of the RAG through pre-retrieval, improving detectability and extraction. To evaluate direct insertion, we generate the same number of watermark texts as in Section 5.2 using the multi-LLM interaction watermarking technique and directly insert them into RAG_{wm}. The results in Table 9 indicate that the WSN and WIRR values for direct insertion are lower than those for relevant-text concatenation. This is because the pre-retrieval step in the latter improves watermark retrieval performance and increases WSN values. Notably, the performance of direct insertion is poor in large-scale knowledge bases (e.g., NQ, HotpotQA, MSMARCO), and it worsens as the size of the base increases. This is because the vast amount of text in the knowledge base introduces more noise, while direct insertion lacks retrieval guarantees, making the watermark susceptible to interference from other texts.

Table 9: Direct Insertion Approach

Dataset	Metrics	LLMs			
		GPT-3.5	LLama	Vicuna	PaLM
TREC-COVID	WSN	13	14	15	12
	WIRR	80.00%	80.00%	80.00%	80.00%
NFCorpus	WSN	17	18	16	15
	WIRR	93.33%	93.33%	93.33%	93.33%
NQ	WSN	9	12	10	11
	WIRR	66.67%	66.67%	66.67%	66.67%
HotpotQA	WSN	8	9	8	9
	WIRR	56.67%	56.67%	56.67%	56.67%
MS-MARCO	WSN	7	8	7	7
	WIRR	46.67%	46.67%	46.67%	46.67%

6.2 Adaptive Attack

The attacker, aware of our watermarking mechanism, could manipulate the RAG system by injecting malicious texts. This manipulation may increase the likelihood that the malicious texts are retrieved instead of the intended watermark content in response to specific verification prompts (e.g., "What is the relationship between e^i_{wm} and e^j_{wm}?"), thereby bypassing the owner's verification process. However, based on the threat model, the adversaries cannot know our watermark entities and relations (i.e., the secret of owners), so they can only randomly select a wide range of entities and craft numerous malicious texts to bypass watermark verification. This will compromise the knowledge correctness of the original RAG, thus making it unusable. For example, consider the smallest knowledge base, NFCorpus, in our experiments, which has 38,194 entities and 41,763 relations. If the adversary randomly selects two entities and one relation to generate a sentence for injection, the number of possible combinations is $C(38{,}194, 2) \times 41{,}763$. Injecting such a large number of texts would significantly compromise the knowledge integrity of the original RAG (with 3,633 sentences). Moreover, if the owner injects 50 watermark tuples (i.e., 50 entity-relation

pairs), the same as our approach, the probability that a randomly generated malicious text matches a watermark pair by coincidence is $\frac{50}{C(38,194,2)} = 7 \times 10^{-8}$, which is nearly 0. Therefore, the adversary cannot bypass watermark verification without significantly degrading the RAG's performance. Additionally, our watermark queries are diverse, including both explicit and implicit queries, which makes it more challenging to craft malicious texts.

6.3 Knowledge Graph Distillation Attack

Since our watermark relies on the construction of watermark entities and relations, an adversary can launch a knowledge graph distillation attack against the watermarked domain-specific knowledge base (i.e., Domain-specific Knowledge Distillation Attack). This involves extracting entities and relations from the knowledge base and constructing a knowledge graph based on the extracted information. The adversary can then distill information from this graph by sorting entities according to their degree within the dataset corpus and generating dense subgraphs based on these entities and their relationships.

We evaluate this attack against NFCorpus task by generating subgraphs of different granularities for high-degree entities (with the entities' distillation (preservation) rate of 5%, 10%, 20%, 40%, 80%, 100%) and their corresponding relations. The evaluation results are shown in Figure 9. We can see that if the watermarked RAGs are distilled at a lower distillation rate, the RAGs have been distilled to be considered as "fail" on the main tasks (e.g., with 20% distillation rate, the CDPA is only 25.39%). The WSN of RAG-WM is 22, far larger than 2. When the distillation rate is high (above 80%), the performance of the main tasks is maintained. The WSN value of RAG-WM is far more extensive than 2, which can effectively detect IP infringement of the stolen RAG. This robustness arises because our watermark texts are generated based on high-frequency watermark entities and relations. Since the attack focus on distilling important entities and relations from the knowledge graph, the high degree of these watermark entities ensures their effective retention in the extracted graph, making the watermark robust against domain-specific knowledge distillation attack.

Additionally, for a general knowledge base with extensive domain knowledge (e.g., Wikidata [52], DBpedia [15]), the adversary might attempt to distill part of the knowledge base for the target domain. To counter this, we can refer to traditional relational database watermarking techniques to partition the knowledge base into groups [23], with each group containing domain-specific texts. Watermarks are then embedded in each group using RAG-WM. This partitioning approach enhances the robustness of our RAG-WM against this attack. RAG-WM is robust against such an attack.

6.4 Piracy Attack

Attackers can use our watermark embedding algorithm to insert a pirated watermark into stolen RAGs and fraudulently claim ownership. However, due to the robustness (Section 5.4) and stealthiness (Section 5.5) of our watermark, attackers cannot remove our watermark or create an RAG with only their own watermark. Thus, attackers can only present an RAG containing both their watermark and the owner's watermark, while the true owner can present an RAG with only their own watermark. It is clear who is the true owner.

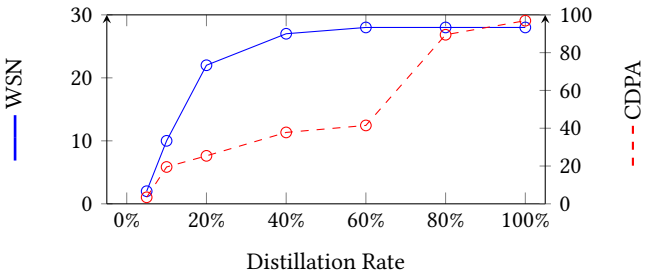


Figure 9: Knowledge Graph Distillation Attack.

7 Conclusion

In this paper, we propose a novel black-box "knowledge watermark" approach RAG-WM, to protect the intellectual property of RAGs. RAG-WM generates watermarked RAGs by leveraging a multi-LLM interaction watermarking technique, which creates watermark texts based on watermark entity-relationship tuples. These watermarks are then injected into the target RAG, and IP infringement is detected by querying the suspect LLM and RAG systems with the watermark queries in a black-box manner. Moreover, we evaluate our watermark on both domain-specific and privacy-sensitive tasks. The results demonstrate that our watermark can effectively detect IP infringement of RAGs in various adversary's deployed LLMs, and is robust against various watermark attacks.

Acknowledgments

This research is supported by the National Research Foundation, Singapore, and the Cyber Security Agency of Singapore under the National Cybersecurity R&D Programme and the CyberSG R&D Programme Office (Award CRPO-GC3-NTU-001), and the startup of Nanyang Technological University. Any opinions, findings, conclusions, or recommendations expressed in these materials are those of the authors and do not reflect the views of the National Research Foundation, Singapore, the Cyber Security Agency of Singapore, or the CyberSG R&D Programme Office. The IIE authors are supported in part by NSFC (92270204), CAS Project for Young Scientists in Basic Research (Grant No. YSBR-118). The Shandong University authors are supported in part by NSFC (62202275) and Natural Science Foundation of Shandong (ZR2022QF012).

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: watermarking deep neural networks by backdooring. In *27th USENIX security symposium (USENIX Security 18)*, 1615–1631.

[2] Rakesh Agrawal, Peter J Haas, and Jerry Kiernan. 2003. A system for watermarking relational databases. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 674–674.

[3] Rakesh Agrawal and Jerry Kiernan. 2002. Watermarking relational databases. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 155–166.

[4] Chroma AI. 2025. *Chroma*. https://docs.trychroma.com/.

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

[6] Maya Anderson, Guy Amit, and Abigail Goldsteen. 2024. Is my data in your retrieval database? membership inference attacks against retrieval augmented generation. *arXiv preprint arXiv:2405.20446*.

[7] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

[8] Mikhail J Atallah, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. 2001. Natural language watermarking: design, analysis, and a proof-of-concept implementation. In *Information Hiding: 4th International Workshop, IH 2001 Pittsburgh, PA, USA, April 25–27, 2001 Proceedings 4*. Springer, 185–200.

[9] Payal Bajaj et al. 2016. Ms marco: a human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

[10] Mahbuba Begum and Mohammad Shorif Uddin. 2020. Digital image watermarking techniques: a review. *Information*, 11, 2, 110.

[11] Sukriti Bhattacharya, Agostino Cortesi, et al. 2009. A distortion free watermark framework for relational databases. In *ICSOFT (2)*. Citeseer, 229–234.

[12] Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*. Springer, 716–722.

[13] Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*. PMLR, 1125–1139.

[14] 2025. *Code of RAG-WM*. https://github.com/873984419/ragwm.

[15] DBpedia Community. 2024. *DBpedia*. https://www.dbpedia.org/.

[16] Mintplex Labs Inc. 2025. *Anything LLM AI*. https://anythingllm.com/.

[17] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

[18] Neel Jain et al. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*.

[19] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

[20] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. 2021. Entangled watermarks as a defense against model extraction. In *30th USENIX security symposium (USENIX Security 21)*, 1937–1954.

[21] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

[22] Nikola Jovanović, Robin Staab, and et al. 2024. Ward: provable rag dataset inference via llm watermarks. *arXiv preprint arXiv:2410.03537*.

[23] Muhammad Kamran and Muddassar Farooq. 2018. A comprehensive survey of watermarking relational databases research. *arXiv preprint arXiv:1801.08271*.

[24] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*. PMLR, 17061–17084.

[25] John Kirchenbauer et al. 2023. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*.

[26] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2024. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36.

[27] Tom Kwiatkowski et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7, 453–466.

[28] John Snow Labs. 2025. *John Snow Labs*. https://www.johnsnowlabs.com/healthcare-llm/.

[29] Douglas B Lenat. 1995. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38, 11, 33–38.

[30] Mingchen Li, Halil Kilicoglu, Hua Xu, and Rui Zhang. 2024. Biomedrag: a retrieval augmented large language model for biomedicine. *arXiv preprint arXiv:2405.00465*.

[31] Yingjiu Li, Huiping Guo, and Sushil Jajodia. 2004. Tamper detection and localization for categorical data using fragile watermarks. In *Proceedings of the 4th ACM workshop on Digital rights management*, 73–82.

[32] Yuying Li, Gaoyang Liu, Chen Wang, and Yang Yang. 2024. Generating is believing: membership inference attacks against retrieval-augmented generation. *arXiv preprint arXiv:2406.19234*.

[33] Peizhuo Lv, Mengjie Sun, Hao Wang, Xiaofeng Wang, Shengzhi Zhang, Yuxuan Chen, Kai Chen, and Limin Sun. 2025. *Extended Version of RAG-WM*. https://sites.google.com/view/lvpeizhuo/publication/.

[34] Peizhuo Lv et al. 2024. Ssl-wm: a black-box watermarking approach for encoders pre-trained by self-supervised learning. In *Proceedings of the 2024 Annual Network and Distributed System Security Symposium, NDSS'24*.

[35] Hasan Mesut Meral, Bülent Sankur, A Sumru Özsoy, Tunga Güngör, and Emre Sevinç. 2009. Natural language watermarking via morphosyntactic alterations. *Computer Speech & Language*, 23, 1, 107–125.

[36] Meta. 2025. *Llama*. https://www.llama.com/.

[37] Meta. 2025. *Llama RAG*. https://ai.meta.com/blog/meta-llama-3-1/.

[38] Microsoft. 2025. *Azure*. https://learn.microsoft.com/zh-cn/azure/ai-studio/concepts/retrieval-augmented-generation.

[39] Travis Munyer, Abdullah Tanvir, Arjon Das, and Xin Zhong. 2023. Deeptextmark: a deep learning-driven text watermarking approach for identifying large language model generated text. *arXiv preprint arXiv:2305.05773*.

[40] OpenAI. 2025. *GPT*. https://openai.com/index/gpt-4/.

[41] Heiko Paulheim. 2018. How much is a triple. In *IEEE International Semantic Web Conference*.

[42] Saksham Rastogi and Danish Pruthi. 2024. Revisiting the robustness of watermarking to paraphrasing attacks. *arXiv preprint arXiv:2411.05277*.

[43] Ryoma Sato, Yuki Takezawa, Han Bao, Kenta Niwa, and Makoto Yamada. 2023. Embarrassingly simple text watermarks. *arXiv preprint arXiv:2310.08920*.

[44] Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. 2007. Watermarking relational databases using optimization-based techniques. *IEEE transactions on Knowledge and Data Engineering*, 20, 1, 116–129.

[45] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. 2003. Rights protection for relational data. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 98–109.

[46] Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11, 1–17.

[47] Weisong Sun, Yun Miao, Yuekang Li, Hongyu Zhang, Chunrong Fang, Yi Liu, Gelei Deng, Yang Liu, and Zhenyu Chen. 2025. Source code summarization in the era of large language models. In *Proceedings of the 47th International Conference on Software Engineering* number 1. IEEE Computer Society, Ottawa, Ontario, Canada, 419–431.

[48] Umut Topkara, Mercan Topkara, and Mikhail J Atallah. 2006. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th workshop on Multimedia and security*, 164–174.

[49] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

[50] Meng-Hsiun Tsai, Fang-Yu Hsu, Jun-Dong Chang, and Hsien-Chu Wu. 2007. Fragile database watermarking for malicious tamper detection using support vector regression. In *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*. Vol. 1. IEEE, 493–496.

[51] Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. Trec-covid: constructing a pandemic information retrieval test collection. In *ACM SIGIR Forum* number 1. Vol. 54. ACM New York, NY, USA, 1–12.

[52] 2024. *Wikipedia*. https://www.wikidata.org/wiki/Wikidata:Main_Page.

[53] Wikipedia. 2024. *Statistical Hypothesis Test*. https://en.wikipedia.org/wiki/Statistical_hypothesis_test.

[54] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

[55] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

[56] Xi Yang, Kejiang Chen, Weiming Zhang, Chang Liu, Yuang Qi, Jie Zhang, Han Fang, and Nenghai Yu. 2023. Watermarking text generated by black-box language models. *arXiv preprint arXiv:2305.08883*.

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

[58] Hanlin Zhang, Benjamin L Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. 2023. Watermarks in the sand: impossibility of strong watermarking for generative models. *arXiv preprint arXiv:2311.04378*.

[59] Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. 2024. {Remark-llm}: a robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, 1813–1830.

[60] Yue Zhang et al. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

[61] Zhi-hao Zhang, Xiao-Ming Jin, Jian-Min Wang, and De-Yi Li. 2004. Watermarking relational database using image. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826)*. Vol. 3. IEEE, 1739–1744.

[62] Huaqin Zhao et al. 2024. Revolutionizing finance with llms: an overview of applications and insights. *arXiv preprint arXiv:2401.11641*.

[63] Pengyuan Zhou, Lin Wang, Zhi Liu, Yanbin Hao, Pan Hui, Sasu Tarkoma, and Jussi Kangasharju. 2024. A survey on generative ai and llm for video generation, understanding, and streaming. *arXiv preprint arXiv:2404.16038*.

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