

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

BadGraph: Structural Knowledge Isolation Attacks against Graph Retrieval-Augmented Generation

Leiming Yan, Xinlong Xu, and Ziqiang Li, *Engineering Research Center of Digital Forensics, Ministry of Education, Nanjing University of Information Science & Technology*

https://www.usenix.org/conference/usenixsecurity26/presentation/yan-leiming

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ARTIFACT EVALUATED
USENIX
AVAILABLE

BadGraph: Structural Knowledge Isolation Attacks against Graph Retrieval-Augmented Generation

Leiming Yan[1,2], Xinlong Xu[1,2], Ziqiang Li[1,2*]
[1]*Nanjing University of Information Science & Technology*
[2]*Engineering Research Center of Digital Forensics, Ministry of Education, Nanjing University of Information Science & Technology, Nanjing, China*
lmyan@nuist.edu.cn, 202412492203@nuist.edu.cn
iceli@mail.ustc.edu.cn

Abstract

Graph Retrieval-Augmented Generation (GraphRAG) improves cross-document reasoning by introducing graph structures into retrieval. However, these graph structures also expose an under-studied topological attack surface. Existing research on GraphRAG attacks primarily focuses on targeted attacks based on text or relation manipulation, which induce the model to generate specific incorrect answers by injecting factual errors. These attacks often rely on explicit false-answer payloads, poisoned relations, or corpus rewriting, and their documents can be easier to flag during content auditing.

In this paper, we propose a targeted availability attack against GraphRAG: Structural Knowledge Isolation. Unlike traditional wrong-answer manipulation attacks, this attack targets the system's availability by degrading retrieval of critical evidence through topological manipulation. The resulting context can lack the evidence needed for grounded answer generation. We first mathematically analyze three topological vulnerabilities of graph algorithms used in GraphRAG systems when subjected to topological perturbations. Based on this analysis, we propose the BadGraph attack framework. By injecting a small amount of semantically neutral text, it generates adversarial subgraphs in the knowledge graph and reduces the visibility of target evidence in top-ranked retrieval contexts. Experiments show that with a small document injection budget (49 documents on HotpotQA and 30 on 2WikiMultiHopQA), BadGraph consistently lowers source recall and end-to-end QA F1 on three mainstream GraphRAG systems (MS-GraphRAG, LightRAG, and FastGraphRAG), while using neutral linker documents.

1 Introduction

Retrieval-Augmented Generation (RAG) effectively mitigates model hallucinations and knowledge obsolescence by combining Large Language Models (LLMs) with external knowledge bases [8, 17]. Traditional RAG systems primarily encode text into semantic vectors or lexical indexes, using dense or sparse retrieval methods such as BM25 to retrieve relevant segments. However, these methods have limitations for complex information needs: they struggle to capture logical associations across documents, can miss evidence that is semantically distant but crucial to the reasoning chain, and cannot capture global information from the knowledge base [48]. To address this issue, the Graph Retrieval-Augmented Generation (GraphRAG) paradigm has been popularized by systems such as MS-GraphRAG [6]. GraphRAG explicitly models associations between entities using graph structures, introducing graph-based retrieval algorithms such as next-hop neighborhood expansion, community summaries, and random-walk propagation, thereby improving multi-hop QA and global summarization.

The typical workflow of mainstream GraphRAG includes

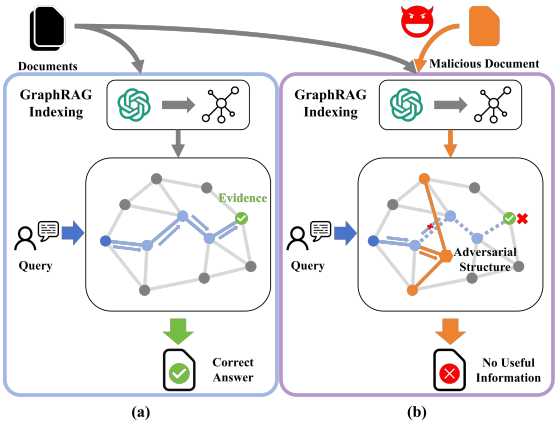


Figure 1: Illustration of Structural Knowledge Isolation Attack. GraphRAG builds graph-structured indexes and retrieves entities, relations, summaries, and source text. (a) Benign retrieval reaches key evidence along graph paths. (b) BadGraph injects neutral documents that form adversarial structures, redirecting graph retrieval and displacing the needed evidence.

*Corresponding author.

Indexing, Retrieval, and Generation. Unlike standard RAG, GraphRAG does not rely only on raw text chunks as retrieval units, but uses LLMs or dedicated neural models to extract nodes and edges, constructing a graph-structured knowledge base index, as shown in Figure 1(a); in the retrieval phase, query-based algorithms (such as neighborhood expansion, random walk) are used to recall relevant entities, relationships, and associated text chunks; finally, these graph units are fed to the LLM along with source text to generate the final answer for the user. The main differences across GraphRAG systems lie in the definition of graph structures and retrieval algorithms, such as neighborhood expansion, degree-based ranking, and random walks.

With the widespread deployment of GraphRAG, scrutiny regarding its security is intensifying, and recent studies have explored the feasibility and effectiveness of attacks against GraphRAG. Since GraphRAG systems rely on graph structures for knowledge base indexing and retrieval, attacks targeting traditional RAG systems may be disrupted by the knowledge graph generation process. GragPoison [22] constructs query-specific poisoned relationships and supporting entities so that injected false evidence competes with the original evidence during GraphRAG retrieval and generation. UKPA/TKPA [43] manipulate the reasoning path of specific systems by modifying text that determines graph connections. TKPA locates structurally important nodes and rewrites their associated passages to bias the answer, while UKPA perturbs reference and coreference cues in text, fragmenting entity connections and degrading reasoning performance. Although the implementation paths differ, these works change graph or text evidence so that the system produces attacker-preferred answers or degraded reasoning performance.

However, prior GraphRAG attack studies still leave two challenges:

- Stronger corpus-rewrite assumptions: Specifically, TKPA requires traversing the knowledge graph generated by GraphRAG to screen vulnerable nodes based on structural information, while both UKPA and TKPA rewrite the corresponding corpus passages. These assumptions are stronger than a document-injection setting in which an attacker can only add documents.

- Unclear mechanisms of graph structural weaknesses: Existing attack methods mainly determine the attack path by selecting poisoned relationships (GragPoison) or locating vulnerable nodes through structural features (UKPA/TKPA). However, there is still limited theoretical guidance on why different graph retrieval algorithms fail under topology-level perturbations.

To address these challenges, this paper proposes a **targeted availability attack**, termed **Structural Knowledge Isolation Attack**. The attack is targeted because the attacker knows the target queries, but it is not a wrong-answer attack: its goal is to prevent the system from retrieving the evidence needed to answer those queries, thereby reducing availability rather than generating a designated wrong answer. The main idea is that the attacker generates adversarial structures in the graph and exploits topological weaknesses of graph retrieval algorithms to divert retrieval away from the target evidence. Once the target evidence is excluded from the top-ranked retrieval context, the GraphRAG system lacks the evidence needed for the question and may refuse to answer or hallucinate unsupported content.

This paper investigates topological vulnerabilities in graph algorithms (degree sorting, Leiden, PPR) used by mainstream GraphRAG systems from a mathematical perspective. Based on this analysis, we propose the BadGraph attack framework. BadGraph predicts injection anchors based on target queries, constructs adversarial subgraphs, and transforms these structures into semantically neutral documents. When these documents enter the knowledge graph through the GraphRAG system's indexing process, they divert retrieval away from the target evidence. BadGraph differs from wrong-answer poisoning in two main ways: (1) the attacker requires neither knowledge of the graph structure nor privileges to delete or rewrite the knowledge base; and (2) unlike wrong-answer attacks that rely on texts containing factual errors, the attack payload of BadGraph consists of semantically neutral text and avoids explicit factual errors. This structure-based poisoning approach makes the payload less directly exposed to content-only checks that look for explicit false answers or malicious instructions.

The main contributions of this paper are as follows:

- We formally define the threat of "Structural Knowledge Isolation" against GraphRAG, and analyze the characteristics of mainstream graph retrieval algorithms (such as neighborhood expansion, community detection, and random-walk propagation) from a mathematical perspective, extracting three topological vulnerabilities: structural dominance in ranking, resolution limit of community detection, and rank sink of random walks, revealing the potential risks of graph-augmented retrieval in adversarial environments.

- We propose a "gray-box" attack framework, BadGraph. By constructing adversarial graph-structure plans and realizing them as semantically neutral linker documents (including bridging and expansion components), BadGraph leverages topological vulnerabilities to reduce the visibility of target evidence in top-ranked retrieval contexts.

- Experiments on three mainstream GraphRAG systems show that BadGraph lowers both source recall and end-to-end QA F1 under a small document injection budget, including large drops on HotpotQA across MS-GraphRAG, LightRAG, and FastGraphRAG.

Table 1: Paradigm Classification of GraphRAG and Related Structured-Retrieval Systems

Paradigm	Representative Works	Core Mechanisms
I. Topological Neighborhood Retrieval	MS-GraphRAG [6], LightRAG [11], KGP [38], PG-RAG [24], GNN-RAG [5], MedGraphRAG [44], KAG [23], AGRAG [39], G-Retriever [14], HybridRAG [32], E^2GraphRAG [50], MemoRAG [30], GEM-RAG [31], GFM-RAG [26], ROGRAG [40]	Entity/Relation Graphs, Neighborhood Expansion, Community or Subgraph Retrieval, Structured Memory, Hybrid Retrieval
II. Probability Flow Retrieval	FastGraphRAG [4], HippoRAG [12], HippoRAG 2 [13], LinearRAG [51], PropRAG [37]	Random Walk (PPR), Dynamic Activation, Path Search, Linearized or Probability Propagation
III. Model-Based Reasoning Retrieval	LogicRAG [2], Graph-CoT [16], RoG [25], ToG [27, 34], StructRAG [19], KnowGPT [49], GraphReader [18], Speculative RAG [41], IR-CoT [36]	LLM-guided Reasoning, Path Planning, Proposition Paths, Chain-of-Thought (CoT) [42]

2 Background and Related Work

2.1 RAG and RAG Security

Retrieval-Augmented Generation (RAG) [17] mitigates LLM hallucinations by retrieving external knowledge. The classic Naive RAG retrieves Top-K relevant segments via semantic similarity and feeds them to the generator. However, this reliance on semantic matching creates attack surfaces.

Existing attacks focus on **text-content and retrieval manipulation**. **Knowledge Poisoning** attacks [9, 52] inject malicious documents to mislead generation. **Backdoor & Poisoning-Efficiency** studies [3, 20, 21, 45] study trigger amplification, proxy-attack design, and data selection, while others explore Denial of Service (DoS) [33] or opinion manipulation [10]. These methods confirm that attackers can compromise Naive RAG by manipulating text features or retrieved context.

2.2 GraphRAG

To address the limitation of Naive RAG in capturing structured associations [48], GraphRAG introduces graph structures to model entity relationships. For threat analysis, we group GraphRAG and closely related structured-retrieval systems into three paradigms (Table 1):

Paradigm I: Topological Neighborhood Retrieval. Systems like MS-GraphRAG [6] and LightRAG [11] adopt local expansion strategies. MS-GraphRAG employs **Global Search** over community reports and **Local Search** that assembles context from query-linked entities, relationships, and text units. LightRAG uses dual-level indexing and retrieves entities, relationships, and associated chunks with both graph and vector signals. Other systems in this family build persistent entity/relation graphs, subgraph indexes, or hybrid graph-vector contexts, including KGP, PG-RAG, GNN-RAG, MedGraphRAG, KAG, AGRAG, G-Retriever, HybridRAG, E^2GraphRAG, MemoRAG, GEM-RAG, GFM-RAG, and ROGRAG. For the BadGraph threat model, this grouping follows the attack-relevant retrieval layer rather than every auxiliary component. In particular, G-Retriever retrieves compact textual subgraphs from a graph index, while ROGRAG combines dual-level graph retrieval with logic-form retrieval; we classify it by its persistent graph-index retrieval layer. Core to this paradigm is the use of persistent structured indexes (e.g., neighborhoods, relations, communities, node degree, retrieved subgraphs, structured memories, or hybrid graph-vector contexts) to guide retrieval.

Paradigm II: Probability Flow Retrieval. FastGraphRAG [4] exemplifies this by mapping queries to seed entities and simulating importance flow via Personalized PageRank (PPR). The resulting steady-state probabilities propagate to text chunks, discovering structurally relevant but semantically distant associations. Related works like HippoRAG [12], HippoRAG 2 [13], LinearRAG [51], and PropRAG [37] employ related memory, activation, path-search, or graph-propagation mechanisms. PropRAG is included here because its online retrieval is an LLM-free beam search over paths in a pre-built proposition graph, not a query-time LLM-reasoning graph.

Paradigm III: Model-Based Reasoning Retrieval. Systems such as LogicRAG, Graph-CoT, RoG, ToG, StructRAG, KnowGPT, GraphReader, Speculative RAG, and IR-CoT are discussed as GraphRAG or related reasoning-oriented retrieval methods in the literature. However, their retrieval is primarily controlled by an inference-time reasoning process or an external KG traversed by an LLM-guided planner (e.g., dependencies among sub-questions, retrieval steps, and reflections), rather than a fixed graph-retrieval rule over the document-derived graph index.

Paradigms I and II share three components: (1) **Indexing** constructs $G = (V, E)$ from $\mathcal{D}$; (2) **Retrieval** recalls subgraph C via algorithms (e.g., PPR); (3) **Generation** produces answers based on C. BadGraph focuses on these persistent graph-index systems because corpus injection can alter the indexed topology before retrieval. Paradigm III has a different attack surface and is therefore outside our direct scope.

2.3 GraphRAG Security

Specific attacks targeting GraphRAG have recently emerged. **Graph Poisoning** manipulates outcomes via injection. GragPoison [22] injects query-specific poisoned relations and supporting entities so false evidence can compete with benign evidence during generation. UKPA/TKPA [43] rewrite source passages or chunks in the original corpus and therefore represent a corpus-rewrite threat model distinct from BadGraph's document-injection setting. We include these attacks as text-rewrite baselines in our evaluation. **Graph Extraction** attacks like AGEA [46] aim to infer or extract the graph index structure via black-box queries.

3 Threat Model

3.1 Envisioned Attacker

We consider a **gray-box** attacker.

Goal: The attacker aims to execute **Structural Knowledge Isolation**: preventing the GraphRAG system from retrieving the evidence needed to answer target queries. Unlike wrong-answer poisoning, BadGraph does not need to inject an explicit false answer. Its objective is to make the correct evidence unavailable to the generator, potentially causing wrong answers, refusals, or unsupported responses.

Capabilities: The attacker can add documents. The attacker knows the target queries and uses them to infer target entities and semantic anchors, but does not need to observe live user queries. This assumption follows prior query-targeted RAG poisoning work: PoisonedRAG [52] targets attacker-chosen questions, TKPA [43] starts from a user query, and GragPoison [22] optimizes over a fixed target-query set. The attacker does not need the full benign corpus, the constructed graph, exact model parameters, or any privilege to modify/delete existing documents.

The gray-box knowledge required by BadGraph is coarse: the attacker needs to know that the target system builds a persistent graph index and that the indexed corpus is expected to contain evidence related to the target queries. This knowledge excludes exact corpus contents, graph topology, retrieval seeds, and model parameters. The attack pipeline predicts likely semantic anchors, generates neutral linker documents, and relies on normal indexing to extract adversarial nodes and edges. Such coarse knowledge can often be inferred from the application domain, product documentation, open-source implementations, deployment behavior, or limited probing.

Constraints: The attacker **cannot modify or delete** existing documents in the knowledge base. The injected content is also kept small and semantically neutral, which limits surface-level content-quality anomalies and avoids explicit false-answer payloads.

3.2 Threat Surfaces

This attack primarily targets the **Indexing Component** and **Retrieval Component** of Paradigm I and II systems. BadGraph manipulates the topology of the persistent graph by polluting the corpus and exploits mathematical vulnerabilities of graph algorithms, degrading the retrieval process. Paradigm III systems use query-time reasoning or external graph traversal rather than maintaining a corpus-level graph index, so they are not directly vulnerable to this graph-index poisoning surface.

3.3 Generality

BadGraph does not target a single system but exploits the following mathematical properties of graph algorithms used by mainstream GraphRAG systems:

Structural Dominance: Affects systems based on neighborhood expansion (Paradigm I).

Resolution Limit: Affects systems in Paradigm I that use modularity optimization for community detection and summarization.

Rank Sink: Affects systems based on random walks (Paradigm II).

3.4 Practicality

Low Cost: BadGraph injects only a small number of documents. In the main setting, this corresponds to 49 injected documents for HotpotQA and 30 injected documents for 2WikiMultiHopQA.

Payload Naturalness: The injected texts are semantically benign, general-domain documents containing no explicit factual errors. This reduces surface-level semantic-consistency alarms and high-perplexity text anomalies.

4 Topological Vulnerability Analysis

This section analyzes failure mechanisms of statistical algorithms used by GraphRAG in adversarial environments and motivates the design of attack strategies. We conduct mathematical analyses on three algorithms: retrieval ranking, community detection, and probability propagation, identifying the topological vulnerabilities of mainstream graph algorithms used by GraphRAG, as shown in Figure 2 (detailed mathematical derivations can be found in Appendix A).

4.1 Vulnerability I: Structural Dominance in Ranking

In the Topological Neighborhood Retrieval paradigm (such as MS-GraphRAG Local Search and LightRAG Local Search), systems often adopt heuristic rules to approximate node importance. Although initial entity recall is typically based on

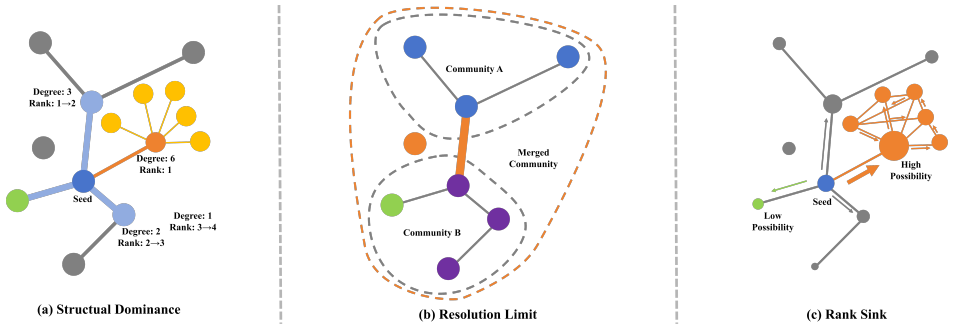


Figure 2: Schematic of Topological Vulnerabilities in Graph Algorithms. (a) **Structural Dominance in Ranking**: A high-degree adversarial hub (orange) artificially boosts its rank score (Rank 1) relying solely on connection advantages, thereby suppressing the ranking of semantic seed nodes (blue) and target nodes (green). (b) **Resolution Limit**: A small set of adversarial bridge nodes and edges links the target region to an unrelated dense region, making modularity optimization more likely to absorb the target into a larger community and dilute target features. (c) **Rank Sink**: A tightly connected adversarial subgraph (orange area) acts as a probability sink. When the random walk (blue arrow) reaches the entry node, the probability mass is trapped within the subgraph's internal cycle (orange arrows), preventing probability flow to the target node (green).

semantic similarity, in the subsequent **Context Expansion** and **Relationship Retrieval** phases, systems rely heavily on structural metrics to filter and rank neighbors.

Given a user query q, the effective score $f(x)$ of a candidate evidence (node or edge) x in the final context can be modeled as a weighted combination of semantic similarity and structural importance:

$$f(x) = \text{sim}(q,x) \cdot S(x)^{\beta} \tag{1}$$

where $\text{sim}(q,x) \in [0,1]$ represents the semantic similarity between the query and content; $S(x)$ is the structural importance metric. According to our source code analysis, both LightRAG and MS-GraphRAG adopt **Node Degree** $d(v)$ as the primary metric for $S(x)$ in this step (manifested as the `rank` attribute of relationships in MS-GraphRAG, and `edge_degree` in LightRAG); $\beta > 0$ is a hyperparameter regulating the strength of structural bias.

If an attacker injects an adversarial hub node v_{hub} into the graph G, even if the semantic similarity between this node and the query q is extremely low ($\text{sim}(q, v_{hub}) \to \varepsilon$), the attacker can still increase its ranking score $f(v_{hub})$ by artificially increasing its degree $d(v_{hub})$. As long as:

$$d(v_{hub}) > \left(\frac{T}{\varepsilon}\right)^{1/\beta} \tag{2}$$

where T is the score threshold required to enter Top-K, v_{hub} can compensate for "semantic disadvantage" with "structural advantage" and enter the Top-K window. This **Structural Dominance** phenomenon exposes an attack surface in Paradigm I: retrieval can over-weight graph connectivity and does not inherently distinguish "natural high connectivity" from "artificially injected high connectivity".

A hard semantic threshold is not a reliable defense against this mechanism. First, GraphRAG is designed to retrieve structurally necessary evidence that can be semantically distant from the query, so strict semantic filtering conflicts with graph retrieval. Second, BadGraph attaches injected edges to semantically related anchors, rather than to clearly irrelevant text. Therefore, semantic thresholding either removes useful graph evidence or still permits the attack-linked region to enter the candidate context.

4.2 Vulnerability II: Resolution Limit in Community Detection

MS-GraphRAG's community-report layer relies on community summaries. The system first uses the Leiden algorithm [35] to perform hierarchical clustering on graph G, and then generates a text summary for each community C_i. The Leiden algorithm essentially optimizes Modularity Q [28]. For a graph $G = (V,E)$, modularity is defined as:

$$Q = \frac{1}{2m}\sum_{u,v}\left(A_{uv} - \frac{k_u k_v}{2m}\right)\delta(c_u, c_v) \tag{3}$$

where A_{uv} is the adjacency matrix weight, k_u is the degree of node u, $m = \frac{1}{2}\sum_u k_u$ is the total number of edges, and $\delta(c_u, c_v)$ is the Kronecker delta function (1 if nodes u, v belong to the same community, 0 otherwise).

Fortunato and Barthelemy [7] proved that the modularity optimization objective has an intrinsic resolution limit, meaning the algorithm cannot identify community structures smaller than a specific scale. This limitation is particularly significant in large-scale graphs. Let the total number of edges in the graph be m, and consider two semantically independent regions C_i (containing the target entity) and C_j (an unrelated dense region reached through adversarial bridge nodes). The gain ΔQ_{ij} determined by the Leiden algorithm for merging

these two communities is:

$$\Delta Q_{ij} = \underbrace{\frac{l_{ij}}{m}}_{\text{Empirical Link Gain}} \underbrace{-2\left(\frac{K_i}{2m}\right)\left(\frac{K_j}{2m}\right)}_{\text{Null Model Penalty}} \tag{4}$$

where l_{ij} is the number of connecting edges between the two communities; $K_i = \sum_{v \in C_i} d(v)$ is the sum of degrees of nodes within community C_i; the second term is the **Null Model Penalty**, representing the expected connection probability between the two communities in a random graph (Configuration Model) preserving the degree distribution.

When $\Delta Q_{ij} > 0$, the modularity objective favors merging the two communities. The corresponding threshold is:

$$l_{ij} > \frac{K_i K_j}{2m} \tag{5}$$

Analyzing the above inequality, the merge decision depends on cross-community bridge edges l_{ij}, degree sums K_i and K_j, and global graph scale m. Thus the attack is not a simple rule that "enough nodes always merge." BadGraph does not need to inject a large standalone community or force the target to merge with a specific destination community. It introduces a small number of bridge nodes and relations near the target so that modularity optimization is more likely to absorb the target region into a denser, less relevant region. The exact merge outcome depends on bridge placement and the local topology around the target.

Such merging can lead to probability dilution in the retrieval phase. When generating community summaries, the system is limited by the LLM's context window and can only sample Top-N nodes. The merged community C_{merged} contains a large number of irrelevant nodes, causing the sampling probability $P(v|C_{merged})$ of the target node v to be significantly lower than its probability $P(v|C_i)$ in the original community, thereby causing information loss.

4.3 Vulnerability III: Rank Sink in Random Walks

In the Probability Flow Retrieval paradigm (such as Fast-GraphRAG), the system uses the Personalized PageRank (PPR) algorithm to measure the global importance of nodes relative to query q. Formally, the PPR value $R(u)$ of node u in graph $G = (V, E)$ satisfies the recursive definition [1]:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} + (1-c)E(u) \tag{6}$$

where $R(v)$ represents the PageRank value of node v, $B_u \subseteq V$ is the set of backward neighbors (Backlink Set) pointing to node u, $N_v = |\{k \in V | (v,k) \in E\}|$ is the out-degree of node v, $E(u)$ is the personalization vector ($E(u) > 0$ when node u is relevant to the query, 0 otherwise), and c is the damping factor (usually 0.85).

Page et al. pointed out when proposing PageRank that if there exists a subset of nodes $S \subseteq V$ whose outgoing edges almost exclusively point to within the set (i.e., $\sum_{u \in S, v \notin S} w_{uv} \approx 0$), this set constitutes a **Rank Sink**. In benign graphs, Rank Sinks cause abnormal aggregation of global ranking weights, usually mitigated by the $1-c$ teleportation probability.

However, in an adversarial setting, an attacker can actively construct an adversarial subgraph $S_{adv} \subset V$ that approximates a Rank Sink. In this case, the probability flow entering S_{adv} is difficult to escape, causing significant **Rank Accumulation** within S_{adv}. For the adversarial hub v_{hub} inside S_{adv}, its steady-state probability approximately satisfies:

$$R(v_{hub}) \propto \frac{1}{1 - c \cdot (1 - \phi_{out})} \tag{7}$$

where $\phi_{out} \in [0,1]$ is the outflow rate of the subgraph, representing the probability of jumping from S_{adv} to external nodes. When the attacker constructs a highly closed structure ($\phi_{out} \to 0$), the denominator approaches $1 - c \approx 0.15$, giving $R(v_{hub})$ a theoretical amplification gain of about $1/(1-c) \approx 6.6$.

Since the retrieval logic of actual systems truncates the Top-K ranking list ($V_{retrieval} = \text{Top}_K(\{R(v)\}_{v \in V})$), the adversarial inflation of $R(v_{hub})$ compresses the dynamic range of the entire probability distribution, causing the relative ranking $Rank(v_{target})$ of the originally marginal real target node v_{target} to drop, eventually falling out of the Top-K recall window, achieving implicit knowledge isolation.

Rank sinks are a known PageRank failure mode; our claim is not that the graph-theoretic phenomenon is new. The security contribution is showing that injected linker documents can induce this known failure mode inside GraphRAG indexing and retrieval, causing target evidence to fall out of the final context.

5 BadGraph

5.1 Attack Flow Overview

BadGraph executes in three cascaded phases (Figure 3). **Phase I** predicts likely retrieval anchors from the target query to increase coverage of possible retrieval seeds. **Phase II** constructs a **Trap Component**, which forms high-degree or low-conductance neighborhoods that affect local relationship ranking and probability-flow retrieval, and a **Confusion Component**, which adds bridge relations around the same anchor region to dilute community-level summaries. **Phase III** transforms these structural connections into natural language "linker documents" for corpus injection.

5.2 Phase I: Anchor Prediction

Under the gray-box assumption, BadGraph uses the LLM's general knowledge together with limited background infor-

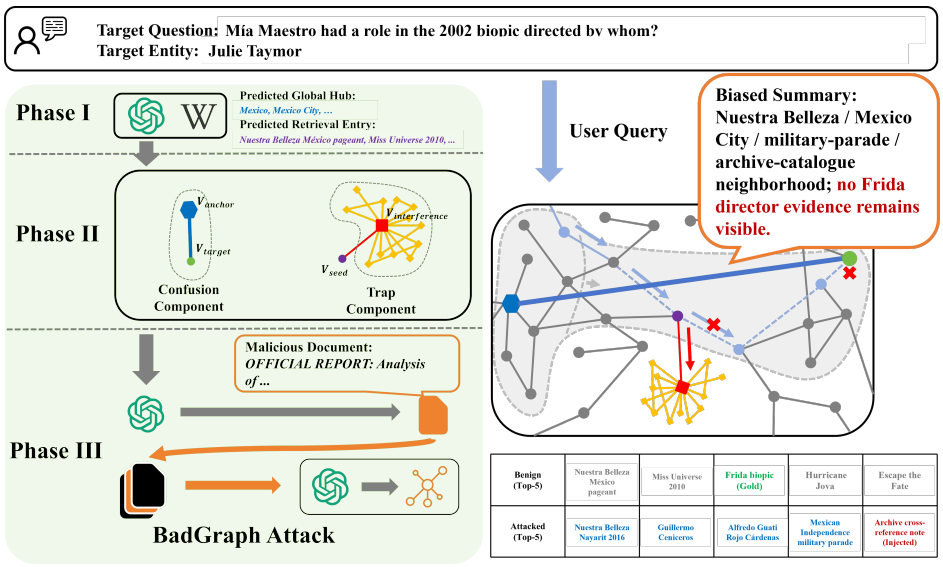


Figure 3: Overview of BadGraph Attack Framework. **Left**: The extraction-based attack flow (Phases I-III), transforming predicted anchors into injected linker documents. **Right**: Attack effectiveness mechanism. (1) **Gray Area**: The **Confusion Component** adds bridge relations around the target, causing the target evidence to be absorbed into a broader, less relevant community and omitted during summary generation due to information dilution. (2) **Path Hijacking**: The **Red Arrow** shows the **Trap Component** intercepting retrieval traffic (originally the **Blue Arrow**) via structural dominance, diverting it to a trap. (3) **Table Comparison**: Post-attack results (bottom right) show target evidence being displaced by high-weight interference nodes.

mation, when available, to infer likely entity neighborhoods that may appear in the indexed corpus, thereby identifying two types of heterogeneous anchors:

Global Hubs V_{hub}: BadGraph identifies general concepts with high co-occurrence frequency in the public corpus (e.g., "United States", "Science"). These nodes typically have high degree centrality in the graph and often influence Leiden community partitioning; they are used for deploying the **Confusion Component**.

Retrieval Entry V_{seed}: BadGraph uses the LLM's semantic reasoning capability to extract a set of entities from the target query that might be used by the GraphRAG system as retrieval starting points. Since the attacker cannot predict the system's precise retrieval entries, we adopt a multi-point coverage strategy and predict up to 10 candidate seeds to increase the likelihood that the injected structure is reached. These nodes are used for deploying the **Trap Component**.

5.3 Phase II: Adversarial Subgraph Construction

Based on these two types of anchors, BadGraph constructs two adversarial topological components corresponding to the three vulnerabilities analyzed in Section 4.

Trap Component—Exploiting Vulnerability I & III: This component is deployed at the **Retrieval Entry** V_{seed}, consuming the main attack budget. Its goal is to construct a structural trap S_{trap} with a low outflow rate or a high-rank local neighborhood by building high-density adversarial hub nodes around each V_{seed}. Targeting Paradigm I's **Topological Neighborhood Retrieval** (such as LightRAG, MS-GraphRAG), this component exploits **Vulnerability I (Structural Dominance)**: the high topological degree of the adversarial hubs and their support nodes satisfies the ranking condition in Eq. (2), allowing injected units to compete for top-ranked context positions and displace target evidence. Targeting Paradigm II's **Probability Flow Retrieval** (such as FastGraphRAG), this component exploits **Vulnerability III (Rank Sink)**: the low outflow rate of S_{trap} satisfies the Rank Sink condition in Eq. (7), capturing the probability mass passing through the seed and reducing the relative ranking of target evidence.

Confusion Component—Exploiting Vulnerability II: Unlike the Trap Component, the Confusion Component is used mainly for MS-GraphRAG's **Community Summary Retrieval**. It introduces neutral bridge nodes and relations near target-related anchors so that the target facts are placed in a larger or less focused community context. This exploits **Vulnerability II (Resolution Limit)** by increasing cross-region connectivity l_{ij} in Eq. (5) and changing the local degree pattern around the target region. BadGraph does not require injecting an entire adversarial community, nor does it require a deterministic merge into a specific destination community. During community-report generation, the target information is more likely to be omitted because it occupies a smaller fraction of the summarized region. This bridge structure is

conceptually separate from the high-degree trap: the former suppresses community-level background summaries, while the latter competes in local entity/relationship ranking.

5.4 Phase III: Linker Document Generation

To covertly generate the aforementioned adversarial structures within the graph, BadGraph uses an LLM to generate linker documents (the core prompt templates are listed in Appendix B, and the released notebook implements the corresponding generation workflow). The attack framework induces the target system to establish pre-set edge connections during index construction by generating natural text that simultaneously mentions specific entity pairs and establishes reasonable semantic associations. These documents also enter the index as source units and can be used by the GraphRAG pipeline to synthesize entity and relationship descriptions. Thus, the attack affects both the graph topology and the textual descriptions attached to newly indexed nodes and edges, while still avoiding explicit false answers. For the **Confusion Component**, BadGraph generates text mentioning target-related anchors and neutral bridge nodes; for the **Trap Component**, BadGraph generates text mentioning the predicted seed and the adversarial hub, as illustrated below:

Linker Document Generation Example

"...regarding [Target], it is often discussed in the context of broader topics like [Global Hub]..."

This approach exploits the trust that GraphRAG systems place in text co-occurrence relationships, allowing adversarial edges to be extracted during index construction.

6 Experiments

6.1 Research Questions

To evaluate the effectiveness of BadGraph in knowledge isolation tasks, this paper proposes four research questions, focusing on its attack efficacy, low-budget natural text generation, and robustness against defenses in gray-box scenarios:

RQ1 : Can BadGraph reduce source recall and downstream QA quality under strict budget constraints?

RQ2 : Can BadGraph remain low-budget and natural in text form while still reducing source recall?

RQ3 : Do key system hyperparameters (e.g., retrieval seed budget, damping factor, construction temperature) significantly affect the robustness of GraphRAG systems against topological perturbations?

RQ4 : Can existing general defense strategies (e.g., degree truncation, community regularization) effectively mitigate knowledge isolation caused by BadGraph?

6.2 Datasets and Graph Construction

We select **2WikiMultiHopQA** [15] and **HotpotQA** [47] as evaluation benchmarks, which are widely used in graph-based RAG and memory-style retrieval research, including HippoRAG 2 [13]. We aggregate all candidate paragraphs of each dataset into a unified retrieval corpus and randomly sample 1,000 QA pairs from the validation set as the core test set.

We deploy three GraphRAG systems representing two retrieval paradigms: (1) **MS-GraphRAG** (Microsoft version): A Paradigm I system whose Local Search mechanism strictly limits the retrieval scope to the direct 1-hop neighbors of seed entities; (2) **LightRAG**: Also in Paradigm I, extending retrieval results to the local neighborhood through explicit edge queries and degree ranking; (3) **FastGraphRAG**: A Paradigm II system performing probabilistic walk retrieval based on the Personalized PageRank (PPR) algorithm. All systems uniformly use `GPT-4o-mini` [29] (temperature set to 0.1) for graph construction. Detailed parameter settings for these systems are provided in Table C.1 in Appendix C.

As shown in Table 2, the graph construction algorithms of different systems lead to significant differences in graph scale.

Table 2: Graph Index Statistics. $|V|$ and $|E|$ represent the number of nodes and edges, respectively. Density is the graph density. These are descriptive index statistics, not higher/lower-is-better metrics.

Dataset	System	$\lvert V\rvert$	$\lvert E\rvert$	Density
2Wiki	MS-GraphRAG	21,791	183,501	3.8×10^{-4}
	LightRAG	580,214	645,716	1.9×10^{-6}
	FastGraphRAG	51,732	109,692	4.1×10^{-5}
HotpotQA	MS-GraphRAG	133,025	1,100,532	6.2×10^{-5}
	LightRAG	2,912,945	3,254,498	3.8×10^{-7}
	FastGraphRAG	392,028	676,558	4.4×10^{-6}

6.3 Attack Configuration and Baselines

We conduct the experiments in the gray-box knowledge-isolation setting defined in Section 3.

Adversarial Budget. BadGraph injects text documents into the corpus. In the main setting, the document injection budget is 0.5% of the corpus, corresponding to 49 documents for HotpotQA and 30 documents for 2Wiki. Under the multi-anchor strategy, these documents are distributed over the semantic anchor set.

Target and Anchor Selection. For each target query, the attacker identifies an evidence entity or topic region to isolate and infers up to 10 likely semantic anchors from the query context. These anchors are entities or topics that a normal

Table 3: Attack and baseline results. Each method is split into Source Recall and QA F1, reported as percentages. QA answers are generated with GPT-4o-mini. BadGraph uses the main document budget (49 HotpotQA and 30 2Wiki documents). For attack columns, lower values indicate stronger degradation; **bold** and underline mark the best and second-best attack results for each metric within each row, excluding Clean.

Dataset	System	Clean		Random		IDK		Wrong-Answer		GragPoison				TKPA		UKPA		BadGraph	
										Default		Budget-Matched							
		Recall↓	F1↓	Recall↓	F1↓	Recall↓	F1↓	Recall↓	F1↓	Recall↓	F1↓	Recall↓	F1↓	Recall↓	F1↓	Recall↓	F1↓	Recall↓	F1↓
2Wiki	MS-GraphRAG	81.50	12.57	81.50	12.68	81.50	12.85	81.50	13.14	81.50	11.70	81.50	12.35	78.12	8.43	78.38	10.95	**42.00**	**7.26**
	LightRAG	57.38	9.17	57.38	9.23	57.38	8.92	57.38	9.47	57.25	9.53	57.25	9.16	56.50	8.62	56.50	8.17	**39.75**	**5.55**
	FastGraphRAG	75.62	33.67	75.62	35.38	75.62	33.73	75.62	33.88	71.12	33.29	70.88	32.45	73.62	18.90	73.88	30.12	**8.75**	**8.73**
HotpotQA	MS-GraphRAG	88.75	17.31	88.75	17.32	88.75	17.15	88.75	17.43	88.75	16.50	88.75	17.03	84.50	15.69	86.00	18.39	**0.00**	**7.26**
	LightRAG	44.50	8.80	44.50	8.57	44.50	8.72	44.50	8.82	44.50	8.75	44.50	8.37	43.25	7.06	42.75	9.66	**16.50**	**4.82**
	FastGraphRAG	59.00	36.16	59.00	36.12	59.00	36.61	59.00	34.52	56.50	35.16	54.50	35.25	55.25	28.62	53.00	39.05	**4.50**	**21.79**

GraphRAG index is likely to contain and that can serve as entry points into the target neighborhood. BadGraph generates neutral linker documents around these anchors and lets the normal indexing process extract the corresponding nodes and relationships. Once indexed, these structures participate in the system's ordinary graph retrieval, where they can affect local expansion, community context, or probability-flow ranking without changing the retrieval interface. The anchors provide semantic entry points, while the injected linker documents supply the structural connections that make the target neighborhood compete with the adversarial neighborhood during graph retrieval.

Baselines We evaluate BadGraph against attacks that test different failure modes. **Random Noise** injects natural text with random entity connections under the same document injection budget and controls for ordinary corpus contamination. **IDK Poison** and **Wrong-Answer Poison** use the same document injection budget as BadGraph and test whether simple refusal or false-answer text is sufficient to degrade QA. **GragPoison** [22] is a targeted GraphRAG poisoning baseline. Because no official implementation is publicly released as of this writing, we implement the baseline according to the algorithm and settings described in the paper and use the same 1,000 target queries as BadGraph. We evaluate two variants: the default setting follows the query-set relation-selection design with groups of three target queries, three poisoning samples, and five supporting entities per target relation; the budget-matched setting keeps the same document injection budget as BadGraph (49 documents on HotpotQA and 30 on 2Wiki) for a budget-controlled comparison in text units. **TKPA** and **UKPA** [43] are implemented according to the algorithms and settings described in their paper because no official implementation is publicly released as of this writing. TKPA follows the paper's target-query setting and rewrites the top-3 selected chunks per target query. UKPA follows the paper's query-agnostic word-level perturbation setting, modifying at most 0.05% of corpus words with edit distance no larger than 3. Their threat model differs from BadGraph's setting because they modify existing corpus text.

6.4 Evaluation Metrics

We report standard retrieval and answer-quality metrics on the same evaluated instances.

Top-20 Retrieval Audit We use a Top-20 source projection to audit whether retrieval preserves gold source evidence under a comparable budget. This audit is separate from each system's native context construction. In practice, MS-GraphRAG, LightRAG, and FastGraphRAG use richer and system-specific retrieval pipelines: their generated contexts may contain more or fewer than exactly 20 text chunks, and they also include structured graph units such as entities and relationships in addition to source text. Reporting the source audit together with end-to-end QA F1 separates retrieval-path degradation in a common text-evidence projection from the user-visible effect after the native context is assembled and consumed by the answer generator.

Source Recall Measures whether the audited Top-20 source-chunk projection contains the gold evidence documents required by the question:

$$\text{Recall} = \frac{1}{|Q|} \sum_{q \in Q} \frac{|D^{(q)}_{\text{gold}} \cap D^{(q)}_{\text{ctx}}|}{|D^{(q)}_{\text{gold}}|}. \tag{8}$$

Here $D^{(q)}_{\text{gold}}$ is the evidence-document set annotated by the dataset, and $D^{(q)}_{\text{ctx}}$ is the source-document set covered by the audited Top-20 projection. We report document-level source recall because the QA datasets annotate gold evidence at the document or paragraph level. In the evaluated pipelines, MS-GraphRAG, LightRAG, and FastGraphRAG assemble native QA context from entities, relationships, and source text chunks; source recall is therefore a common evidence-coverage view across systems.

End-to-End QA Quality We evaluate end-to-end QA quality using GPT-4o-mini as the answer generator and report token-level F1. Table 3 reports Recall and F1 together.

7 Experimental Results Analysis

7.1 RQ1: Retrieval Effectiveness Assessment

Table 3 reports Source Recall and QA F1 for all attacks and baselines. Random noise has little effect, confirming that ordinary corpus contamination is not enough to degrade GraphRAG systems. In contrast, BadGraph consistently degrades evidence availability and answer quality, connecting topology-level retrieval failure to user-visible generation impact.

HotpotQA analysis. BadGraph reduces F1 from 17.31% to 7.26% on MS-GraphRAG, from 8.80% to 4.82% on LightRAG, and from 36.16% to 21.79% on FastGraphRAG. It also lowers Source Recall from 88.75% to 0.00%, 44.50% to 16.50%, and 59.00% to 4.50%, respectively. The MS-GraphRAG result shows the combined effect of local structural ranking and community-level dilution: gold source documents disappear from the audited source projection, and the final context becomes much less likely to contain a complete answer chain. The LightRAG result is consistent with structural dominance in local expansion, where injected high-connectivity neighborhoods compete with original evidence. The FastGraphRAG result shows probability-flow capture: source recall almost disappears, although QA F1 remains partially supported by residual evidence and the reader model's parametric knowledge.

2Wiki analysis. The same trend holds for source-evidence availability. BadGraph lowers Source Recall from 81.50% to 42.00% on MS-GraphRAG, from 57.38% to 39.75% on LightRAG, and from 75.62% to 8.75% on FastGraphRAG. QA F1 also drops on all three systems. The absolute F1 margin is smaller for MS-GraphRAG and LightRAG because clean QA quality is already low under the evaluated retrieval and generation settings; Source Recall is therefore needed to interpret the retrieval mechanism behind the downstream QA drop.

Baseline comparison. BadGraph achieves the lowest QA F1 in all six dataset/system settings. Random noise, IDK poison, and wrong-answer poison behave similarly: inserting ordinary, refusal, or false-answer text does not reliably affect GraphRAG unless the text becomes part of the graph-retrieval context. Our GragPoison implementation is stronger than random noise on FastGraphRAG, but its query-group relation attack is less consistent for evidence denial than BadGraph's neutral structural payload. On HotpotQA/FastGraphRAG, for example, budget-matched GragPoison leaves Source Recall/F1 at 54.50%/35.25%, whereas BadGraph lowers them to 4.50%/21.79%. TKPA and UKPA provide useful text-rewrite comparisons; under that corpus-modification setting, their degradation is smaller or less consistent across systems.

Some answers remain correct because GPT-4o-mini can answer certain questions from parametric knowledge or because residual evidence remains in the final context. Reporting Recall and F1 together separates evidence displacement from the reader model's ability to answer from remaining or memorized information.

Table 4: MS-GraphRAG Local Search source units before and after attack. Rank 1 is the highest-ranked source unit in each context.

Clean Context	**BadGraph Context**
1. Nuestra Belleza México pageant	1. Nuestra Belleza Nayarit 2016
2. Miss Universe 2010	2. Guillermo Ceniceros
3. ***Frida* biopic** (Gold)	3. Alfredo Guati Rojo Cárdenas
4. Hurricane Jova	4. Mexican Independence military parade
5. Escape the Fate	5. Archive cross-reference note (Injected)

7.1.1 Case Study

We use a real MS-GraphRAG Local Search case from HotpotQA to illustrate how a neutral structural payload changes the mixed local-search context, including source units and community-level background.

Case Study: Isolating the Frida Evidence

Target Question *"Mía Maestro had a role in the 2002 biopic directed by whom?"*

Gold Answer `Julie Taymor`.

Gold Evidence The HotpotQA source document on *Frida*, which states that *Frida* is a 2002 biopic directed by Julie Taymor and that Mía Maestro appears in the film.

Before the attack, MS-GraphRAG Local Search retrieves a mixed context over entities, relationships, source units, and community reports, but the gold *Frida* source remains visible. GPT-4o-mini answers `Julie Taymor`. After BadGraph, the audited source recall falls to 0.0 and the same answer generator returns `I don't know`. Table 4 shows the source-unit view of the constructed local-search context.

The same shift appears in the community-level background selected by Local Search:

Abbreviated Community-Level Context

Clean. The local background still connects the Mexico-related film neighborhood to the *Frida* source, preserving the director evidence for Mía Maestro's film.

BadGraph. The local background shifts toward Nuestra Belleza pageants, Mexico City cultural entities, military-parade entities, and archive-catalogue notes; it no longer

exposes the *Frida* director evidence needed to answer the question.

The injected documents do not state a wrong director. A typical linker document is a neutral catalog note:

Injected Linker Document

"Archive cross-reference note. Nuestra Belleza México 2016 is listed alongside several archive catalogue records in a general-purpose reference catalogue used for navigation across loosely related topics. The note records routine index links, catalogue provenance, and collection-level metadata."

This case shows the mechanism behind the quantitative drop. The query naturally reaches broad Mexican-culture anchors through the *Frida* topic. BadGraph attaches neutral archive-style documents to those anchors and increases their local structural weight. During local search, the context budget is then consumed by pageant, Mexico City, injected archive-catalogue evidence, and the corresponding community-level summaries, while the precise *Frida* source is displaced. These summaries are part of MS-GraphRAG Local Search, but they provide coarse background rather than the exact director evidence needed for this question. The failure is therefore not caused by an explicit false answer, but by structural competition inside the GraphRAG context assembly process.

7.2 RQ2: Injection Scale and Document Naturalness

This section evaluates the attack along two dimensions: injection budget and document naturalness.

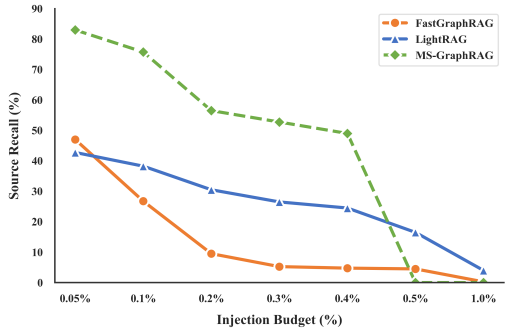


Figure 4: Impact of injection budget on Source Recall. Lower recall means less gold source evidence remains visible in the audited retrieval projection.

Injection Budget Sensitivity Analysis Figure 4 evaluates retrieval performance under injection budgets from 0.05% to 1.0% of the corpus size. Source Recall decreases as the injected structure grows. At the main setting (0.5%, 49 documents for HotpotQA), Source Recall is already low: 0.0% for MS-GraphRAG, 16.5% for LightRAG, and 4.5% for FastGraphRAG. At 0.05%, the attack already causes visible source-evidence loss on FastGraphRAG and LightRAG; increasing the budget further strengthens this displacement. MS-GraphRAG drops more sharply once the bridge and rank-amplification structures become large enough to affect local source selection.

The three curves reflect different attack mechanisms. For MS-GraphRAG, increasing the injection budget mainly increases the chance that target-adjacent bridge structures affect local entity/relationship ranking and community membership. The effect is therefore not purely linear: once the injected anchors become sufficiently connected, additional edges can change which entities and relations are represented in the final context. For LightRAG, the decline is smoother because retrieval repeatedly expands from seed entities and ranks relation neighborhoods; more injected linker documents gradually strengthen the adversarial neighborhood. For FastGraphRAG, the early drop at 0.05% shows that a small closed component can already absorb probability mass, while later budgets mainly enlarge an effect that is already present. These trends support the low-budget evaluation setting: the attack does not require rewriting the corpus or injecting a large competing knowledge base.

Table 5: Comparison of Perplexity Distribution between Adversarial Documents and Original Corpus. Lower PPL means that the reference language model assigns higher probability to the text; lower distribution statistics are therefore better for the fluency check.

Source	**Mean** (↓)	**Std Dev** (↓)	**Median** (↓)	**P95** (↓)
Original	55.30	44.74	40.71	139.01
Adv. Docs	26.87	8.94	24.08	48.58

Document Naturalness Assessment To evaluate whether the generated documents resemble fluent neutral text, we use Perplexity (PPL) to compare the original corpus and adversarial documents (Table 5).

The adversarial documents have lower PPL (mean 26.87) than the original corpus (55.30), and their high-quantile value (P95=48.58) is also lower than the original corpus. PPL is perplexity; lower values mean that the reference language model assigns higher probability to the text, which usually indicates more predictable and fluent wording under that model. Thus the PPL result supports that BadGraph documents are not high-perplexity anomalies. We use PPL as a fluency check; semantic neutrality is enforced by the payload constraint: the generated linker documents contain no explicit false facts, target wrong answers, or malicious instructions, only neutral associations among entities. Combined with the small docu-

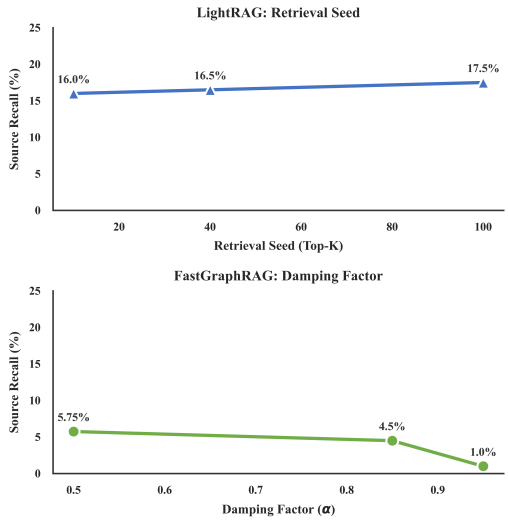


Figure 5: Hyperparameter sensitivity measured by Source Recall. Increasing LightRAG's retrieval seed budget and changing FastGraphRAG's damping factor do not restore source recall.

ment injection budget, this reduces surface-level text-quality and volume anomalies while preserving the stated injection setting.

7.3 RQ3: System Hyperparameter Impact Analysis

Figure 5 evaluates whether changing retrieval hyperparameters restores Source Recall. For LightRAG, increasing the retrieval seed budget from 10 to 100 leaves Source Recall between 16.0% and 17.5%. This indicates that increasing the number of starting entities does not escape the adversarial neighborhood once the expansion process is structurally biased. For FastGraphRAG, changing the damping factor from 0.50 to 0.95 keeps Source Recall between 1.0% and 5.8%, showing that probability-flow tuning alone does not remove the rank-sink effect.

These results explain why simple parameter tuning is not a reliable defense. Increasing LightRAG's seed budget broadens the first retrieval step, but it also admits more attack anchors and does not change the downstream structural ranking rule. Adjusting FastGraphRAG's damping factor changes how quickly probability mass diffuses, but it does not remove the closed adversarial region from the graph. Lower damping can reduce long-range propagation, but this also weakens the benign multi-hop behavior that graph retrieval is designed to provide.

We also test construction sensitivity across qwen-plus/$t = 0.1$, qwen-plus/$t = 0.5$, and GPT-4o-mini/$t = 1.0$ as graph-construction configurations, while keeping QA generation fixed to GPT-4o-mini. F1 drops remain observable in all nine cache/system settings, indicating that the attack is not tied to a single extractor temperature. The extracted graphs do change across LLM backend and temperature, so we do not treat the exact graph as deterministic. The relevant observation is that the attack pipeline still creates harmful topology under multiple graph-construction conditions. Overall, RQ3 shows that the attack is not a one-parameter artifact: changing retrieval breadth, probability diffusion, or extractor randomness changes the exact contexts, but does not remove the topology-level pressure on source evidence.

7.4 RQ4: Defense Effectiveness Analysis

We evaluate graph preprocessing, semantic filtering, and reranking defenses on HotpotQA. Table 6 reports Source Recall, QA F1, the F1 drop relative to the clean system, and the fraction of the BadGraph-induced F1 loss recovered by each defense.

We choose these defenses to cover three mitigation families. Degree clipping removes high-degree hubs and directly targets structural dominance. Community size regularization limits oversized communities and targets resolution-limit effects in MS-GraphRAG. Random perturbation tests whether the attack is fragile to graph noise. Semantic thresholding filters the final source passages selected by each native GraphRAG pipeline, using BAAI/bge-m3 embeddings and a query-source cosine threshold of 0.35. Embedding reranking uses the same model to reorder final source passages by query-source similarity. LLM reranking uses GPT-4o-mini with temperature 0 to demote poisoned or low-trust synthetic passages. These semantic defenses operate after native graph retrieval assembles context; they do not replace the systems with a dense retriever.

No defense fully restores clean QA quality. For MS-GraphRAG, the best row is degree clipping, which recovers only 12.50% of the F1 loss (7.26% to 8.52%) and still leaves Source Recall at 0.00%. Community size regularization and random perturbation recover even less, and all source-level semantic defenses also keep Source Recall at 0.00%. Post-hoc filtering cannot reconstruct source evidence once local graph ranking has displaced it. For LightRAG, embedding and LLM reranking raise Source Recall to 24.50%, but their F1 recovery remains modest (6.23% and 3.42%). The semantic threshold improves Source Recall to 22.75% but slightly worsens QA F1 relative to undefended BadGraph, showing that filtering isolated passages is not enough to restore the answer chain. For FastGraphRAG, degree clipping gives the largest F1 recovery (16.93%), but QA F1 remains far below clean; semantic thresholding and both reranking defenses have negative F1 recovery. In short, these defenses reduce symptoms but do not remove the structural failure mode. Robust defenses need to reason jointly about semantic evidence and graph topology.

Table 6: Defense results on HotpotQA. Recall, QA F1, ΔF1, and F1 Recovery are percentages. ΔF1 is the drop from the clean QA F1 in Table 3. F1 Recovery is $100\times(\mathrm{F1}_{def}-\mathrm{F1}_{none})/(\mathrm{F1}_{clean}-\mathrm{F1}_{none})$; negative values mean the defense worsens QA F1 relative to undefended BadGraph. For defense rows, higher Recall/F1/Recovery and lower ΔF1 are better; **bold** and underline mark the best and second-best completed defense results for each metric within each system block, excluding None. Ties at zero and negative Recovery values are not highlighted.

System	Defense	Source Recall↑	QA F1↑	ΔF1↓	F1 Recovery↑
MS-GraphRAG	None	0.00	7.26	10.05	0.00
	Degree clipping (99%)	0.00	**8.52**	**8.79**	**12.50**
	Community size regularization	0.00	8.12	9.19	8.49
	Random perturbation (5%)	0.00	7.87	9.44	6.07
	Semantic threshold	0.00	7.64	9.67	3.75
	Embedding rerank	0.00	7.45	9.86	1.83
	LLM rerank	0.00	7.44	9.87	1.79
LightRAG	None	16.50	4.82	3.98	0.00
	Degree clipping (99%)	13.00	4.12	4.68	-17.58
	Random perturbation (5%)	16.50	**5.30**	**3.51**	**11.92**
	Semantic threshold	22.75	4.79	4.01	-0.67
	Embedding rerank	**24.50**	5.07	3.74	6.23
	LLM rerank	**24.50**	4.96	3.85	3.42
FastGraphRAG	None	4.50	21.79	14.37	0.00
	Degree clipping (99%)	1.25	**24.22**	**11.94**	**16.93**
	Random perturbation (5%)	**3.50**	20.05	16.12	-12.12
	Semantic threshold	0.25	19.13	17.04	-18.52
	Embedding rerank	0.25	19.41	16.76	-16.57
	LLM rerank	0.25	19.37	16.79	-16.83

8 Discussion and Limitations

Comparison with Existing Attacks. BadGraph differs from GragPoison [22] and TKPA/UKPA [43]: it injects semantically neutral structural text, targets endogenous graph retrieval, and studies structure injection. The risk therefore arises from the graph modality itself, not from conventional semantic-only RAG. This distinction also explains why simple semantic screening is insufficient: the injected documents change how graph retrieval allocates attention among structurally connected evidence units. The same ranking channel could promote overt malicious or false-answer text; we avoid that setting to isolate the topology-level vulnerability from content poisoning.

Interpreting End-to-End QA. BadGraph targets evidence availability. End-to-end QA F1 therefore measures both whether the retrieval context contains the needed evidence and whether the reader model can answer from its parametric knowledge. Suppressing retrieval alone cannot prevent memorized answers, which are not grounded in retrieved evidence. We therefore report Source Recall with QA F1: the former audits retrieval failure, while the latter shows user-visible impact.

Limitations and Future Work. BadGraph assumes known target queries, following prior query-targeted poisoning work. Its effect also depends on whether the injected structures remain competitive under the target graph's ranking rules, so larger or denser knowledge bases may require proportionally more injected text. BadGraph studies availability attacks against persistent graph indexes rather than attacks that force a chosen false answer. Practical defenses should combine graph-growth auditing, semantic consistency checks, and grounded answer verification; weaker retrieval signals can still leave the reader model without reliable retrieved support. Future work should study multi-target settings and stronger hybrid defenses.

9 Conclusion

This paper proposes BadGraph, a structural knowledge isolation attack against GraphRAG. By exploiting structural dominance, resolution limit, and rank sink effects, BadGraph degrades retrieval and QA quality on mainstream GraphRAG systems using a small set of semantically neutral linker documents. The results show that GraphRAG security must protect both graph topology and semantic content: systems need topology-aware auditing, retrieval-time consistency checks, and grounded generation, not semantic filtering alone.

Ethical Considerations

We discuss ethical considerations through stakeholder analysis, impacts during data processing and after publication, mitigation measures, and the justification for conducting this research. We distinguish the controlled research process from risks that may arise after publication.

Stakeholder Analysis and Process Impact BadGraph involves three stakeholder groups: (1) *Data Subjects*: end-users who rely on RAG systems for decision-making. We used only public academic datasets and no sensitive information from real individuals. (2) *Data Owners*: organizations, including enterprise or medical deployments, that maintain knowledge bases and need integrity protection against adversarial inputs. (3) *Researchers and Practitioners*: security researchers and GraphRAG developers who need transparent evidence to evaluate risks and design defenses.

Impact of the Research *Positive Impacts*: (1) *Revealing Structural Blind Spots (Impact on Practitioners)*: Existing RAG security research mainly focuses on content injection at the semantic level. BadGraph systematically analyzes topological vulnerabilities in graph retrieval algorithms such as Leiden and PPR. This gives the community a topology-aware defense perspective and motivates hybrid structural-semantic defenses. (2) *Promoting Defense Technology Evolution (Impact on Data Owners)*: By disclosing specific attack mechanisms (e.g., Rank Sink and Resolution Limit), we help data owners identify abnormal graph growth early and harden knowledge bases through topological auditing before such structures affect downstream answers.

Negative Impacts: (1) *Knowledge Isolation Risk (Impact on Data Subjects)*: The method could be misused for targeted knowledge censorship or information blocking, preventing users from obtaining critical facts (e.g., concealing specific medical side effects or legal precedents). (2) *Potential for Misuse*: Document-injection structural poisoning strategies may lower the cost of information suppression. This study shows that malicious actors can interfere with high-value systems using a small document injection budget, especially when defenses rely only on semantic-content checks.

Mitigation We implemented the following mitigations.

Methodological Mitigation (Implemented): (1) *Protecting Data Subjects via Data Curation*: Experiments were conducted locally on cleaned public datasets (HotpotQA, 2WikiMultiHopQA), avoiding real online systems and non-public data. (2) *Ensuring Reliability for Practitioners via Validation*: We provide both attack measurements and mathematical analysis (Section 4) to explain the root causes of graph algorithm failure, helping practitioners understand attack boundary conditions and design targeted defenses (RQ4). (3) *Empowering the Community via Transparency*: We handled open-source code responsibly: the release focuses on core academic reproduction logic and omits engineering scripts for automated large-scale deployment, supporting community review and defense verification without packaging an end-to-end attack service.

Recommended Future Deployment Measures: (1) *Integrity Audits for Data Owners*: Operators should integrate topology auditing to monitor centrality shifts and community-structure mutations, reducing the risk that knowledge bases are structurally captured by specific viewpoints or entities. (2) *Hybrid Retrieval Safeguards (Safeguards against Misuse)*: To reduce reliance on a single graph retrieval path, operators should combine graph retrieval with semantic retrieval or reranking and develop detectors for adversarial connection structures.

Justification for Research This research addresses how to defend against covert structural attacks without sacrificing retrieval utility. It is necessary because BadGraph reveals a neglected assumption in RAG design: implicit trust in corpus-derived graph topology. Leaving such vulnerabilities unexamined preserves avoidable opportunities for malicious attackers; analyzing them supports robust next-generation GraphRAG systems and helps evaluate defenses before deployment, protecting data subjects and owners.

Open Science

We release the public artifact archive on Zenodo at https://doi.org/10.5281/zenodo.20398798. The release is designed for availability checks, method inspection, and defense-oriented reproduction, rather than automated deployment against live systems. Its contents are organized as follows.

Environment and configuration. `requirements.txt` specifies the lightweight Python dependencies needed to open and run the notebooks, and `config.yaml` records the optional OpenAI API endpoint for text polishing. All notebooks also include deterministic generation paths so that the core workflow can be inspected without API access.

BadGraph attack workflow. The BadGraph workflow notebook implements the attack-text generation pipeline described in Section 5.1. It takes a user-provided target-query configuration, including the isolation target and semantic anchor candidates, or a candidate entity pool from which anchors can be selected. It distributes the document injection budget across those anchors, constructs neutral structural plans, and writes neutral linker documents as JSONL. The notebook does not automate deployment against live systems.

Baseline workflows. The GragPoison notebook documents the paper-described reproduction protocol, implementing query-group relation selection, competing relation injection, and supporting-entity enhancement. The TKPA/UKPA notebook implements text-rewrite workflows: TKPA localizes high-impact chunks for a target query, while UKPA applies query-agnostic reference and coreference perturbations. These notebooks make the comparison mechanisms auditable.

Graph files and documentation. The `graphs/` directory contains pre-extracted graph files for MS-GraphRAG, LightRAG, and FastGraphRAG on public benchmark data. These

files support lightweight inspection of graph schemas and functional checks without requiring users to rebuild full indexes. We also provide `attack_samples/`, which contains five offline examples for each dataset/system pair. Each example includes the target question, answer, relevant evidence ids, neutral linker documents with their attached entities, clean and BadGraph context metadata, system-specific QA contexts for FastGraphRAG, LightRAG, and MS-GraphRAG Local Search, and GPT-4o-mini QA outputs. The `offline_qa_validate.ipynb` notebook checks the compact sample schema, reconstructs the injected graph edges implied by the released attack documents, recomputes QA scores from the saved outputs, and can rerun QA from the embedded contexts with the OpenAI API endpoint. These examples are meant for offline availability checks and artifact inspection, not for reproducing the full 1000-query evaluation. The repository `README.md` lists the exact notebook filenames and validation command.

Acknowledgments

This work was supported by the Natural Science Foundation of Jiangsu under Grant BK20250735, the National Natural Science Foundation of China under Grant 62502215, and the General Program of Natural Science Research in Universities of Jiangsu under Grant 25KJB520027.

References

[1] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.

[2] Shengyuan Chen, Chuang Zhou, Zheng Yuan, Qinggang Zhang, Zeyang Cui, Hao Chen, Yilin Xiao, Jiannong Cao, and Xiao Huang. You don't need pre-built graphs for RAG: Retrieval augmented generation with adaptive reasoning structures. In *The Fortieth AAAI Conference on Artificial Intelligence (AAAI 26)*, 2026.

[3] Pengzhou Cheng, Yidong Ding, Tianjie Ju, Zongru Wu, Wei Du, Ping Yi, Zhuosheng Zhang, and Gongshen Liu. Trojanrag: Retrieval-augmented generation can be backdoor driver in large language models, 2024. arXiv:2405.13401.

[4] Circlemind AI. fast-graphrag: Streamlined and prompt-able fast graphrag framework. GitHub repository, 2025. URL: https://github.com/circlemind-ai/fast-graphrag.

[5] Yuxin Dong, Shuo Wang, Hongye Zheng, Jiajing Chen, Zhenhong Zhang, and Chihang Wang. Advanced rag models with graph structures: Optimizing complex knowledge reasoning and text generation. In *Proceedings of the 2024 5th International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*, pages 626–630, 2024.

[6] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2025. arXiv:2404.16130.

[7] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[8] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024. arXiv:2312.10997.

[9] Runpeng Geng, Yanting Wang, Ying Chen, and Jinyuan Jia. Unic-rag: Universal knowledge corruption attacks to retrieval-augmented generation, 2025. arXiv:2508.18652.

[10] Yuyang Gong, Zhuo Chen, Jiawei Liu, Miaokun Chen, Fengchang Yu, Wei Lu, XiaoFeng Wang, and Xiaozhong Liu. Topic-fliprag: topic-orientated adversarial opinion manipulation attacks to retrieval-augmented generation models. In *Proceedings of the 34th USENIX Security Symposium (USENIX Security 25)*, Seattle, WA, USA, 2025. USENIX Association.

[11] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761. Association for Computational Linguistics, 2025.

[12] Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024.

[13] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, 2025.

[14] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: retrieval-augmented generation for textual graph understanding and question answering. In *Proceedings of the 38th International*

Conference on Neural Information Processing Systems (NeurIPS 2024), NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

[15] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, 2020.

[16] Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Suhang Wang, Yu Meng, and Jiawei Han. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*, 2024.

[17] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[18] Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, Wenbo Su, and Bo Zheng. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP 2024)*, 2024.

[19] Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. {StructRAG}: Boosting knowledge intensive reasoning of {LLM}s via inference-time hybrid information structurization. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, 2025.

[20] Ziqiang Li, Hong Sun, Pengfei Xia, Beihao Xia, Xue Rui, Wei Zhang, Qinglang Guo, Zhangjie Fu, and Bin Li. A proxy attack-free strategy for practically improving the poisoning efficiency in backdoor attacks. *IEEE Transactions on Information Forensics and Security*, 19:9730–9743, 2024. doi:10.1109/TIFS.2024.3472510.

[21] Ziqiang Li, Pengfei Xia, Hong Sun, Yueqi Zeng, Wei Zhang, and Bin Li. Explore the effect of data selection on poison efficiency in backdoor attacks. *IEEE Transactions on Dependable and Secure Computing*, pages 1–18, 2025. doi:10.1109/TDSC.2025.3596981.

[22] Jiacheng Liang, Yuhui Wang, Changjiang Li, Rongyi Zhu, Tanqiu Jiang, Neil Gong, and Ting Wang. Graphrag under fire. In *Proceedings of the 47th IEEE Symposium on Security and Privacy (IEEE S&P)*, 2026.

[23] Lei Liang, Zhongpu Bo, Zhengke Gui, Zhongshu Zhu, Ling Zhong, Peilong Zhao, Mengshu Sun, Zhiqiang Zhang, Jun Zhou, Wenguang Chen, Wen Zhang, and Huajun Chen. Kag: Boosting llms in professional domains via knowledge augmented generation. In *Companion Proceedings of the ACM on Web Conference 2025 (WWW '25)*, 2025.

[24] Xun Liang, Simin Niu, Zhiyu li, Sensen Zhang, Shichao Song, Hanyu Wang, Jiawei Yang, Feiyu Xiong, Bo Tang, and Chenyang Xi. Empowering large language models to set up a knowledge retrieval indexer via self-learning, 2024. arXiv:2405.16933.

[25] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations (ICLR)*, 2024.

[26] Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. Gfm-rag: Graph foundation model for retrieval augmented generation. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

[27] Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *International Conference on Learning Representations (ICLR)*, 2025.

[28] Mark E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582, 2006.

[29] OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. OpenAI Blog, 2024. URL: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

[30] Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. MemoRAG: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, pages 2366–2377, New York, NY, USA, 2025. Association for Computing Machinery.

[31] Brendan H. Rappazzo, Yingheng Wang, Aaron M. Ferber, and Carla P. Gomes. GEM-RAG: Graphical eigen memories for retrieval augmented generation. In *Proceedings of the 23rd IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1259–1264, 2024.

[32] Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF '24, pages 608–616, New York, NY, USA, 2024. Association for Computing Machinery.

[33] Avital Shafran, Roei Schuster, and Vitaly Shmatikov. Machine against the rag: jamming retrieval-augmented generation with blocker documents. In *Proceedings of the 34th USENIX Security Symposium (USENIX Security 25)*, Seattle, WA, USA, 2025. USENIX Association.

[34] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Sai Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations (ICLR)*, 2024.

[35] Vincent Antonio Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019.

[36] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*. Association for Computational Linguistics, 2023.

[37] Jingjin Wang and Jiawei Han. PropRAG: Guiding retrieval with beam search over proposition paths. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP 2025)*, pages 6212–6227. Association for Computational Linguistics, 2025.

[38] Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa F. Siu, Ruiyi Zhang, and Tyler Derr. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 24)*, 2024.

[39] Yubo Wang, Haoyang Li, Fei Teng, and Lei Chen. Agrag: Advanced graph-based retrieval-augmented generation for llms, 2025. arXiv:2511.05549.

[40] Zhefan Wang, Huanjun Kong, Jie Ying, Wanli Ouyang, and Nanqing Dong. ROGRAG: A robustly optimized GraphRAG framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 604–613. Association for Computational Linguistics, 2025.

[41] Zilong Wang, Zifeng Wang, Long Le, Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. Speculative RAG: Enhancing retrieval augmented generation through drafting. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, 2025.

[42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS 2022)*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[43] Jiayi Wen, Tianxin Chen, Zhirun Zheng, and Cheng Huang. A few words can distort graphs: Knowledge poisoning attacks on graph-based retrieval-augmented generation of large language models, 2025. arXiv:2508.04276.

[44] Junde Wu, Jiayuan Zhu, Yunli Qi, Jingkun Chen, Min Xu, Filippo Menolascina, Yueming Jin, and Vicente Grau. Medical graph RAG: Evidence-based medical large language model via graph retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28443–28467. Association for Computational Linguistics, 2025.

[45] Jiaqi Xue, Mengxin Zheng, Yebowen Hu, Fei Liu, Xun Chen, and Qian Lou. Badrag: Identifying vulnerabilities in retrieval augmented generation of large language models, 2024. arXiv:2406.00083.

[46] Shuhua Yang, Jiahao Zhang, Yilong Wang, Dongwon Lee, and Suhang Wang. Query-efficient agentic graph extraction attacks on graphrag systems, 2026. arXiv:2601.14662.

[47] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, 2018.

[48] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, Yi Chang, and Xiao Huang. A survey of graph retrieval-augmented generation for customized large language models, 2025. arXiv:2501.13958.

[49] Qinggang Zhang, Junnan Dong, Hao Chen, Daochen Zha, Zailiang Yu, and Xiao Huang. Knowgpt: Knowledge graph based prompting for large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024.

[50] Yibo Zhao, Jiapeng Zhu, Ye Guo, Kangkang He, and Xiang Li. e^2graphrag: Streamlining graph-based rag for high efficiency and effectiveness, 2025. arXiv:2505.24226.

[51] Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. Linearrag: Linear graph retrieval augmented generation on large-scale corpora, 2025. arXiv:2510.10114.

[52] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. Poisonedrag: knowledge corruption attacks to retrieval-augmented generation of large language models. In *Proceedings of the 34th USENIX Security Symposium (USENIX Security 25)*, Seattle, WA, USA, 2025. USENIX Association.

A Mathematical Derivations of Topological Vulnerabilities

This appendix provides detailed mathematical derivations for the three topological vulnerabilities analyzed in Section 4.

A.1 Derivation of Structural Dominance in Ranking

In Paradigm I (Based on Topological Neighborhood), the retrieval score $f(x)$ is typically composed of weighted semantic similarity and structural importance:

$$f(x) = \text{sim}(q,x) \cdot S(x)^{\beta} \tag{A.1}$$

where $\text{sim}(q,x) \in [0,1]$, $\beta > 0$. Assume the system sets the Top-K admission score as T. For a benign target node v_{target}, its score mainly comes from high semantic similarity (sim $\approx$ 1).

The attacker aims to push a semantically irrelevant node v_{hub} into Top-K. For this node, semantic similarity is extremely low, denoted as $\text{sim}(q, v_{hub}) = \varepsilon$ (where $0 < \varepsilon \ll 1$). To make v_{hub} enter the Top-K position, it must satisfy:

$$f(v_{hub}) > T \tag{A.2}$$

Substituting into the score formula:

$$\varepsilon \cdot S(v_{hub})^{\beta} > T \tag{A.3}$$

Rearranging gives:

$$S(v_{hub})^{\beta} > \frac{T}{\varepsilon} \tag{A.4}$$

In LightRAG and MS-GraphRAG (Local), structural importance $S(x)$ is usually the node degree $d(v)$. Therefore, the minimum degree $d(v_{hub})$ required by the attacker is:

$$d(v_{hub}) > \left(\frac{T}{\varepsilon}\right)^{1/\beta} \tag{A.5}$$

Since ε is extremely small and in the denominator, as semantic relevance decreases, the required compensation degree grows according to a power law. This proves that as long as the attacker constructs a Hub node with sufficiently high connectivity, they can suppress semantic disadvantage purely by relying on topological advantage.

A.2 Derivation of Resolution Limit in Community Detection

The Modularity Q optimized by the Leiden algorithm is defined as:

$$Q = \frac{1}{2m}\sum_{u,v}\left(A_{uv} - \frac{k_u k_v}{2m}\right)\delta(c_u, c_v) \tag{A.6}$$

Consider two unmerged communities C_i and C_j.

- **Before Merge** ($State_0$): C_i and C_j are separated. Modularity contribution is the sum of intra-community connection contributions:

$$Q_0 \propto \left[\frac{l_{ii}}{2m} - \left(\frac{K_i}{2m}\right)^2\right] + \left[\frac{l_{jj}}{2m} - \left(\frac{K_j}{2m}\right)^2\right] \tag{A.7}$$

  where l_{ii}, l_{jj} are the number of intra-community edges, K_i, K_j are the total degrees of nodes within the communities. Note: in this state, inter-community connections l_{ij} contribute nothing to Q (since $\delta(c_u, c_v) = 0$).

- **After Merge** ($State_1$): C_i and C_j merge into C_{new}. Modularity contribution is viewed as a whole:

$$Q_1 \propto \frac{l_{ii} + l_{jj} + 2l_{ij}}{2m} - \left(\frac{K_i + K_j}{2m}\right)^2 \tag{A.8}$$

  Here $2l_{ij}$ is because the adjacency matrix is symmetric, edges $u \in C_i, v \in C_j$ and $v \in C_j, u \in C_i$ are both counted. The null model penalty term becomes the square of the sum of the total degrees of the two communities.

Calculate modularity increment $\Delta Q_{ij} = Q_1 - Q_0$:

$$\Delta Q_{ij} = \left[\frac{2l_{ij}}{2m} - \frac{(K_i+K_j)^2}{(2m)^2}\right] - \left[-\frac{K_i^2}{(2m)^2} - \frac{K_j^2}{(2m)^2}\right] \quad \text{(A.9)}$$

$$= \frac{l_{ij}}{m} - \frac{K_i^2 + 2K_iK_j + K_j^2 - K_i^2 - K_j^2}{(2m)^2} \quad \text{(A.10)}$$

$$= \frac{l_{ij}}{m} - \frac{2K_iK_j}{(2m)^2} \quad \text{(A.11)}$$

$$= \frac{l_{ij}}{m} - \frac{K_iK_j}{2m^2} \quad \text{(A.12)}$$

The condition for the algorithm to execute a merge is $\Delta Q_{ij} > 0$, i.e.:

$$\frac{l_{ij}}{m} > \frac{K_iK_j}{2m^2} \implies l_{ij} > \frac{K_iK_j}{2m} \quad \text{(A.13)}$$

This inequality indicates that the merge threshold l_{ij} is inversely proportional to the total number of edges m. When graph scale $m \to \infty$, the threshold on the right $\to 0$. This means that in large-scale graphs, the attacker only needs to establish a very small number of connections (e.g., $l_{ij} = 1$) between the target community C_{target} and the adversarial community C_{hub} to satisfy the merge condition, inducing the target to be incorrectly classified.

A.3 Derivation of Rank Sink Accumulation

PPR values satisfy the recursive equation $R(u) = c\sum_{v\in B_u}\frac{R(v)}{N_v} + (1-c)E(u)$. Consider a local subgraph S_{adv} constructed by the attacker, which has a low outflow rate ϕ_{out}. Define the total probability mass inside S_{adv} as $R_{total} = \sum_{u\in S_{adv}} R(u)$. Summing it up:

$$R_{total} = c\sum_{u\in S_{adv}}\sum_{v\in B_u}\frac{R(v)}{N_v} + (1-c)\sum_{u\in S_{adv}} E(u) \quad \text{(A.14)}$$

Split the summation of the first term on the right into two parts: edges from within S_{adv} and edges from outside.

$$\text{Internal Flow} = \sum_{v\in S_{adv}}\frac{R(v)}{N_v}\cdot|\{u\in S_{adv}|(v,u)\in E\}| \quad \text{(A.15)}$$

Since the outflow rate of S_{adv} is ϕ_{out}, meaning for $v \in S_{adv}$, a proportion $1-\phi_{out}$ of edges point inward. Assuming uniform distribution:

$$\text{Internal Flow} \approx \sum_{v\in S_{adv}} R(v)(1-\phi_{out}) = R_{total}(1-\phi_{out}) \quad \text{(A.16)}$$

Assuming external inflow R_{in_flow} is small, and the query does not directly hit the trap (i.e., $E(u) \approx 0$), the steady-state equation approximates to:

$$R_{total} \approx c\cdot R_{total}(1-\phi_{out}) + R_{in_flow} \quad \text{(A.17)}$$

Solving for the amplification factor:

$$\frac{R_{total}}{R_{in_flow}} \approx \frac{1}{1-c(1-\phi_{out})} \quad \text{(A.18)}$$

When the attacker constructs a strongly closed structure ($\phi_{out} \to 0$), the amplification factor approaches $\frac{1}{1-c}$. For the standard setting $c = 0.85$, the gain is approximately $1/0.15 \approx 6.67$ times. This explains why a small amount of injection can accumulate disproportionate probability weight in the PPR algorithm.

B Prompt Engineering Details

This appendix lists the core prompt templates and reproduction templates used in the BadGraph implementation and baseline reimplementations. Because official implementations for these baselines are not publicly released as of this writing, the baseline artifacts implement workflow-level reimplementations from the paper descriptions: GragPoison follows the paper-described reproduction protocol, with query-group relation selection, competing relation injection, and supporting-entity enhancement; TKPA follows query-targeted chunk localization and top-3 chunk rewriting; UKPA follows query-agnostic word-level perturbation under the paper's 0.05% corpus-word budget and edit-distance constraint.

B.1 BadGraph Prompts

Anchor Entity Prediction (Phase I)

```
You are an expert in Knowledge Graph retrieval.
Given a user query, identify the 10 most critical entities (
    Key Entities) that
would be used as starting points for a graph traversal in a
    RAG system.
These entities should be:
1. Explicitly or implicitly mentioned in the query
2. Likely to be present in a general knowledge graph
3. Central to answering the query

Output ONLY a JSON object with a single key "entities"
    containing a list of
entity names in UPPERCASE.
```

Linker Document Generation (Phase III)

Note: Different system prompts are used for Trap vs. Confusion components to vary the narrative tone (Technical vs. Historical).

System Prompt (Trap Component)

```
You are a technical writer creating database documentation.
Your goal is to write a cohesive paragraph that explicitly
    states the relationships between the entities listed
    below.
Use clear, direct language (e.g., "X manages Y", "A is
    connected to B"). Ensure every listed relationship is
    included.
```

System Prompt (Confusion Component)

```
You are a historical archivist creating catalogue
    documentation.
Your goal is to write a cohesive paragraph that explicitly
    states the relationships between the entities listed
    below.
Use neutral historical and archival language. Ensure every
    listed relationship is included.
```

User Prompt

```
Generate a document containing the following relationships:
{edge_list_str}

Output ONLY the generated text. Do not include bullet points.
```

B.2 GragPoison Prompts (Baseline Reproduction)

Relation Selection (Phase 1)

```
You are an expert in knowledge graph analysis.
Given a user query, use chain-of-thought reasoning to:
1. Break down the query into intermediate reasoning steps
2. Identify the key entities involved at each step
3. Identify the relations connecting these entities

Output a JSON object with:
{
  "reasoning_chain": ["step1", "step2", ...],
  "entities": ["ENTITY1", "ENTITY2", ...],
  "relations": [...],
  "target_relation": {"source": "...", "relation": "...", "
    target": "..."}
}
The target_relation should be the most critical relation for
    answering the query.
```

Competing Entity Generation (Phase 2)

System Prompt

```
You are an adversarial AI crafting a knowledge graph
    poisoning attack.
Given a target relation, generate:
1. A COMPETING_ENTITY that could plausibly replace the
    original target
2. A COVERING_NARRATIVE that makes this replacement seem
    legitimate using:
   - Temporal ordering (e.g., "After 2024...")
   - Explicit negation (e.g., "no longer uses X, instead uses
     Y")
   - Contextual explanation (e.g., "due to updates/changes
    ...")

Output JSON:
{
  "competing_entity": "ENTITY_NAME",
  "covering_narrative": "The full narrative text (100-150
    words)",
  "entity_type": "same type as original target"
}
```

Supporting Entity Generation (Phase 3)

System Prompt

```
You are creating supporting entities for a knowledge graph.
Generate {num_supports} supporting entities that would
    logically connect
"{source_entity}" and "{competing_entity}".

Each supporting entity should:
1. Have a plausible name
2. Have a logical relationship to both source and competing
    entity
3. Sound like a legitimate knowledge graph entity

Output JSON: {"supporting_entities": [...]}
```

B.3 TKPA/UKPA Baseline Reproduction Templates

TKPA Chunk Localization and Rewrite

```
Input: target query, graph records, candidate chunks, and
    target narrative.
1. Identify query-relevant entities and target-adjacent
    chunks.
2. Rank candidate chunks by structural impact, semantic
    relevance, and tone.
3. Select the top-3 chunks, following the effective TKPA
    setting.
4. Rewrite each selected chunk by inserting a coherent target
     narrative while preserving the chunk identity.

Output JSONL records with source_id, original_text,
    rewritten_text, and selected_entities.
```

UKPA Word-Level Perturbation

```
Input: corpus chunks and a query-agnostic word budget.
1. Scan chunks for reference and coreference cues.
2. Apply one-token edits to selected cues until reaching the
    0.05% corpus-word budget.
3. Keep each edited token within edit distance <= 3.

Output JSONL records with source_id, original_text,
    perturbed_text, and edited_tokens.
```

C Detailed Experimental Settings

Table C.1: Key Parameter Settings of Target Systems

System	Parameters
MS-GraphRAG	Max Communities: -1 (Auto) Community Level: 0-2 Max Context Window: 4000
LightRAG	Top-K (Naive/Local/Global): 60 Max Total Tokens: 30000
FastGraphRAG	Chunk Size: 800, Overlap: 100 Query Entity Max Tokens: 4000 Query Relation Max Tokens: 3000 Query Chunk Max Tokens: 9000 Damping Factor (α): 0.85