让窃取变得毫无用处：GraphRAG 系统中基于掺假的专有知识图谱保护

王伟杰[1,2,*], 吕佩卓[3,*,†], 王燕[1,5,†], 戴如杰[1], 徐国坤[1], 吕秋剑[1], 刘航成[3], 黄伟庆[1], 董伟[3], 张家恒[2]
[1]中国科学院信息工程研究所 [2]新加坡国立大学 [3]南洋理工大学 [4]北京工业大学 邮箱：wangweijie@iie.ac.cn, lvpeizhuo@gmail.com, {wangyan,dairujie2024,xuguokun,lvqiujian}@iie.ac.cn, hangcheng.liu@ntu.edu.sg, huangweiqing@iie.ac.cn, wei dong@ntu.edu.sg, jhzhang@nus.edu.sg

arXiv:2601.00274v1 [cs.CR] 1 Jan 2026

摘要—图检索增强生成（GraphRAG）已成为在知识密集型应用中利用专有知识图谱（知识图谱）增强大语言模型（大语言模型）的关键技术。由于这些知识图谱通常代表组织极具价值的知识产权，因此面临着被窃取并用于私人用途的重大风险。在此情境下，攻击者在隔离环境中行动。这种私人用途威胁使数字水印等被动防御措施失效，因为它们需要访问输出才能进行检测。与此同时，GraphRAG 对低延迟的要求，使得开销高昂的强加密不切实际。为应对这些挑战，我们提出了 AURA，这是一种基于数据掺假的新型框架，旨在使任何被盗知识图谱都无法被对手使用。我们的框架会预先向知识图谱中注入看似合理却不真实的掺杂物。对攻击者而言，这些掺杂物会损害检索上下文，导致回答出现事实错误。相反，对于授权用户，密钥可通过加密元数据标签在内容传递给大语言模型之前，高效过滤掉所有掺杂物，从而确保查询结果完全准确。评估结果表明，该方法有效：AURA 将未经授权的系统的准确率降至仅 5.3%，同时使授权用户的保真度保持在 100%，且开销可以忽略不计。此外，AURA 能够抵御各种清理尝试，保留 80.2% 的掺杂物。

一、引言

图检索增强生成（GraphRAG）已被广泛用于缓解大语言模型（LLM）中的幻觉，并增强其在特定领域、知识密集型及隐私敏感型任务中的能力。例如，辉瑞 [1] 利用它加速药物发现，Siemens 则借此推动其智能制造战略 [2]。支撑这些先进系统的专有知识图谱（知识图谱）存储着特定领域的知识，其中包括大量知识产权和敏感信息。尽管这些商业知识图谱的构建成本并未公开，但所需投资十分可观。专家主导项目的规模可见一斑，例如 Cyc [3], ，其估计耗资 1.2 亿美元，折合每条事实陈述约 5.71 美元 [4]。

专有知识图谱的高价值使其成为知识产权盗窃的主要目标。攻击者可能通过外部网络入侵窃取知识图谱，也可能利用恶意内部人员实施窃取。例如，一名明星工程师窃取了超过 14,000 份与 Waymo 激光雷达系统相关的专有文件，随后加入了一家竞争对手 [5]；2020年年，欧洲药品管理局遭到黑客攻击，攻击者借此获取了辉瑞/BioNTech 的机密疫苗数据 [6]。攻击者成功窃取知识图谱后，可以将其部署在私有 GraphRAG 系统中，复制其强大功能，避免高昂的投资成本。主要监管框架已认识到这一威胁的严重性：欧盟人工智能法案 [7]和美国国家标准与技术研究院人工智能风险管理框架 [8] 均强调了强有力的数据安全与韧性的必要性，凸显了开发有效知识图谱保护措施的重要性。

由于面临私有使用的威胁，且 GraphRAG 对性能有严格要求，保护 GraphRAG 系统中的知识图谱尤其具有挑战性。在私有部署场景中，攻击者会在隔离环境中运行被盗知识图谱，因此数字水印等常见防御措施 [9]无法奏效。数字水印需要访问系统输出才能追踪泄露。然而，在私有环境下，所有者无法获取输出，因此不可能检测滥用行为或追究攻击者的责任。与此同时，交互式 GraphRAG 对低延迟的要求，使得强加密方案（例如同态加密）不切实际。对文本和嵌入进行完全加密，意味着每次查询都需要解密图中的大部分内容。这一过程会带来难以承受的计算开销和延迟，因而不适用于现实场景 [10]。

为应对这些挑战，我们提出 AURA（通过污染主动降低效用），这是一种新颖的框架，可使被盗知识图谱对对手毫无用处，同时保持

* 共同第一作者。 † 共同通讯作者。

第一作者在新加坡国立大学担任访问学者期间开展了这项工作，谨此致谢该校的支持。

GraphRAG 系统的性能开销极小。AURA 通过数据掺假向知识图谱中注入看似可信但实际虚假的信息。为确保效率和鲁棒性，该流程首先识别一组数量最少、影响最大的关键节点。随后，采用混合生成策略，生成在语义和结构层面都看似可信的复杂掺杂物。由于我们的目标是使知识图谱在未经授权的 GraphRAG 系统中无法使用，因此我们只选取生成的掺杂物中对大语言模型输出影响最大的部分进行注入。因此，当攻击者使用被盗副本时，这些掺杂物会作为上下文被检索出来，削弱大语言模型的推理能力，导致其给出事实错误的回答。相反，授权系统会使用只有所有者知晓的密钥，通过加密元数据标签识别并过滤所有掺杂物，再将内容传递给大语言模型，从而确保查询结果完全准确。这种设计确保只有持有密钥的用户才能区分注入的掺杂物与真实数据，从根本上使知识图谱的效用与密钥的保密性紧密关联，并防止其被私自使用。

我们在四个基准数据集（MetaQA、WebQSP、FB15k-237 和 HotpotQA）以及四种不同的大语言模型（GPT-4o、Gemini-2.5-flash、Qwen-2.5-7B 和 Llama2-7B）上评估了 AURA。首先，AURA 能有效降低被盗知识图谱对未授权用户的实用性，使未经授权的系统性能降至仅 4.4% 至 5.3% 的准确率。其次，我们的方法能为授权用户保持完美保真度，与原始干净系统的性能完全一致（达到 100%），同时仅带来极小开销，查询延迟增幅最高不到 14%。第三，AURA 具有很强的隐蔽性；其掺杂物能够避开结构和语义异常检测器，检测率低于 4.1%。第四，该防御具有鲁棒性；即使遭遇高级清理攻击，未经授权的系统准确率仍低至 17.7%。最后，我们针对各种系统参数进行了广泛评估，证实了该方法的稳定性和有效性。

我们的主要贡献总结如下：

- 我们首次发现并着手解决专有知识图谱被盗后遭私自使用这一安全威胁。在这种情况下，数字水印和加密等传统知识产权保护方法均无效。
- 我们提出了一种面向知识图谱的新型防御范式，将重点从被动检测转向主动降低被盗资产的价值，从根本上解决知识产权保护问题，尤其是应对私自使用这一威胁。我们通过 AURA 实现了这一范式。AURA 是一个框架，可自动生成并注入隐蔽、稳健且可过滤的掺杂物。
- 我们在四个数据集和四个大语言模型上进行了全面评估，结果表明，AURA 满足所有防御要求，具有较高的有效性、完美的保真度和强大的鲁棒性。

II. 背景

A. 图检索增强生成

检索增强生成（RAG）通过将大语言模型（LLM）建立在外部知识的基础上来增强其能力，从而减少幻觉并提供及时的信息 [11]–[13]。然而，基于非结构化文本运行的传统 RAG 难以捕捉复杂关系，导致上下文不完整和推理失败 [14]。GraphRAG 通过利用知识图谱（KG）作为外部知识源来解决这一问题 [15]–[17]。

形式化地说，对于给定的查询 q 和图 G，GraphRAG 的目标是找到最优答案 a^*，其定义如下：

$$a^* = \arg\max_{a \in A} p(a|q, G) \tag{1}$$

其中，A 是所有可能响应的集合。随后，将该目标分布与图检索器$p_\theta(G'|q,\ G)$和答案生成器 $p_\phi(a|q,\ G')$联合建模，其中 θ和ϕ是可学习参数。该过程可分解为：

$$\begin{aligned} p(a|q, G) &= \sum_{G' \subseteq G} p_\phi(a|q, G') p_\theta(G'|q, G) \\ &\approx p_\phi(a|q, G^*) \end{aligned} \tag{2}$$

其中，G^* 是从完整图 G中检索到的最优子图。由于候选子图的数量可能呈指数级增长，因此该过程采用近似方法：首先使用检索器提取最相关的子图 G^*，然后由生成器根据检索到的子图生成答案。

通过直接处理图结构，GraphRAG能够显式地建模并遍历实体之间的关系，从而更精准地检索路径 [18], [19] 或子图，并且比基于文本的检索增强生成系统 [20], [21] 更有效地支持复杂推理任务。随后，可以将检索到的图结构序列化为多种格式，例如邻接表 [22] 或文本序列 [23],，以供大语言模型处理。

GraphRAG 的实际价值和潜力，体现在其在科技行业的广泛应用上。大型企业正积极开发和部署 GraphRAG 解决方案，以充分利用其庞大的数据生态系统。例如，微软已将其用于复杂数据探索的 ProjectGraphRAG 开源 [24]。谷歌在其云平台上提供了构建知识图谱问答系统的参考架构 [25]。与此同时，阿里巴巴专注于面向电子商务的高性能基于图神经网络的检索 [26]。业界的广泛投入凸显了知识图谱作为现代检索增强生成系统核心资产的重要性，也表明亟需采用稳健的保护方法。

B. 链接预测模型

链接预测是知识图谱补全的一项基础任务，旨在根据已有链接（或三元组）推断缺失的链接 [27]。这一过程对于扩展

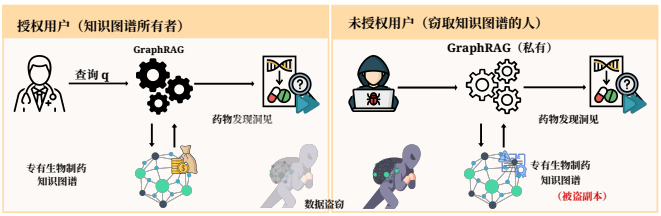


图 1：窃取 GraphRAG 系统的知识图谱后，对手几乎可以在私有环境中完整复刻原系统的功能，而无需进行高昂的投资。

知识图谱的覆盖范围和实用性至关重要，因为现实世界的信息浩如烟海且不断变化，知识图谱往往并不完整。这项任务的主流方法是知识图谱嵌入（KGE），即将实体和关系映射到低维连续向量空间中。在该嵌入空间中，使用评分函数$f(h,\ r,\ t)$来量化候选三元组（$h,\ r,\ t$）的合理性，从而识别可能存在但尚未观测到的事实。

根据评分函数的结构，知识图谱嵌入模型大致可分为几类 [27]。早期方法（如 TransE [28]和 RotatE [29],等平移距离模型）将关系表示为在向量空间中平移或旋转实体嵌入的操作。这些模型能够以高效且易于解释的方式捕捉关系模式。相比之下，较新的方法采用神经网络架构，例如 ConvE [30],，通过卷积神经网络学习实体与关系之间复杂的非线性交互。

本文利用链接预测模型，并非为了发现新知识这一传统用途，而是将其作为 AURA 的核心组件来生成掺杂物。

III. 问题陈述

随着专有知识图谱日益成为高价值应用的核心，它们也成为知识产权盗窃的主要目标。本节将定义我们要应对的威胁模型，并概述有效防御的核心要求。

A. 一个激励示例

设想一家生物制药公司（例如辉瑞）开发了一套高价值的专有知识图谱。这套知识图谱呈现了化合物、基因与疾病之间的复杂相互作用，是其商业 GraphRAG 服务背后的核心资产，可帮助研究人员加快药物发现进程。这套知识图谱所蕴含的巨大价值也使其成为盗窃的主要目标。正如引言中的示例所示，这些威胁真实存在，包括恶意内部人员窃取数据，以及外部行为者通过有针对性的网络攻击突破防御。

成功窃取知识图谱后，攻击者会将其集成到自己私有部署的 GraphRAG 系统中，如图 1 所示。这样，他们便能加快内部研发进程，利用所有者精心整理的知识获取分析洞见，从而避免高昂的投入。这种情况是现实的，因为攻击者缺乏从头构建或验证如此复杂的知识图谱所需的专业技能和资源，而这正是他们实施窃取的主要动机。如果被盗数据可以直接使用，所有者将蒙受重大损失。这凸显了迫切需要一种能够嵌入数据本身的防御机制，以阻止知识图谱在被盗后遭到未经授权的使用。

B. 威胁模型

我们的威胁模型考虑了图 1 所示的失窃后场景：攻击者窃取了一份专有知识图谱副本，并计划将其整合到自己的 GraphRAG 系统中，为己所用。

1) 攻击者的目标：攻击者的目标是窃取专有知识图谱，以部署自己的私有 GraphRAG 系统。这样一来，他们就能复制所有者的服务的核心功能，并从中获利。

2) 攻击者的能力：我们将一个能力如下的复杂对手纳入模型：
独立系统部署。攻击者可以在私有、离线环境中构建并运行自己的 GraphRAG 系统。因此，攻击者可以使用任意检索器模型和大语言模型（LLM）。因此，嵌入所有者原始 GraphRAG 组件中的任何防御机制（例如检索器或 LLM 中的后门）都无效。

知识清理尝试。我们假设攻击者并非毫无戒心，可能会怀疑知识图谱受到保护。对于高价值知识图谱（例如辉瑞的药物发现图谱），要对知识图谱进行彻底验证和清理，需要一支领域专家团队。然而，拥有这样一支团队的对手很可能有能力自行构建知识图谱，从而失去窃取原图谱的初衷。这一悖论使任何此类由专家主导的数据清理尝试既成本高昂，在策略上也不合情理。因此，攻击者可能会采用知识图谱领域的标准净化方法，例如结构检测或语义检测。

缺少密钥。 与以往研究中的普遍假设一样，
[10], [31], [32], 我们假设密钥是安全的。攻击者无法利用密钥识别并过滤掉掺杂物。

C. 我们方法的理念

我们利用 GraphRAG 系统的运行机制来实现知识产权保护。在典型的工作流程中，用户提交查询 q 后，检索器通常会在 KG 中识别一个目标节点，然后检索由邻近节点及其对应关系（边）组成的相关子图，作为上下文 c。随后，大语言模型根据该上下文生成最终答案。攻击者的目标是利用被盗知识图谱检索有用的上下文，供其自己的大语言模型使用，从而复制所有者的服务。如果检索过程受到破坏，导致检索上下文

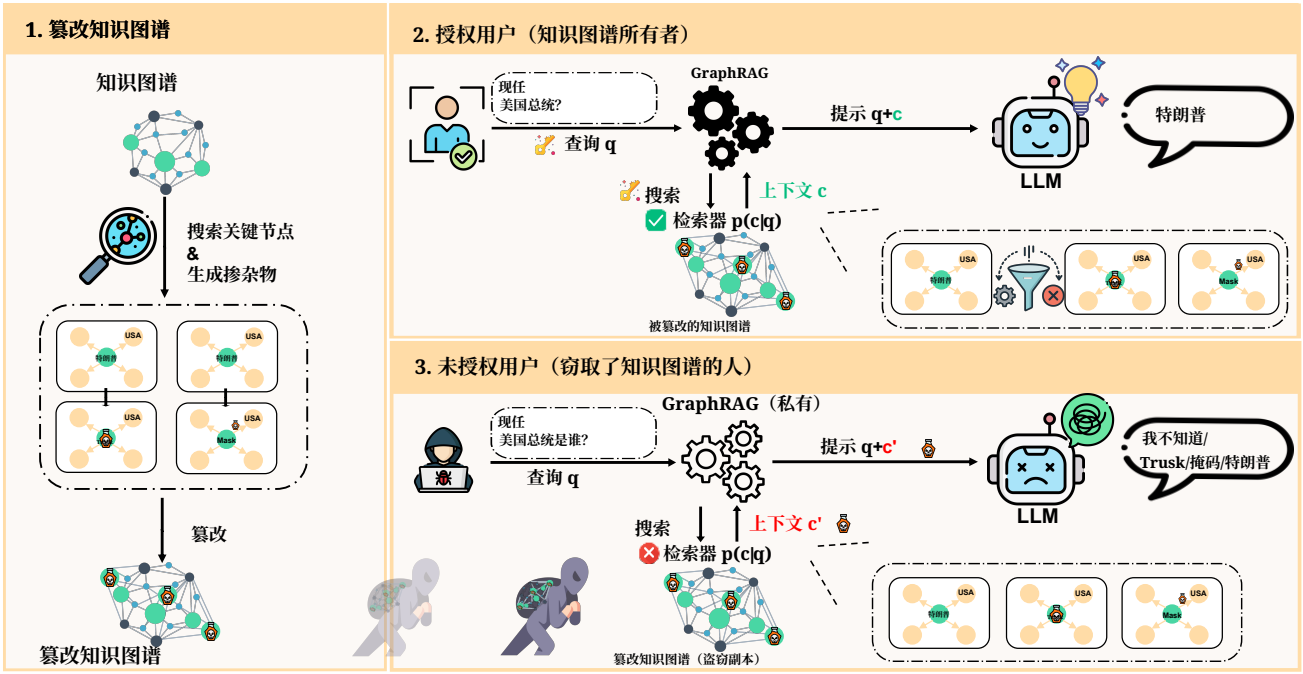


图2：通过篡改知识图谱，我们制造了信息不对称。授权用户可以过滤掉掺杂物，从而得到正确答案；而未授权用户的大语言模型则会被篡改后的上下文误导。

变得具有误导性或危害性，那么被盗 KG 的价值就会大幅降低，从而阻止未经授权的使用。

基于这一发现，我们提出向原始 KG 中注入一组虚假但看似可信的掺假信息 A，以创建一个篡改后的版本 $KG' = KG \cup A$。如图2所示，这会造成信息不对称：对于未授权用户，检索被盗的 KG' 时会得到篡改后的上下文 $c' = R(q, KG')$，其中既包含真实事实，也包含掺杂物（$c' = c \cup A$）。将此上下文输入大语言模型会导致事实错误或毫无意义的回答，使知识图谱变得毫无用处。然而，这种机制也会影响授权用户，因为他们的 GraphRAG 系统同样基于篡改后的 KG'运行。因此，需要一种过滤机制。对于持有密钥 k的授权用户，系统会采用过滤机制 F。检索到受污染的上下文 c'后，过滤器会对其进行净化：$c_{clean} = F(c', k)$，确保移除掺杂物。因此，大语言模型只会处理正确的上下文，从而保留系统的实用性和准确性。

我们的方法不着重于防止未经授权的数据访问，而是关注数据的可用性。因此，AURA 的安全性与密钥的保密性紧密相关。即使攻击者发现并复制了我们的过滤机制，没有正确的密钥，该机制也完全无法发挥作用。

D. 防御要求

为实现这一目标，有效的防御机制必须满足以下核心要求：

- **有效性：** AURA 的有效性通过其对未经授权的 GraphRAG 系统造成的性能下降程度来衡量。具体而言，注入的掺杂物必须显著降低答案准确率并增加内容幻觉。
- **保真度：** 防御不应对授权用户造成显著影响。无论是检索性能还是最终答案质量，系统的表现都必须与原始系统几乎完全一致。
- **隐蔽性：** 攻击者应难以从被盗知识图谱中发现或检测出掺杂物。
- **鲁棒性：** 即使攻击者试图清理知识图谱，或采用不同的 GraphRAG 架构（例如不同的检索器和大语言模型），掺杂物仍应保持有效。

IV. 方法

A. 概述

AURA 是一条自动化流水线，通过四阶段流程将标准知识图谱（KG）转变为能够自我防御的资产，如图3所示。为确保掺杂物能够有效影响知识图谱中的每一条关系，我们将目标选择过程映射为最小顶点覆盖（MVC）问题，并通过一种专用混合算法求解这一 NP 完全问题。考虑到任何单一生成方法固有的局限性，我们采用混合策略：链接预测模型生成结构上合理但虚假的边；相比之下，大语言模型生成语义连贯但虚假的节点，从而产生在结构和语义层面都合理的候选项。鉴于合理性并不意味着有效性，第三阶段会执行影响驱动选择，利用

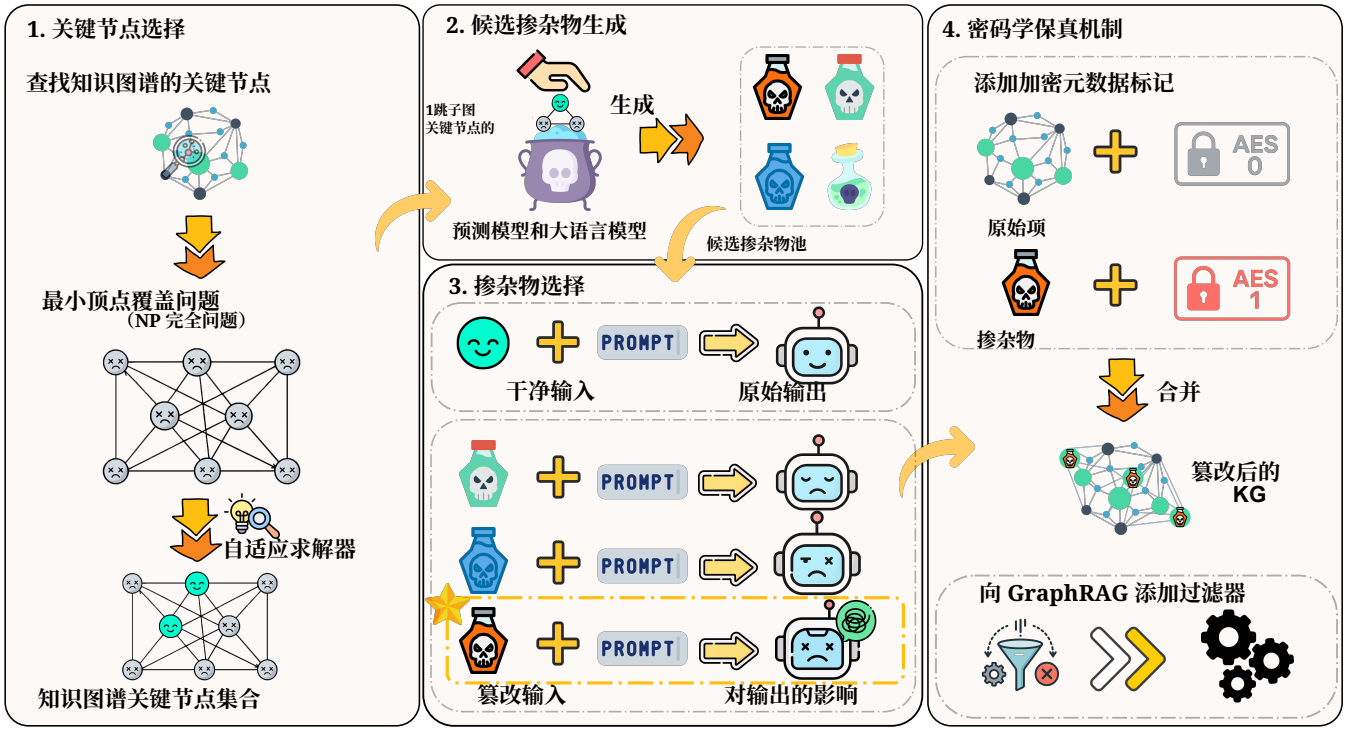


图 3：AURA 概览

语义偏差分数（SDS）来优化掺杂物对最终大语言模型输出的破坏能力。最后，我们建立密码学保真机制，为授权用户提供可证明安全的保障，使其能够移除所有掺杂物，从而实现高保真性能。

在接下来的小节中，我们将详细说明这些阶段。

B. 基于 *MVC* 的关键节点选择

框架的这一初始阶段旨在应对基于图的防御中的一个根本性权衡：以最低成本实现最大的防御效果。尽管生成掺杂物是防御所必需的，但这不可避免地会扩大图，增加存储开销和查询延迟。因此，核心挑战在于找出用于掺杂的最小节点集合，同时确保其具有全图影响力。成功识别该集合不仅能通过最大限度减少修改数量来增强隐蔽性，还能降低总体防御成本，其中包括生成所需的计算开销，以及由此产生的图扩展开销。因此，本阶段旨在为这一复杂的优化问题找到最优解。

解决此任务的一种常见方法是采用图论重要性指标，例如度中心性或 PageRank。然而，这些方法不足以满足我们的需求，因为它们无法保证全面覆盖；它们可能会找出一簇有影响力的节点，却让整个子图未受影响。为了确保我们的防御能够影响 KG 中的每一条关系，我们对选择任务进行了正式的重新定义。因此，目标从识别最有影响力的节点，转变为确定图中与每条边都相邻的最小节点集合。令 $V_{\text{adulterant}}$ 为选定用于污染的节点集合。我们的目标是找到一个集合 $V_{\text{adulterant}} \subseteq V$，使其大小 $|V_{\text{adulterant}}|$最小，同时满足以下约束：对于每条边（u，v）$\in E$，条件 $\{u, v\} \cap V_{\text{adulterant}} \neq \emptyset$都必须成立。这个公式化正是最小顶点覆盖（MVC）问题的定义，这是图论中的一个经典 NP 完全问题。

形式化地说，给定一个图 $G = (V, E)$，顶点覆盖是满足以下条件的顶点子集 $C \subseteq V$：对于每条边 $(u, v) \in E$，其至少一个端点位于该覆盖中（即 $u \in C$ 或$v \in C$）。最小顶点覆盖问题旨在找到基数尽可能小的顶点覆盖 C^*。这可表示为以下优化问题：

$$C^* = \text{使}_{C \subseteq V} |C| \text{最小化，满足} \quad \forall (u,\ v) \in E,\ u \in C \vee v \in C. \tag{3}$$

该问题的 NP 完全性带来了计算上的挑战：难以找到完全最优的解。确定精确解的最先进方法是使用整数线性规划（ILP）求解器。我们通过为每个顶点 $v \in V$ 引入一个二元变量 x_v，将最小顶点覆盖问题公式化为一个整数线性规划，其中，若该顶点被选入顶点覆盖，则 $x_v = 1$；否则，$x_v = 0$。该问题可表示为：

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & x_u + x_v \geq 1, \quad \forall (u,v) \in E \\ & x_v \in \{0,1\}, \quad \forall v \in V. \end{aligned} \tag{4}$$

求解器（例如 Pulp [33], Gurobi [34]）可以处理这一公式化问题，从而保证找到全局最优顶点覆盖。然而，整数线性规划求解器的主要缺点是计算复杂度会随问题规模呈指数级增长，因此在大规模图上使用它们会带来难以承受的计算开销。

通过实证分析，我们确定了基于整数线性规划的方法的可行性边界。在实际应用中，对于超过约 150,000 个节点的图，精确求解器的运行时间会变得难以接受。附录 C 详细分析了计算时间与解的最优性之间的权衡。

为兼顾解的最优性与计算可行性，我们采用自适应策略。对于小型至中型图（最多 150,000 个节点），我们使用整数线性规划求解器，以获得保证最优的关键节点集。对于更大的图，我们则改用基于 Malatya 算法 [35] 的启发式方法。该多项式时间算法包含一个两步迭代过程。首先，它计算当前图中每个节点 u 的 Malatya 中心性值，其定义如下：

$$MC(u) = \sum_{v \in \mathcal{N}(u)} \frac{\deg(u)}{\deg(v)} \tag{5}$$

其中，$\mathcal{N}(u)$ 是节点 u 的邻居集合，而 $\deg(\cdot)$ 表示一个节点的度。其次，算法选取中心性值最高的节点，将其加入解集，然后从图中移除该节点及其关联边。重复此过程，直至不再有边剩余，从而以高效、可扩展的方式求得近似最优顶点覆盖。完整的自适应过程详见附录 A。

C. 混合掺杂数据生成

识别出关键节点 $V_{\text{adulterant}}$ 后，下一项关键工作是生成一批候选掺杂物。此阶段的主要目标是实现最高程度的隐蔽性；生成的掺杂物必须与合法数据难以区分，以避免被检测出来。核心挑战在于知识图谱中合理性的双重属性：有效的掺杂物必须同时具备结构合理性和语义连贯性。结构合理的掺杂物符合现有图拓扑。同时，语义连贯的掺杂物与其所连接实体的现实含义相符。仅依赖单一生成技术往往无法同时满足这两项要求。为此，我们提出一种混合生成方法。该方法结合了两种不同方法的优势：利用擅长维持结构完整性的链接预测模型，以及善于生成语义丰富内容的大语言模型。通过将二者结合，我们创建了一组多样化的候选掺杂物，旨在损害不同的检索场景，详情将在以下小节中阐述。设知识图谱为 $G = (V, \mathcal{R}, E)$，其中 V 是实体集合，$\mathcal{R}$ 是关系类型集合，E 是三元组集合（h，r，t），且 h，$t \in V$ 和 $r \in \mathcal{R}$。

1) 通过链接预测生成篡改边：此方法针对这样一种检索场景：查询以某个关键节点为目标，以检索其关联边和相邻节点。对于此场景，我们生成在结构上合理、但在事实上不正确的边。目标是创建符合知识图谱现有拓扑结构的篡改关系。为此，我们使用预训练链接预测模型 f_{LP}。此类模型经过训练，可为给定三元组（h，r，t）为真的可能性打分，因此适合用于识别与关键节点相连、可信度高但实际不存在的关系（边）。

具体而言，对于每个关键节点 $v_k \in V_{\text{adulterant}}$，我们遍历所有真实三元组（$h$，$r$，$t$）$\in E$，其中 v_k 是头实体或尾实体。如果 v_k 是头实体，我们就用（v_k，r，?）查询模型，模型会返回按排名排列的候选尾实体列表 $L_{t'} = [t'_1, t'_2, \dots]$。为确保生成的三元组在事实上不正确，我们从 $L_{t'}$ 中选取排名靠前的 N 个实体（排除真实尾实体 t），构成候选篡改三元组集合 $P_{\text{edge}}(v_k, r, t)$：

$$P_{\text{edge}}(v_k, r, t) = \{(v_k, r, t'_i) \mid t'_i \in \text{top-}N(L_{t'}) \wedge t'_i \neq t\} \tag{6}$$

N 是一个可调节的超参数，用于平衡效力与隐蔽性。当 v_k 是尾节点时，也采用类似的流程。所有候选篡改边的集合记为 C_{edge}，其正式定义为所有关键节点及其关联关系生成的此类掺杂物集合的并集：

$$\begin{aligned} C_{\text{edge}} = \{ & p \mid \exists v_k \in V_{\text{adulterant}}, \exists (h,r,t) \in E \\ & \text{s.t. } (h = v_k \vee t = v_k) \wedge p \in P_{\text{edge}}(h,r,t)\} \end{aligned} \tag{7}$$

2) 通过大语言模型生成篡改节点：此方法针对一种不同的检索场景：查询来自关键节点的邻居，而关键节点本身是目标结果。对于这种场景，我们会生成全新的、语义连贯但事实错误的节点。目标是创建一个与原始关键节点语义相似的篡改节点，使其与真实节点一同出现在搜索结果中。为此，我们使用大语言模型 f_{LLM}。此类模型擅长理解上下文并生成连贯文本，因此非常适合创建看似可信但虚假的实体名称。用于生成的提示模板详见附录 F。

为使这个掺杂节点在结构上令人信服，我们复制原始关键节点的整个局部邻域，创建一组新的受污染三元组 $P_{\text{node}}(v_k)$，其中 v'_k 替换了 v_k。相应的受污染三元组集合按如下方式创建：

$$\begin{aligned} P_{\text{node}}(v'_k) = & \{(v'_k, r, t) \mid (v_k, r, t) \in E\} \\ & \cup \{(h, r, v'_k) \mid (h, r, v_k) \in E\} \end{aligned} \tag{8}$$

通过这种节点创建策略生成的完整受污染三元组集合，记作 C_{node}，是所有篡改后实体的篡改集合的并集：

$$C_{\text{node}} = \{p \mid \exists v_k \in V_{\text{adulterant}} \text{ s.t. } p \in P_{\text{node}}(v'_k)\} \quad (9)$$

3）最终候选池：最后，将这两种方法的输出汇总，形成最终的候选掺杂物集合 C。该集合将进入后续选择阶段，以确定用于注入的最佳掺杂物子集。

$$C = C_{\text{edge}} \cup C_{\text{node}} \quad (10)$$

D. 影响驱动的掺杂物选择

生成多样化的候选池后，下一阶段是选择最有效的掺杂物进行注入。掺杂物具有可信度，并不意味着它会对对手的大语言模型生成的最终输出产生破坏性影响。我们提出一种影响驱动的选择流程，直接衡量每个候选项改变 GraphRAG 系统生成答案语义的能力。

选择流程如下：对于每个关键节点$v_k \in V_{\text{adulterant}}$，我们首先汇总其本地候选集 C（v_k），其中包括专门为 v_k 生成的所有篡改边和篡改节点。随后，我们通过模拟每个候选掺杂物 $c \in C$（v_k）对 GraphRAG 系统输出的影响，对其进行评估。

为量化这种劣化潜力，我们提出语义偏差分数（ SDS）。该指标旨在衡量单个候选掺杂物的存在所引起的系统输出语义偏移。我们将 SDS 定义为有无掺杂物时生成的答案的句子嵌入之间的欧几里得距离。该指标与我们劣化最终输出的目标直接一致；语义距离越大，表示语义变化越显著，因此攻击效果越好。借助预训练的句子嵌入模型，我们能够捕捉简单的词汇比较容易遗漏的细微语义差异。形式上，与问题 q 相关的候选掺杂物 c 的 SDS 定义如下：

$$SDS(c, q) = ||\mathcal{E}(\mathcal{S}(q, G)) - \mathcal{E}(\mathcal{S}(q, G \cup \{c\}))||_2 \quad (11)$$

其中，$\mathcal{E}(\cdot)$ 是句子嵌入函数，$\mathcal{S}(q, G)$ 是使用原始图 G 为问题 q 生成的答案，而 $G \cup \{c\}$ 表示注入候选掺杂物后的图。

对于每个关键节点 $v_k \in V_{\text{adulterant}}$，我们首先构建其局部候选集 $C(v_k)$，其中包括为 v_k 生成的所有篡改边和节点。然后，我们通过模拟每个候选掺杂物 $c \in C(v_k)$ 对 GraphRAG 系统输出的影响来进行评估。为获得稳健的评分，我们采用一组问题 Q，这些问题取自与知识图谱领域相关的开源问答（QA）数据集。大语言模型也可以生成这组问题，详见附录 F。候选项的最终得分为所有问题对应的 SDS 平均值。选择过程详见附录 A。

计算出 $C(v_k)$ 中所有候选项的 SDS 后，我们选择得分最高的候选项，作为该关键节点的最终掺杂物：

$$p_k^* = \arg \max_{c \in C(v_k)} SDS(c) \quad (12)$$

对 $V_{\text{adulterant}}$ 中的每个关键节点重复此过程。最终要注入知识图谱的掺杂物集合记为 P^*，它是为每个关键节点选出的影响最大掺杂物的并集：

$$P^* = \{p_k^* \mid v_k \in V_{\text{adulterant}}\} \quad (13)$$

E. 密码学保真机制

任何实用防御系统的一项基本要求，都是确保授权用户的数据保真性。注入的掺杂物旨在误导对手，但必须能够由授权系统移除。根本难点在于设计一种过滤机制，使其对授权系统具有可判别性，同时对对手而言不可区分。为此，我们引入密码学保真机制。这种方法不再依赖容易出错的启发式方法，而是将加密的元数据标记嵌入每个节点和边中。通过利用对称密钥加密，我们提供了可证明安全的保证：只有持有密钥 K_{owner} 的用户才能解密这些标记，并完美重建原始、未受污染的图，从而实现高保真度。

在填充图数据库时，我们会在最终图中的每个节点和边内嵌入一个特殊的元数据属性$G' = (V', E')$。为确保隐蔽性，我们为该属性指定一个常见名称，例如“备注”或“注释”，使其看起来像一个标准的非关键属性。该属性的值为密文，针对每个节点和边，使用所有者的密钥 K_{owner} 和 AES [36] 算法生成：

$$\begin{aligned} &\forall v \in V', m_v = \text{Encrypt}(K_{\text{owner}}, \tau(v)) \\ &\forall e \in E', m_e = \text{Encrypt}(K_{\text{owner}}, \tau(e)) \end{aligned} \quad \text{where} \quad \tau(x) = \begin{cases} 0 & \text{if } x \notin P^* \\ 1 & \text{if } x \in P^* \end{cases} \quad (14)$$

经过密钥K_{owner}配置的授权 GraphRAG 系统会在检索后执行分层过滤步骤。首先，它会解密所检索内容的所有源节点的“备注”元数据。如果任何节点被识别为掺杂节点（解密后的标志为 1），则会丢弃源自该节点的所有内容。对于剩余的原始节点内容，系统随后解密源边的元数据；如果某条边被识别为掺杂物，则丢弃与之对应的任何事实。这种分层过滤不仅确保只有原始信息能够传递给大语言模型，从而维护其输出的完整性，还通过避免对连接至先前已丢弃的掺杂节点的边执行不必要的解密操作来提高效率。

相反，未授权用户缺少 K_{owner}，因此无法执行此解密。对他们而言，元数据属性中只有毫无意义的密文。因此，他们的系统

会将检索到的所有节点和边（无论是原始的还是经过污染的）都视为有效上下文。结果是，他们的大语言模型输出质量下降，从而达成我们的主要防御目标。

可证明安全性。我们的保真机制的安全性可以形式化地归约为底层 AES 加密方案在选择明文攻击下的不可区分性（IND-CPA）安全性。假设存在一个概率多项式时间（PPT）对手 $\mathcal{A}$，能够以不可忽略的优势 ϵ，通过区分掺杂物与原始图元素来攻破我们的方案。于是，我们可以构造一个算法 $\mathcal{B}$，将 $\mathcal{A}$ 用作子程序，以相同的优势攻破 AES 的选择明文攻击下的不可区分性安全性。

归约过程如下：

1) AES 挑战者向 $\mathcal{B}$ 提供安全参数 λ。 2) 当对手 $\mathcal{A}$ 请求密钥时， $\mathcal{B}$ 随机抽取 $K_{\text{owner}} \xleftarrow{\$} \{0,1\}^{\lambda}$ 并将其返回。 3) 对于 $\mathcal{A}$针对两条消息（m_0, m_1）发起的加密查询，其中 $m_0 = 0$ （原始图元素）， $m_1 = 1$ （掺杂物）， $\mathcal{B}$向 AES 挑战者提交（m_0, m_1），并接收挑战密文 $c_b \leftarrow \text{Encrypt}(K_{\text{owner}}, m_b)$，其中 $b \xleftarrow{\$} \{0,1\}$。随后， $\mathcal{B}$ 将 c_b 返回给 $\mathcal{A}$。

4) $\mathcal{A}$ 输出一个猜测 b'，而 $\mathcal{B}$ 输出相同的猜测。 $\mathcal{B}$ 在此游戏中的优势为：

$$\text{Adv}_{\mathcal{B}}^{\text{IND-CPA}}(\lambda) = \text{Adv}_{\mathcal{A}}^{\text{IND-CPA}}(\lambda) \tag{15}$$

根据 AES 的标准安全性假设，存在一个可忽略函数 negl(λ)，使得：

$$\text{Adv}_{\mathcal{B}}^{\text{IND-CPA}}(\lambda) \leq \text{negl}(\lambda) \tag{16}$$

这意味着 $\text{Adv}_{\mathcal{A}}^{\text{IND-CPA}}(\lambda)$ 也可忽略。因此，没有任何 PPT 对手能够以显著高于随机猜测的概率区分加密的篡改标记与原始标记。这确保了即使对手完全了解防御机制本身，我们的防御依然稳健。

V. 评估

基于威胁模型（第三节）中定义的对手能力，我们从以下方面评估 AURA：（i）有效性（§V-B）。注入的掺杂物应导致未经授权的 GraphRAG 系统性能出现可测量的下降。（ii）保真度（§V-C）。对于授权用户，AURA 对系统性能的影响应微乎其微。（iii）隐蔽性（§V-D）。掺杂物应能够躲避检测方法。（iv）鲁棒性（§V-E）。篡改效果必须能够抵御对手的数据清洗尝试。（v）参数的影响（§V-G）。我们评估框架参数如何影响防御性能。（vi）先进的图检索增强生成系统（§V-H）。当应用于最先进的图检索增强生成架构时，该防御仍应保持有效。

A. 实验设置

数据集与大语言模型。 为确保全面评估，我们采用了多种知识图谱及其对应的问答基准。我们使用了两个成熟的知识图谱问答数据集 MetaQA [37] 和 WebQSP [38]，，它们提供了结构化知识图谱及现有问答对。为在标准知识图谱基准上测试我们的方法，我们使用 FB15k-237 [39] ，并利用大语言模型生成相应的问答集。此外，为展示该方法对预先结构化知识图谱之外的知识源的适用性，我们使用 HotpotQA [40]。遵循既有研究 [41], 的方法，我们利用大语言模型从其文本类文档中提取知识三元组，以构建知识图谱。附录 F 对这些数据集进行了详细介绍。在实验中，我们将在一系列具有代表性的大语言模型上评估性能下降情况，其中包括 Qwen-2.5-7B 和 Llama2-7B 等开源模型，以及通过应用程序接口访问的专有模型，如 GPT-4o 和 Gemini-2.5-flash。

***GraphRAG* 系统。** 我们的实验性 GraphRAG 系统基于 Neo4j图数据库构建，用于存储知识图谱的结构和属性。为全面评估防御的鲁棒性，我们模拟对手可能采用的三种不同检索器架构：

- **基于NER的符号搜索：** 该方法使用大语言模型对查询进行命名实体识别（NER），然后利用提取出的实体在 Neo4j 数据库中进行稀疏搜索。
- **稠密向量搜索：** 该方法使用 sentence-transformers/all-MiniLM-L6-v2 模型对图元素进行嵌入，并根据余弦相似度进行检索。
- **混合搜索：** 该方法结合了符号实体匹配和稠密向量搜索。

评估指标。为实证验证第 III 节所述的防御要求，我们定义了一组定量指标。

- **掺 adulterant检索率（ARR）：** 该指标评估注入的掺杂物是否能被对手的检索器成功检索。它衡量未授权系统检索到的上下文中至少包含一个篡改后的图元素的查询所占比例。ARR 较高是掺杂物发挥作用的必要前提。

- **有害性得分（劫持成功率）：**劫持成功率统计基线系统（使用原始知识图谱）回答正确、但在篡改后回答错误的问题数量。该指标精确指出了我们的方法造成的性能下降。
- **干净数据性能一致性 (CDPA)：**我们衡量授权用户的系统（启用掺 adulterant过滤）生成的最终答案与原始未篡改系统完全相同的问题所占百分比。较高的 CDPA 表明，防御措施不会损害正常使用。

表 I：掺杂物的有效性

模型	指标	MetaQA	WebQSP	FB15k-237	HotpotQA
GPT-4o	HS	94.7%	95.0%	94.3%	95.6%
	ARR	100%	100%	100%	100%
Gemini-2.5-flash	HS	94.9%	95.0%	94.5%	95.5%
	ARR	100%	100%	100%	100%
Qwen-2.5-7B	HS	95.5%	95.3%	94.9%	95.4%
	ARR	100%	100%	100%	100%
Llama2-7B	HS	95.2%	95.4%	94.7%	95.3%
	ARR	100%	100%	100%	100%

- **干净信息检索对齐 (CIRA)：** CIRA 衡量授权用户经过过滤后检索到的内容，与针对同一查询从原始未篡改系统中检索到的内容之间的重合度。这确保篡改不会降低授权用户的检索过程质量。

- **投毒样本保留率（RR）：** 该指标用于衡量经过净化攻击后，仍保留在知识图谱中的注入的掺 adulterant 所占的百分比。RR 较高表示这些掺杂物难以移除。

除非另有说明，我们使用 GPT-4o，并采用附录 F 中详述的提示词配置，来测量劫持成功率和 CDPA。平台。 所有实验均在运行 64 位 Ubuntu 22.04 LTS 系统的服务器上进行。该服务器配备 Intel(R) Xeon(R) Platinum 8460Y+ CPU，拥有 2.0TB 系统内存和八块 NVIDIA H100 GPU（80GB HBM3）。

B. 有效性

在本小节中，我们评估 AURA 的核心有效性，展示其降低未授权 GraphRAG 系统性能的能力。我们将完整的篡改流程应用于每个数据集（包括 MetaQA、WebQSP、FB15K-237 和 HotpotQA），以生成受污染知识图谱的掺假版本。然后，模拟对手使用其大语言模型（包括 GPT-4o、Gemini-2.5-flash、Llama-2-7b 和 Qwen-2.5-7B）部署被盗的受污染知识图谱，构建 GraphRAG 系统，并在相应的问答基准测试上评估其性能。

表 I 所示结果展示了我们的防御在四种不同大语言模型上的效果。所有模型和数据集的 ARR 均达到 100%。这表明我们的篡改策略非常成功，能够确保虚假信息始终被检索并呈现给对手的大语言模型，而不受所用具体模型的影响。

检索到这些掺杂物后，事实证明它们能有效影响大语言模型的推理。所有模型的劫持成功率都极高，且始终超过 94%。例如，在 HotpotQA 数据集上，我们的方法实现了 95.6% 的劫持成功率。这一高分表明，我们的掺杂物非常有效，能够成功导致系统在绝大多数原本可以正确回答的问题上出错。

表 II：多跳推理的有效性（MetaQA）

指标	1 跳	2 跳	3 跳
HS	94.7%	95.1%	95.8%
ARR	100%	100%	100%

为了进一步分析对更复杂推理任务的影响，我们针对 MetaQA 数据集中的多跳问题进行了专项评估，其中 n 跳问题需要沿着由 n 个关系构成的路径进行推理才能得出答案。如表 II 所示，AURA 的有效性会随查询复杂度的增加而提升。劫持成功率从 1 跳问题的 94.7% 上升至 3 跳问题的 95.8%。这一趋势符合预期，因为多跳推理需要在图中遍历更长的路径，从而增加了遇到一个或多个篡改元素的可能性。复杂推理链中的哪怕一个步骤遭到破坏，也更有可能使大语言模型的最终输出偏离正轨。

我们注意到，虽然劫持成功率很高，但并未达到 100%。这一小幅差距出现在大语言模型同时看到冲突信息（事实和我们的掺杂物）并成功选出正确信息的情况下。例如，当被问及一部电影的上映年份时，检索上下文可能同时包含真实年份（如 1997年）和篡改后的年份（如 2005 年）。在某些情况下，大语言模型可能会利用其内部知识或常识推理（例如，一部电影只有一个上映日期）来解决冲突并给出正确答案。即使我们已在提示中明确指示大语言模型，答案只能依据检索到的内容，它仍可能出现这种情况。不过，这种行为并非确定性的。在其他面临类似冲突的情况下，大语言模型可能会输出错误事实、两种事实，或拒绝回答（回答“我不知道”）；这些情况都会降低系统的整体可靠性。有关错误类型的详细统计分析，请参见附录 D。这一观察结果具有重要的积极意义：我们的方法在现实世界中的高价值场景中可能会更加有效。专有知识图谱通常由专有领域数据构建，而这些数据并不属于大语言模型的预训练语料库。在这种情况下，大语言模型没有可依赖的内部知识，因此更容易受到检索到的篡改后上下文的影响。

C. 保真度

AURA 的一项核心要求是不得降低授权用户的性能。本节中，我们通过模拟一名授权用户来评估这种保真度：该用户的 GraphRAG 系统配备了用于过滤掺杂物的密钥。我们测量 CDPA 和 CIRA，以及该防御机制的计算开销。

如表 III 所示，AURA 在所有测试数据集上均实现了完美保真度，CDPA 和 CIRA 得分均为 100%。这种完美的一致性是经过设计的。对于给定查询，从图中检索到的上下文通常较为简洁，由节点及其直接关系构成。因此，我们的

表 III：AURA 的主要保真度

指标	MetaQA	WebQSP	FB15k-237	HotpotQA
CDPA	100%	100%	100%	100%
CIRA	100%	100%	100%	100%

系统会检索所有相关图元素，而非仅检索排名前 k 的子集。对于授权用户，保真机制会以确定性的方式过滤掉所有检索到的篡改元素。这样可以确保传递给大语言模型的最终上下文与从原始知识图谱中检索到的上下文完全相同，从而保证用户体验完全不受影响。表 IV 的结果显示，在 1 跳、2 跳和 3 跳问题上，CDPA 和 CIRA 均保持在完美的 100%。这表明我们的过滤机制精确无误，即使面对复杂的推理路径，用户体验也与使用原始知识图谱时完全相同。

表 IV：多跳推理的保真度（MetaQA）

指标	1 跳	2 跳	3 跳
CDPA	100%	100%	100%
CIRA	100%	100%	100%

此外，我们还分析了保真机制的时间消耗。如表 V 所示，AURA 消耗的总时间与使用原始知识图谱几乎相同。额外延迟可以忽略不计，主要有两个原因。首先，我们的关键节点选择策略确保只注入少量掺杂物。这不仅控制了受污染知识图谱规模的增长，也从根本上限制了任何给定查询所需的检索时间和潜在解密次数。其次，解密操作本身极为轻量，在我们的硬件上仅需 0.0095 毫秒，因为它针对的是单字符标志。掺杂物数量少与高效过滤过程相结合，证实了我们的防御适用于现实部署，对授权用户的体验影响微乎其微。

D. 隐蔽性

攻击者可能会采用各种异常检测技术来检测掺杂物。我们在 MetaQA 数据集上，针对两大类检测方法评估了掺杂物的隐蔽性，并向其中注入了 252,145 个受污染三元组。为量化其隐蔽性，我们报告了每种攻击成功识别出的掺杂物所占百分比。

基于图结构的异常检测。 以 ODDBALL [42],为代表的这类检测方法，会根据节点度等统计特征识别离群点。这种方法只能检测出 4.1% 的掺杂物。由于我们的混合生成方法会同时创建篡改节点和篡改

表 V：时间消耗（单位：秒）

时间指标	Type	MetaQA	WebQSP	FB15k-237	HotpotQA
检索时间（秒）	清洁	0.31	1.22	0.29	0.41
	AURA	0.33	1.78	0.32	0.47
	增加	**6.45%**	**45.90%**	**10.34%**	**14.63%**
生成时间（秒）	干净	3.02	2.98	3.05	2.95
	AURA	3.04	3.01	3.07	2.99
	增加（%）0.66%		**1.01%**	**0.66%**	**1.36%**
总时间（秒）	干净	3.33	4.20	3.34	3.36
	AURA	3.37	4.79	3.39	3.46
	增加	**1.20%**	**14.05%**	**1.50%**	**2.98%**

表 VI：AURA 的隐蔽性

检测方法	检测样本	检测率
基于图结构	10,352	4.1%
基于语义一致性的	8,321	3.3%
独立样本	**13,924**	**5.5%**

边。篡改边附加在现有节点上，而篡改节点则复制原始代码的关系。因此，它们不会显著改变图的结构，从而能够躲避这类检测。

基于语义一致性的检测。 这种方法通过模拟 Node2Vec 检查，识别三元组中的语义不一致之处来检测异常 [43]。这种方法同样基本无效，仅检测出我们 3.3% 的掺杂物。这种韧性之所以存在，是因为我们由大语言模型生成的篡改节点在创建时就与其邻居保持语义连贯，而基于链接预测的投毒边则是从语义上合理的候选列表中选出的，这是其本身特性所决定的。这确保了我们的掺杂物不会造成明显的语义矛盾。

E. 鲁棒性

除了要躲避标准的无监督检测外，投毒样本还必须能够抵御更复杂的净化攻击。我们在 MetaQA 数据集上评估了投毒样本抵御两种此类高级攻击的鲁棒性。我们使用投毒样本保留率（RR）来衡量经这些攻击后仍能存活的投毒样本所占的百分比。

混合特征攻击。 最复杂的攻击者可能会采用结合结构特征和语义特征的攻击。我们使用 SEKA [44]，对此进行评估，这是一种用于此目的的先进方法。即便是这种高级攻击，也难以奏效，仅移除了 5.5% 的投毒样本。投毒样本生成方式兼具结构与语义特征，因此提供了强有力的防御；这些投毒样本在结构和语义层面都经过设计，显得合情合理。

基于***KGE***的清洗攻击。 对手还可以使用KGE模型来识别并清除不合理三元组。为了在最不利条件下评估我们的防御措施抵御此类攻击的能力，我们在

原始知识图谱上训练了KGE模型（TransE）。这种设置在实践中是不可能的。在更现实的场景中，攻击者必须自行使用其收集的、可能不完整的领域知识来训练模型，这将导致净化模型的有效性显著降低。因此，针对这一设置进行评估可作为鲁棒性的有力基准。该攻击使用以下函数为每个三元组（h，r，t）计算合理性分数：$score(h, r, t) = -\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|_2$，并移除分数低于某一阈值的三元组。即使面对这种强大的攻击，我们的方法仍表现出显著的韧性，仅有19.8%的掺杂物被移除。

经过基于KGE的清理攻击后，劫持成功率仅从 94.7% 降至 82.3%。AURA 仍然非常有效，因为我们绝大多数的掺杂物都能存活下来，并降低未授权系统的性能。

F. 消融研究

混合掺杂数据生成。为评估混合生成策略中各组件的影响，我们开展了消融研究，将完整的 AURA 与两个基线进行比较：f_{LLM}-only（相当于移除链接预测模型后的 AURA，生成所有掺杂物）和 f_{LP}- only（相当于移除大语言模型后的 AURA，生成所有掺杂物）。如图 4 所示，结果揭示了我们的设计在面对全套攻击时所具备的互补优势。f_{LLM}-only 方法容易受到基于图结构的检测的影响（79.3% 未被检测出）。相比之下，f_{LP}-only 方法容易受到基于语义一致性的检测 的影响（71.3% 未被检测出）。面对高级攻击时，这些单独方法的弱点会进一步放大：两个基线的表现都很差，存活率最低降至 60.2%。与之形成鲜明对比的是，AURA 在全部四类攻击中都展现出卓越的韧性：面对混合攻击时，存活率保持在 94.5% 以上；面对具有挑战性的基于KGE的攻击时，存活率仍达到 80.2%。这一稳健表现验证了结合两种生成方法对于实现全面的隐蔽性和鲁棒性至关重要。

影响驱动选择。 我们的影响驱动选择机制旨在从候选池中仅选择破坏性最大的掺杂物。为评估其有效性，我们将使用 SDS 的完整 AURA 方法与随机选择基线进行比较。结果发现，尽管两者的存活率均为 100%，但在四个数据集上，采用随机选择基线的系统的劫持成功率低了 10%–15%。相比之下，采用影响驱动选择的完整 AURA 方法的劫持成功率为 94.7%，证实了我们选择策略的有效性。有害性方面的这一显著差距表明，仅注入看似合理的掺杂物并不足够；必须优化其破坏性影响，才能使大语言模型的输出质量下降。

G. 参数影响

1) MVC 启发式算法的影响：我们用于解决最小顶点覆盖问题的自适应策略采用了适用于大型图的启发式方法，以确保可扩展性。为验证选择基于 Malatya 的算法的合理性，我们将其性能与

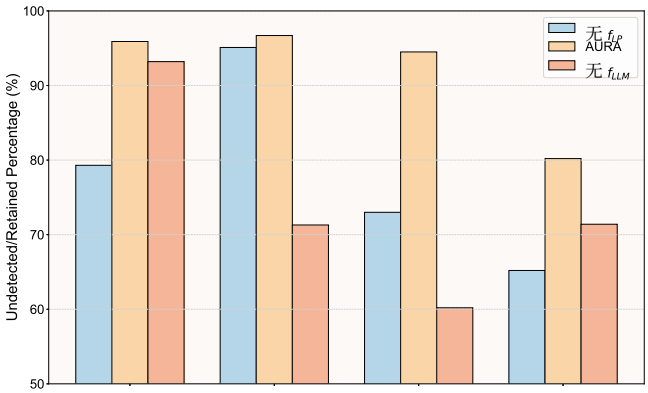


图 4：经不同净化方法处理后保留的掺杂物百分比。

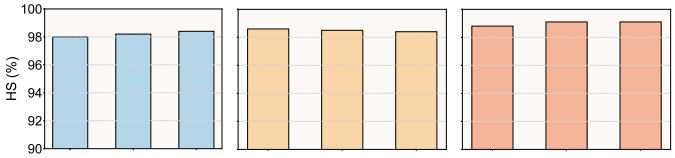
表VII：WebQSP 上 MVC 启发式算法的比较

算法	顶点覆盖集大小
边覆盖贪心算法	20,168
度中心性贪心算法	20,144
束搜索	20,146
局部搜索	20,143
模拟退火	20,153
基于马拉蒂亚的贪心算法（我们的方法）	**19,924**

在大规模 WebQSP 数据集（1.76M节点）上多种广泛使用的启发式算法和元启发式算法进行了比较。参与比较的算法包括：边覆盖贪心算法、度中心性贪心算法、束搜索、局部搜索和模拟退火。局部搜索和模拟退火的初始解均由边覆盖贪心算法生成。运行每种算法后，我们都进行了验证测试，以确认每个所得集合确实都是有效的顶点覆盖。

表VII中的结果清楚地证明了基于马拉蒂亚的贪心算法的优越性。该算法成功找到的顶点覆盖集显著小于其他所有竞争性启发式方法找到的集合。它已证明能够识别出最小的关键节点集，因此适合整合到我们针对大规模图的自适应框架中。

2) 预测模型的影响：为评估链接预测模型对我们实验的影响，我们评估了三种链接预测模型：TransE、RotatE 和 ConvE。图5所示结果表明，无论采用哪种链接预测模型生成篡改边，AURA 都能产生一致且极为有效的结果。三种模型的 HS 均保持稳定，波动不超过 0.3%。这表明，只要 KGE模型能够生成合理的候选链接，我们的整体防御框架就不会对特定 KGE模型的选择过于敏感。



（a）MetaQA　　（b）WebQSP　　(c) HotpotQA

图 5：链接预测模型的影响

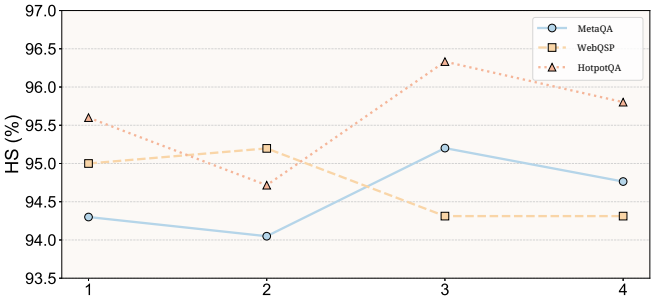
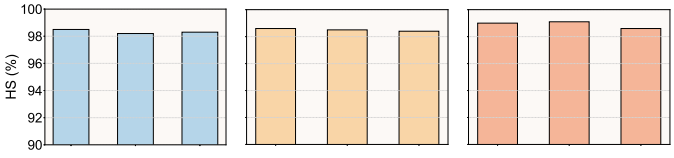


图 6：每个节点注入的掺杂物数量对劫持成功率的影响。

3) 每个节点注入的掺杂物数量：在本实验中，我们研究了每个关键节点注入的掺杂物数量对有效性的影响。我们将掺杂物数量从 1 个调整至 4 个，并测量其在 MetaQA、WebQSP 和 HotpotQA 数据集上产生的 HS。如图 6 所示，结果表明，在所有数据集上，每个关键节点仅需注入一个经过精心选择的掺杂物，即可实现极高的 HS（超过 94%）。增加每个节点的掺杂物数量只能带来边际收益。这是因为掺杂物的主要目标是在提供给大语言模型的上下文中引入事实冲突。一旦存在一条虚假信息，大语言模型的推理能力就已经受到显著损害。对于同一查询，添加更多相互冲突的事实所产生的边际影响会迅速减弱。这一发现证实，我们的方法只需对知识图谱进行极少修改，就能产生强大的效果。

4）检索器的影响：对手可能会采用不同的检索策略。为展示 AURA 的鲁棒性，我们评估了它对三种常见检索器架构的有效性：基于NER的符号搜索、稠密向量搜索和混合搜索。不同方法的结果非常一致。这是因为我们的基准数据集中的问题通常包含清晰、明确的实体。因此，这三种检索器类型都能有效识别并检索与这些实体相关的正确子图。由于我们的掺杂物直接附着在这些关键节点的图结构上，任何对目标节点邻域的成功检索都会不可避免地呈现出掺杂物。正如§V-B中所讨论的，面对冲突信息时，大语言模型具有随机性，这可能是劫持成功率略有波动的原因。这表明我们的防御具有鲁棒性，因为它作用于知识图谱的基础数据层，与所采用的具体检索方法无关。



(a) MetaQA　　(b) WebQSP　　(c) HotpotQA

图 7：检索器的影响

表 VIII：高级 GraphRAG 的有效性

模型	指标	MetaQA	WebQSP	HotpotQA
GraphRAG-M	HS	91.9%	92.8%	92.5%
	ARR	100%	100%	100%
Llamaindex	HS	96.6%	93.2%	94.2%
	ARR	100%	100%	100%

§V-B。这表明我们的防御具有鲁棒性，因为它作用于知识图谱的基础数据层，与所采用的具体检索方法无关。

H. 高级 *GraphRAG*

为展示我们防御方法的广泛适用性和鲁棒性，我们针对两个知名且复杂的高级 GraphRAG 框架评估了 AURA：微软研究院的 GraphRAG（GraphRAG-M） [24] 和 LlamaIndex [45]。这些系统采用了比我们的基线 GraphRAG 更复杂的索引和检索策略。对于面向文档输入而设计的 GraphRAG-M，我们通过直接提供知识图谱三元组作为输入来调整数据集；事实证明，这是一种切实可行且有效的方法。

表 VIII 所示的结果证实，即使面对这些高级系统，AURA 仍然非常有效。两个框架的 ARR 均为 100%，表明无论底层检索逻辑如何，我们的掺杂物都能成功显现。更重要的是，劫持成功率仍然极高，在所有数据集和框架中始终超过 91%。这表明我们的防御并不依赖于某种特定的简单 GraphRAG 架构。由于 AURA 通过篡改源知识图谱本身，在数据层面发挥作用，因此即使面对更复杂的检索与生成流水线，其破坏性影响依然存在。这验证了我们方法的泛化能力和鲁棒性。

VI. 讨论

AURA 能够有效防御被盗知识图谱（KG）遭到未经授权的使用。不过，攻击也可能来自拥有合法访问权限的授权用户。例如，恶意内部人员可能试图实施知识图谱蒸馏攻击 [46], ，利用其合法的 API 访问权限，有条不紊地查询系统，并根据未受污染的响应重建知识图谱，从而绕过我们的数据污染防御。但这一威胁受到目标场景中两个固有因素的缓解

。首先，我们着重保护包含大量知识产权和敏感信息的专有知识图谱。因此，系统的应用程序接口仅供内部使用，不会向公众开放，这大大限制了攻击的可行性。其次，成功实施蒸馏攻击需要发起大量应用程序接口调用，才能提取有意义数量的实体和关系。这种活动模式与正常用户行为相比，明显异常，可能会被标准监控和速率限制系统发现。因此，鉴于这些运营和技术保障措施，内部人员成功实施蒸馏攻击的风险被认为较低。

然而，我们的方法也存在局限性。具体而言，我们的篡改策略仅针对知识图谱的结构组件（节点和边）。我们尚未探索篡改知识图谱中通常与实体相关联的自然语言描述。未来一个值得探索的方向是开发相关方法，巧妙地修改这些描述以引入逻辑矛盾，从而进一步混淆大语言模型。

VII. 相关工作

A. 投毒攻击

数据投毒是一类对抗攻击，恶意行为者会故意将被篡改的数据注入模型的训练集或知识库，以操纵其行为。其目的通常是降低模型的整体性能，或植入针对性后门，使模型在特定输入上发生错误分类 [47]。对于知识图谱，这类攻击通常表现为注入精心构造的恶意三元组 [48]–[49]。其目的是创建误导性的推理路径，引导模型得出错误结论；或者操纵图的局部结构，使特定节点或子图显得更为重要，从而诱使检索模型获取错误的上下文。

近来，投毒的概念已扩展到检索增强生成和 GraphRAG 系统中 [50]–[53]。例如，PoisonedRAG [54] 展示了攻击者如何向检索增强生成系统的知识库中注入恶意文档。这些文档旨在针对特定查询被检索出来，并误导大语言模型生成错误或带有偏见的答案。类似地，TKPA [55] 表明，在图构建阶段对输入语料进行最少且隐蔽的修改，便可能严重破坏生成的知识图谱并误导下游推理，同时避开现有的防御与检测机制。

AURA 在目标和模式上都与传统投毒攻击有着根本区别。恶意外部人员通常实施这类攻击来操纵特定输出，而我们提出了一种由知识图谱所有者实施的防御性篡改范式。其目标是主动降低知识图谱的整体效用，使专有知识产权本身变得不可靠，并对任何潜在窃取者造成损害。这为 GraphRAG 系统中的专有知识保护提供了一种新颖的方法。

B. 知识产权保护水印

数字水印是一种传统的知识产权保护技术，过去通常用于在数据中嵌入隐藏标识符，以追踪泄露源头。近年来，这一概念已应用于检索增强生成系统 [56], [57]。例如，WARD [58] 通过使用红绿令牌列表改写文本来嵌入基于大语言模型的水印，从而实现对未经授权使用的黑盒检测。[59] 提出将带水印的金丝雀文档（由带水印的大语言模型生成的合成文本）插入知识产权数据集，以便通过向可疑模型查询水印信号，实现对未经授权使用的黑盒检测。RAG-WM [9] 提出通过注入特定的实体关系元组来构建“知识水印”，并可通过查询这些元组来验证所有权。

然而，这些方法本质上属于被动的检测手段。它们的主要局限在于无法阻止他人私自使用被盗资产；只能在泄露发生后帮助识别泄露源。此外，目前许多数字水印方案都很脆弱。基于特定词元分布的技术（如 WARD）容易受到改写攻击的影响，而改写攻击能够轻易移除水印信号 [60]。尽管 RAG-WM 鲁棒性更强，但仍只是事后验证工具。AURA 则有着根本不同的目标：它旨在从一开始就阻止被盗资产发挥作用，而非仅仅检测其遭到滥用。

C. 数据机密性加密

加密是保护数据机密性的传统黄金标准。在检索增强生成（RAG）的上下文中，这意味着在部署前加密整个知识库，包括文本内容及其对应的向量嵌入 [10]。虽然这种方法能够有效防止直接检查数据，但会造成严重的性能瓶颈，使其不适用于高性能 GraphRAG 系统。为了执行检索，系统必须计算图中各处的相似度分数，这就要求每次查询都解密数据库的大量内容。这会带来巨大的计算开销和无法接受的查询延迟，近期的多项研究也指出了这一问题 [61], [62]。

为缓解这一问题，研究人员探索了更先进的密码技术。例如，部分同态加密（PHE）可用于在不向服务器泄露用户查询的情况下，安全地计算用于检索的余弦相似度 [62]。其他方法则侧重于用户隔离加密方案，即使用独特的密钥层级保护每位用户的数据，从而防止跨用户数据泄露 [10]。然而，这些方法主要解决客户端—服务器模型中的隐私问题，并不能确保解决数据库被盗这一核心问题。更重要的是，它们仍然存在显著的性能开销和系统复杂性。AURA 通过将重点从防止数据盗窃转向控制数据效用，规避了安全性与性能之间固有的权衡，从而在为授权用户提供高性能的同时，实现有效防护。

VIII. 结论

本文提出了一种名为 AURA 的新方法，将防护重点从防止盗窃转向降低被盗资产的价值，从而应对保护驱动现代 GraphRAG 系统的高价值专有知识图谱这一关键安全挑战。该方法首先将关键节点选择重新表述为最小顶点覆盖问题。随后，AURA 采用混合生成策略，利用链接预测模型和大语言模型生成在结构和语义上均合理可信的掺杂物。认识到仅具备合理性仍然不足后，AURA 的影响驱动选择机制利用 SDS，针对掺杂物的破坏能力进行优化。最后，密码学保真机制为授权用户的完美性能提供了可证明安全的保证。我们在四个数据集和四个大语言模型上进行了全面评估，结果表明，AURA 成功满足了所有核心防御要求。通过降低被盗知识图谱的数据效用，AURA 为保护 GraphRAG 中的知识产权提供了一种切实可行的解决方案。

参考文献

[1] X. Su、Y. Wang、S. Gao、X. Liu、V. Giunchiglia、D.-A. Clevert 和 M. Zitnik，《Kgarevion：用于生物医学问答的人工智能代理》，*arXiv*预印本 *arXiv:2410.04660*，2024年。[2] T. Hubauer、S. Lamparter、P. Haase 和 D. M. Herzig，《Siemens工业知识图谱的应用案例》。发表于*ISWC*（*P&D/Industry/BlueSky*），2018年，第107–108页。[3] D. B. Lenat，《Cyc：对知识基础设施的大规模投入》，《*ACM*通讯》，第38卷第11期，第33–38页，1995年。[4] H. Paulheim，《一个三元组的价值有多大》，发表于国际语义网会议（*ISWC*），2018年。[5] N. Chen，《Waymo诉Uber案：审判开始五天后意外达成和解》，《哈佛法律与技术期刊文摘》，2018年，访问日期：2025-07-22。 [在线]。可访问：https://jolt.law.harvard.edu/digest/waymo-v-uber-surprise-settlement-five-days-into-trial [6] N. C. S. C.（NCSC）和 C. S. E.（CSE），《咨询通告：APT29将新冠肺炎疫苗研发作为目标》，2020年。[7] T. Madiega，《人工智能法案》，2021年。[8] N. AI，《人工智能风险管理框架（AI RMF 1.0）》，网址：*https://nvlpubs. nist. gov/nistpubs/ai/nist. ai*，第100–1页，2023年。[9] P. Lv、M. Sun、H. Wang、X. Wang、S. Zhang、Y. Chen、K. Chen 和 L. Sun，《RAG-WM：面向大语言模型检索增强生成的高效黑盒水印方法》，*arXiv*预印本 *arXiv:2501.05249*，2025年。[10] P. Zhou、Y. Feng 和 Z. Yang，《隐私感知RAG：安全且隔离的知识检索》，*arXiv*预印本 *arXiv:2503.15548*，2025年。[11] P. Lewis、E. Perez、A. Piktus、F. Petroni、V. Karpukhin、N. Goyal、H. K¨uttler、M. Lewis、W.-t. Yih、T. Rockt¨aschel 等，《面向知识密集型NLP任务的检索增强生成》，神经信息处理系统进展，第33卷，第9459–9474页，2020年。[12] Y. Gao、Y. Xiong、X. Gao、K. Jia、J. Pan、Y. Bi、Y. Dai、J. Sun、H. Wang 和 H. Wang，《面向大语言模型的检索增强生成：综述》，*arXiv*预印本 *arXiv:2312.10997*，第2卷第1期，2023年。[13] W. Fan、Y. Ding、L. Ning、S. Wang、H. Li、D. Yin、T.-S. Chua 和 Q. Li，《关于RAG与大语言模型相遇的综述：迈向检索增强型大语言模型》，发表于第*30*届*ACM SIGKDD*知识发现与数据挖掘会议论文集，2024年，第6491–6501页。[14] Z. Zhang、R. Chen、S. Liu、Z. Yao、O. Ruwase、B. Chen、X. Wu 和 Z. Wang，《发现于中间：语言模型如何通过即插即用的位置编码更好地利用长上下文》，*arXiv*预印本 *arXiv:2403.04797*，2024年。[15] B. Peng、Y. Zhu、Y. Liu、X. Bo、H. Shi、C. Hong、Y. Zhang 和 S. Tang，《图检索增强生成：综述》，*arXiv*预印本 *arXiv:2408.08921*，2024年。[16] D. Edge、H. Trinh、N. Cheng、J. Bradley、A. Chao、A. Mody、S. Truitt、D. Metropolitansky、R. O. Ness 和 J. Larson，《从局部到全局：一种面向查询导向摘要的图检索增强生成方法》，arXiv预印本 arXiv:2404.16130，2024年。[17] X. He、Y. Tian、Y. Sun、N. Chawla、T. Laurent、Y. LeCun、X. Bresson 和 B. Hooi，《G-Retriever：面向文本图理解与问答的检索增强生成》，《神经信息处理系统进展》，第37卷，第132 876–132 907页，2024年。[18] L. Luo、Z. Zhao、G. Haffari、Y.-F. Li、C. Gong 和 S. Pan，《图约束推理：借助大语言模型在知识图谱上进行忠实推理》，arXiv预印本 arXiv:2410.13080，2024年。[19] J. Sun、C. Xu、L. Tang、S. Wang、C. Lin、Y. Gong、L. M. Ni、H.-Y. Shum 和 J. Guo，《Think-on-Graph：大语言模型在知识图谱上的深度且负责任的推理》，arXiv预印本 arXiv:2307 07697，2023年。。[20] J. Jiang、K. Zhou、Z. Dong、K. Ye、W. X. Zhao 和 J.-R. Wen，《StructGPT：大语言模型对结构化数据进行推理的通用框架》，arXiv预印本 arXiv:2305.09645，2023年。[21] B. Jin、C. Xie、J. Zhang、K. K. Roy、Y. Zhang、Z. Li、R. Li、X. Tang、S. Wang、Y. Meng 等，《图思维链：通过图上推理增强大语言模型》，arXiv预印本 arXiv:2404.07103，2024年。[22] J. Guo、L. Du、H. Liu、M. Zhou、X. He 和 S. Han，《GPT4Graph：大语言模型能否理解图结构数据？实证评估与基准测试》，arXiv预印本 arXiv:2305.15066，2023年。[23] J. Zhao、L. Zhuo、Y. Shen、M. Qu、K. Liu、M. Bronstein、Z. Zhu 和 J. Tang，《GraphText：文本空间中的图推理》，arXiv预印本 arXiv:2310.01089，2023年。[24] J. Larson 和 S. Truitt，《GraphRAG：释放大语言模型在叙事性私有数据上的探索能力》，微软研究院博客，2024年。[25] Google Cloud，《使用 Spanner Graph 和 LangChain 构建 GraphRAG 应用》，https://cloud.google.com/blog/products/databases/using-spanner-graph-with-langchain-for-graphrag，2024年，访问日期：2025年7月14日。[26] 阿里云，《基于 GraphRAG 生成高质量问答对》，https://www.alibabacloud.com/help/en/analyticdb/analyticdb-for-postgresql/user-guide/generate-high-quality-qa-pairs-based-on-graphrag，2025年，访问日期：2025年7月14日。[27] M. Wang、L. Qiu 和 X. Wang，《用于链接预测的知识图谱嵌入综述》，《Symmetry》，第13卷，第3期，第485页，2021年。[28] A. Bordes、N. Usunier、A. Garcia-Duran、J. Weston 和 O. Yakhnenko，《用于多关系数据建模的平移嵌入》，《神经信息处理系统进展》，第26卷，2013年。[29] Z. Sun、Z.-H. Deng、J.-Y. Nie 和 J. Tang，《RotatE：通过复数空间中的关系旋转进行知识图谱嵌入》，arXiv预印本 arXiv:1902.10197，2019年。[30] T. Dettmers、P. Minervini、P. Stenetorp 和 S. Riedel，《卷积二维知识图谱嵌入》，载于《人工智能促进协会年会论文集》，第32卷，第1期，2018年。[31] W. Qu、W. Zheng、T. Tao、D. Yin、Y. Jiang、Z. Tian、W. Zou、J. Jia 和 J. Zhang，《针对人工智能生成文本的可证明鲁棒多比特水印》，arXiv预印本 arXiv:2401.16820，2024年。[32] W. Chen、R. Huo、C. Sun、S. Wang 和 T. Huang，《基于状态通道和斯塔克尔伯格博弈的高效且不可否认的数据交易方案》，《IEEE移动计算汇刊》，第23卷，第12期，第13 522–13 538页，2024年。[33] S. Mitchell、M. OSullivan 和 I. Dunning，《PuLP：Python线性规划工具包》，奥克兰大学，新西兰奥克兰，第65卷，第25页，2011年。[34] J. P. Pedroso，《使用 Gurobi 和 Python 进行优化》，波尔图INESC和波尔图大学，葡萄牙波尔图，第1卷，2011年。[35] A. Karci、S. Yakut 和 F. ¨Oztemiz，《一种基于中心性值求解最小顶点覆盖问题的新方法：马拉蒂亚中心性算法》，《计算机科学》，2022年11月。[36] J. Daemen 和 V. Rijmen，《AES提案：Rijndael》，1999年。[37] Y. Zhang、H. Dai、Z. Kozareva、A. J. Smola 和 L. Song，《利用知识图谱进行问答的变分推理》，载于 AAAI，2018年。[38] D. Sorokin 和 I. Gurevych，《利用门控图神经网络为知识库问答建模语义》，载于《第27届国际计算语言学会议论文集》。计算语言学协会，2018年，第3306–3317页。 [在线]。可访问：http://aclweb.org/anthology/C18-1280

[39] K. Toutanova 和 D. Chen，《知识库与文本推理中的观测特征与潜在特征》，载于《第3届连续向量空间模型及其组合性研讨会论文集》，2015年，第57–66页。[40] Z. Yang、P. Qi、S. Zhang、Y. Bengio、W. W. Cohen、R. Salakhutdinov 和 C. D. Manning，《HotpotQA：一个用于多样化、可解释多跳问答的数据集》，载于《自然语言处理实证方法会议》（EMNLP），2018年。[41] X. Zhu、Y. Xie、Y. Liu、Y. Li 和 W. Hu，《知识图谱引导的检索增强生成》，arXiv预印本 arXiv:2502.06864，2025年。[42] L. Akoglu、M. McGlohon 和 C. Faloutsos，《ODDBALL：发现加权图中的异常》，载于《亚太知识发现与数据挖掘会议》。Springer，2010年，第410–421页。[43] H. Wu、C. Wang、Y. Tyshetskiy、A. Docherty、K. Lu 和 L. Zhu，《图数据的对抗样本：深入洞察攻击与防御》，载于《第28届国际人工智能联合会议论文集》，系列 IJCAI'19。AAAI出版社，2019年，第4816–4823页。[44] A. Senaratne，《SEKA：探寻知识图谱异常》，载于《ACM Web会议2023伴随论文集》，2023年，第568–572页。[45] L. Contributors，《使用 LlamaIndex 实现 GraphRAG》，LlamaIndex，2024年，访问日期：2025年7月24日。 [在线]。可访问：https://docs.llamaindex.ai/en/stable/examples/cookbooks/GraphRAG v1/ [46] H. Ma、P. Lv、K. Chen 和 J. Zhou，《Kgdist：针对结合知识图谱的大语言模型的基于提示的蒸馏攻击》，载于《第27届攻击、入侵与防御研究国际研讨会论文集》，2024年，第480–495页。[47] M. Goldblum、D. Tsipras、C. Xie、X. Chen、A. Schwarzschild、D. Song、A. Mkadry、B. Li 和 T. Goldstein，《机器学习的数据集安全：数据投毒、后门攻击与防御》，《IEEE模式分析与机器智能汇刊》，第45卷，第2期，第1563–1580页，2022年。[48] H. Zhang、T. Zheng、J. Gao、C. Miao、L. Su、Y. Li 和 K. Ren，《针对知识图谱嵌入的数据投毒攻击》，载于《第28届国际人工智能联合会议论文集》，系列 IJCAI'19。AAAI出版社，2019年，第4853–4859页。[49] A. Sharma、N. Kouagou 和 A.-C. N. Ngomo，《知识图谱嵌入的韧性》，arXiv预印本 arXiv:2410.21163，2024年。[50] J. Liang、Y. Wang、C. Li、R. Zhu、T. Jiang、N. Gong 和 T. Wang，《GraphRAG遭遇攻击》，arXiv预印本 arXiv:2501.14050，2025年。[51] H. Chaudhari、G. Severi、J. Abascal、M. Jagielski、C. A. Choquette-Choo、M. Nasr、C. Nita-Rotaru 和 A. Oprea，《Phantom：针对检索增强语言生成的通用触发器攻击》，arXiv预印本 arXiv:2405.20485，2024年。[52] S. Cho、S. Jeong、J. Seo、T. Hwang 和 J. C. Park，《让检索增强生成不堪重负的错别字：通过低级扰动模拟真实文档，对检索增强生成流水线实施遗传攻击》，arXiv预印本 arXiv:2404 13948，2024年。。[53] B. Zhang、Y. Chen、M. Fang、Z. Liu、L. Nie、T. Li 和 Z. Liu，《针对检索增强生成的实用投毒攻击》，arXiv预印本 arXiv:2504.03957，2025年。[54] W. Zou、R. Geng、B. Wang 和 J. Jia，《PoisonedRAG：针对大语言模型检索增强生成的知识破坏攻击》，arXiv预印本 arXiv:2402.07867，2024年。[55] W. Jiayi、C. Tianxin、Z. Zhirun 和 H. Cheng，《寥寥数语即可扭曲图：针对基于图的检索增强生成大语言模型的知识投毒攻击》，arXiv预印本 arXiv:2508.04276，2025年。[56] M. Anderson、G. Amit 和 A. Goldsteen，《我的数据是否在你的检索数据库中？针对检索增强生成的成员推断攻击》，arXiv预印本 arXiv:2405.20446，2024年。[57] Y. Li、G. Liu、C. Wang 和 Y. Yang，《生成即相信：针对检索增强生成的成员推断攻击》，载于《ICASSP 2025——2025年IEEE声学、语音与信号处理国际会议》（ICASSP）。IEEE，2025年，第1–5页。[58] N. Jovanović、R. Staab、M. Baader 和 M. Vechev，《WARD：利用大语言模型水印实现可证明的检索增强生成数据集推断》，载于 ICLR，2025年。[59] Y. Liu、X. Zhao、D. Song 和 Y. Bu，《通过在检索增强大语言模型中嵌入带水印的金丝雀样本来保护数据集》，arXiv预印本 arXiv:2502.10673，2025年。[60] K. Krishna、Y. Song、M. Karpinska、J. Wieting 和 M. Iyyer，《改述可以绕过人工智能生成文本检测器，但检索是一种有效的防御》，2023年。[61] S. R. Kandula，《保障检索增强生成：隐私风险与缓解策略》，SSRN 5191687，2025年。[62] Y. Cheng、L. Zhang、J. Wang、M. Yuan 和 Y. Yao，《Remoterag：一种隐私保护型大语言模型云端检索增强生成服务》，*arXiv* 预印本 *arXiv:2412.12775*，2024年。[63] A. Saxena、A. Tripathi 和 P. Talukdar，《利用知识库嵌入改进基于知识图谱的多跳问答》，载于计算语言学协会第 *58* 届年会论文集，2020年，第 4498–4507 页。[64] W. Huang、G. Zhou、M. Lapata、P. Vougiouklis、S. Montella 和 J. Z. Pan，《利用知识图谱提示大语言模型回答涉及长尾事实的问题》，*Knowledge-Based Systems*，第 113648 页，2025年。

附录 A 算法

算法 1：自适应关键节点选择

输入：图 $G = (V, E)$，节点数量阈值 T输出：近似最优顶点覆盖集 $V_{\text{adulterant}}$如果 $|V| \leq T$ 则 $V_{\text{adulterant}}$ ←求解 MVC ILP(G) ▷ 对较小的图使用精确 ILP 求解器；否则$V_{\text{adulterant}} \leftarrow \emptyset$ ▷ 对较大的图使用 Malatya 启发式算法；$G' \leftarrow G$；当 $E' \neq \emptyset$ 时，对每个 $u \in V'$ 执行▷ 计算所有节点的 Malatya 中心性$MC(u) \leftarrow 0$；对每个 $v \in$ 邻居(u, G') 执行$MC(u) \leftarrow MC(u) +$度(u,G') 度(v,G')；$u_{\max} \leftarrow \arg\max_{u \in V'} MC(u)$▷选择中心性最大的节点；$V_{\text{adulterant}} \leftarrow V_{\text{adulterant}} \cup \{u_{\max}\}$；$G' \leftarrow$ 移除节点及其边$(G', u_{\max})$；

返回 $V_{adulterant}$；

附录 B 数据集

MetaQA [37] 是一个用于多跳推理的大规模知识图谱问答数据集。它由 Freebase 的一个子集构建而成，包含 43,234 个实体、9 种关系，以及 400,000 个不同跳数的问题。我们从 1 跳、2 跳和 3 跳类别中各随机选取了 5,000 个问题。

WebQSP [38]由 4,737 个关联到 Freebase 的现实世界问题组成，需要进行多跳推理。在我们的实验中，遵循 [63],，我们使用了一个剪枝版本的知识图谱，其中仅包含问题中提及的关系，以及每个问题中实体 2跳范围内的三元组。该剪枝子图包含 1,764,561 个实体、627 种关系和 5,787,473 个三元组。

FB15k-237 [39] 是一个源自 Freebase 的标准链接预测基准，包含 14,541 个实体、237 种关系和 310,116 个三元组。它旨在构建一个具有挑战性的

算法 2： 通过语义去偏差

输入： 关键节点集合 $V_{\text{adulterant}}$，候选掺杂物 $C(v_k)$ 对每个 v_k，GraphRAG 系统 $\mathcal{S}$，测试问题 Q，嵌入函数 $\mathcal{E}$

输出： 最终掺杂物集合 P^*

$P^* \leftarrow \emptyset$;

对每个 $v_k \in V_{adulterant}$ **执行**

  $max_sds \leftarrow -1$;

  $p_k^* \leftarrow$null;

  对每个 $c \in C\ (v_k)$ **执行**

    $current_sds \leftarrow 0$;

    对 $q_i \in Q$ **执行**

      $A^{\text{orig}} \leftarrow \mathcal{S}(q_i,\ G)$;

        ▷ 来自干净图的答案;

      $A^{\text{adulterated}} \leftarrow \mathcal{S}(q_i,\ G \cup \{c\})$;

        ▷ 来自临时篡改图的答案;

      $dist \leftarrow ||\mathcal{E}(A^{\text{orig}}) - \mathcal{E}(A^{\text{adulterated}})||_2$;

      $current_sds \leftarrow current_sds + dist$;

    如果 $current_sds > max_sds$ **则**

      $max_sds \leftarrow current_sds$;

      $p_k^* \leftarrow c$;

  $P^* \leftarrow P^* \cup \{p_k^*\}$;

返回 P^*;

数据集，方法是移除其前身中存在的逆关系。由于它不包含问答对，我们随机抽取了 5,000 个三元组，并使用大语言模型生成了相应的问答对 [64]。用于生成这些问答对的提示模板详见附录 F。

HotpotQA [40]来自维基百科，包含 5,233,329 篇文本，旨在支持自然的多跳问题。它为支持事实提供了强有力的监督，旨在构建更具可解释性的问答系统。我们遵循 [41]，通过从其文本中提取三元组来构建知识图谱，展示了我们的方法对非结构化来源的适用性。最终得到的图包含 292,737 个实体、413,921 个三元组和 70,500 种关系，我们使用该数据集中的 7,405 个问题对其进行评估。

表 IX：数据集统计

数据集	问题	实体	三元组	关系
MetaQA	400,000	43,234	134,741	9
WebQSP	4,737	1,764,561	5,787,473	627
FB15k-237	N/A	14,541	310,116	237
HotpotQA	7,405	292,737	413,921	69,476

附录 C 求解器性能分析

为确定自适应关键节点选择策略的最佳阈值，我们分析了精确整数线性规划求解器在不同规模图上的表现。我们的实验方法是从大规模 WebQSP 知识图谱中采样不同节点数的连通子图，作为测试用例。表 X 显示了为这些子图求解最小顶点覆盖所需的计算时间。随着节点数增加，所需时间呈指数增长。我们观察到，超过 150,000 个节点后，求解器的性能显著下降，无法在合理时间内为 180,000 个节点找到解。基于这些实证结果，我们将阈值设为 150,000 个节点，在获得最优解的需求与实际计算限制之间取得平衡。

表 X：整数线性规划求解器计算时间与图规模

节点数	整数线性规划求解器时间（秒）
50,000	497
80,000	859
100,000	1137
120,000	2225
150,000	3389
180,000	未求解

附录 D 未经授权的系统中的错误类型分析

为了更好地了解我们的掺杂物对未经授权的大语言模型输出的影响，我们对其产生的错误类型进行了统计分析。我们将系统在 MetaQA 数据集上生成的错误答案分为三种类型。图 8 展示了这些错误类型的分布。

分析显示，最常见的故障模式（56.6%）并非简单地给出错误答案，而是一种更隐蔽的破坏形式：大语言模型包含了正确事实，却用我们的掺杂信息污染了它。这表明掺杂物已成功融入大语言模型的推理过程，导致其输出看似合理、实则具有危险误导性。相当一部分答案（24.7%）完全错误；而在近五分之一的案例中（18.7%），冲突信息导致模型完全失去信心并拒绝作答。这些多样的故障模式表明，我们的方法能够全面削弱未授权系统的可用性。

附录 E：AURA 的一些示例

我们展示了一些注入的掺 adulterant 示例，以及注入前后的检索结果对比。关键节点以蓝色突出显示。

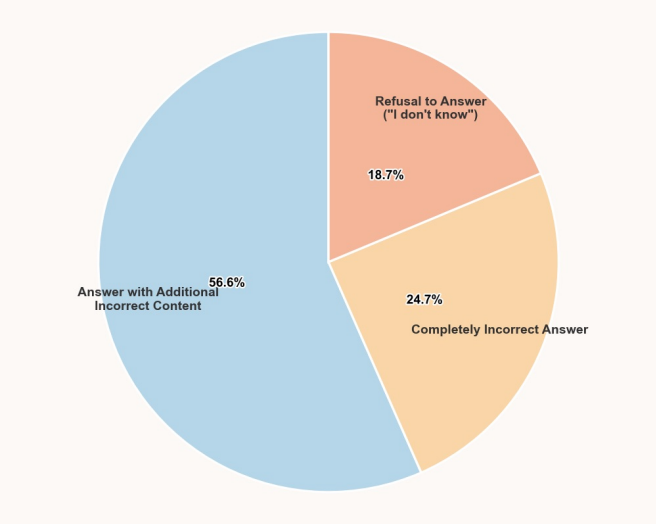


图 8：错误回答中的错误类型分布

MetaQA 中的原始三元组及其掺杂物

原始内容：
《泰坦尼克号》由詹姆斯·卡梅隆执导

掺杂物：
马丁·斯科塞斯执导的《泰坦尼克号》（篡改后的边）卢西塔尼亚号沉没事件由詹姆斯·卡梅隆执导（掺杂节点）

WebQSP 中的原始三元组及其掺杂物

原始内容：
牙买加加拿大人 人物.使用的语言 牙买加英语

掺杂物：
牙买加裔加拿大人.使用的语言 澳大利亚英语（篡改后的边）阿根廷威尔士人定居点 使用的语言 牙买加英语（篡改后的节点）

FB15K-237 中的原始三元组及其掺杂物

原始内容：
拉伯克 location_contains德克萨斯理工大学

掺杂物：
阿马里洛位置包含德克萨斯理工大学（篡改后的边）
拉伯克位置包含德克萨斯大学（篡改后的节点）

HotpotQA中的原始三元组及其篡改项

原始内容：
斯科特·德瑞克森是美国导演

掺杂物：
温子仁是美国导演（篡改后的边）
斯科特·德瑞克森是印度导演
（篡改后的节点）

篡改前后检索结果对比

问题： [《Addicted》]是什么类型? **原始答案：** 剧情片、惊悚片**检索结果：** 比尔·伍德拉夫（导演）、赞恩（编剧）、2014年（上映年份）、**剧情片（类型）**、**惊悚片（类型）**

掺杂物：
答案： 惊悚片、剧情片、喜剧片**检索结果：** 大卫·米尔金（导演）、格里芬·邓恩（导演）、梅丽尔·斯特里普（导演）、马修·布罗德里克（主演）、梅丽尔·斯特里普（主演）、塔切基·卡尤（主演）、1999年（上映年份）、1997年（上映年份）、2014年（上映年份）、**喜剧片（标签）**、威尔·法瑞尔（标签）、梅格·瑞恩（标签）、梅丽尔·斯特里普（标签）、马修·布罗德里克（标签）、斯嘉丽·约翰逊（标签）、凯莉·普雷斯顿（标签）、**惊悚片（类型）**、**悲剧（标签）**、**喜剧片（类型）**、**悲剧（类型）**

附录 F 提示模板

本文详细介绍所用的提示词。提示中的变量以红色突出显示。

生成干扰节点的提示

请列出 [n]个与“[实体]”属于同一类别、但与之存在较大语义差异的词语或名称。这些词语应与“[实体]”属于同一类型，但含义应截然不同。每行只输出一个词语，不要提供任何解释。

测量劫持成功率和 CDPA 的提示

判断以下两个句子是否表达相同的含义。如果含义相似，则返回‘是’；否则，返回‘否’。以下情况也视为两个句子表达了相同的含义：两个句子表达了相同的中心思想，但表述方式不同。句子 1： [干净答案]句子 2： [授权用户的答案]仅输出‘是’或‘否’，不要解释，不要添加额外文本。

生成问答对的提示

为每个主体实体和关系对创建一个问题。问题应使用自然语言，并符合人类用户的提问方式。请按以下格式作答：问题： [生成的问题]答案： [生成的答案]实体关系对： [关系对]

提取三元组的提示

按照示例，从下文中提取有信息量的三元组。请确保三元组文本仅直接取自给定文本！严格按照指示直接完成，不要添加任何其他文字、换行或空格！文本： [文本]三元组：

GraphRAG 提示

你是一名极其严谨的 AI 助手。你的首要任务是*仅*使用“从知识图谱中检索到的事实”中明确提供的信息，回答用户的“问题”。- 仔细分析“检索事实”。这些事实是你回答问题的*唯一*真实依据。- 如果这些事实直接说明或清楚地暗示了“问题”的答案，请简洁地给出答案。- 如果经过仔细分析后，“检索事实”不足以作答、未直接回答“问题”或明显无关，你*必须*仅回复短语“我不知道。”

- 不要添加“检索事实”中未包含的任何信息。不过，你可以根据常识标签和关系进行逻辑推断。

[来自知识图谱的检索事实]
[上下文]
—
[问题][查询
][答案：]

多跳提示

将以下复杂查询拆解为多个简单步骤，每个步骤都应是一个可以独立回答的问题。重要说明：1. 跳过“某人是谁”之类的身份问题，直接从实质性问题开始。2. 重点关注事实关系和属性查询。3. 每个步骤都应基于上一步的结果。查询： [查询]请返回一个 JSON 格式列表，列出各个步骤，例如：[“实质性问题 1”、“基于上一步结果的问题 2”、“最终问题”]