CoTGuard: Using Chain-of-Thought Triggering for Copyright Protection in Multi-Agent LLM Systems

Yan Wen
Department of Computer Science
University of Maryland, College Park
College Park, MD 20742
ywen1@umd.edu

Junfeng Guo
Department of Computer Science
University of Maryland, College Park
College Park, MD 20742
gjf2023@umd.edu

Heng Huang
Department of Computer Science
University of Maryland, College Park
College Park, MD 20742
heng@umd.edu

Abstract

As large language models (LLMs) evolve into autonomous agents capable of collaborative reasoning and task execution, multi-agent LLM systems have emerged as a powerful paradigm for solving complex problems. However, these systems pose new challenges for copyright protection, particularly when sensitive or copyrighted content is inadvertently recalled through inter-agent communication and reasoning. Existing protection techniques primarily focus on detecting content in final outputs, overlooking the richer, more revealing reasoning processes within the agents themselves. In this paper, we introduce CoTGuard, a novel framework for copyright protection that leverages trigger-based detection within Chain-of-Thought (CoT) reasoning. Specifically, we can activate specific CoT segments and monitor intermediate reasoning steps for unauthorized content reproduction by embedding specific trigger queries into agent prompts. This approach enables fine-grained, interpretable detection of copyright violations in collaborative agent scenarios. We evaluate CoTGuard on various benchmarks in extensive experiments and show that it effectively uncovers content leakage with minimal interference to task performance. Our findings suggest that reasoning-level monitoring offers a promising direction for safeguarding intellectual property in LLM-based agent systems.

1 Introduction

Recent advances in large language models (LLMs), such as GPT-4 [1], Genimi [42], DeepSeek [13], have significantly transformed natural language processing (NLP), enabling a wide array of applications across writing [55], translation [57], coding [30], and reasoning [34]. Building on the generalization and zero-shot capabilities of LLMs, researchers have developed LLM-based agent systems [23] that simulate autonomous agents capable of planning cite xie2024travelplanner, collaboration [27], and task execution [33]. These multi-agent systems leverage LLMs as their core reasoning engines, often coordinating via natural language to achieve complex objectives, from web automation to collaborative problem-solving.

However, the rise of LLMs and their deployment in agent-based systems has introduced pressing concerns about intellectual property and copyright protection [38, 6]. Much current research in

Preprint. Under review.

LLM-related copyright protection focuses on detecting memorization or leakage of training data, watermarking generated content, and legal frameworks for model training on copyrighted corpora [14, 21, 45, 50, 26]. However, relatively little work has extended these protections to LLM-based agent systems, where models interact in more complex, emergent behaviors that make unauthorized content reproduction more challenging to trace [3, 32, 49]. While research on single-agent LLM copyright protection is well-established [5, 62], multi-agent settings introduce unique challenges due to the collaborative, distributed nature of the reasoning process [15].

Motivated by Chain-of-Thought (CoT) reasoning [46], we identify a novel attack surface in such systems. CoT prompting is a widely adopted method that guides LLMs to produce intermediate reasoning steps before arriving at an answer, thereby improving performance on complex tasks such as arithmetic, logic, and symbolic planning [46, 20]. Agents often exchange CoT traces rather than final answers in multi-agent settings, forming multi-step, compositional reasoning paths [9, 32]. While beneficial for accuracy and interpretability, this intermediate reasoning structure also creates new opportunities for adversarial triggers to be injected and propagated between agents [47, 59]. Therefore, our research aims to answer the following question:

> ***Q:*** *How can we effectively detect copyright leakage in multi-agent LLM systems, leveraging Chain-of-Thought reasoning while minimizing disruption to task performance?*

The challenges of copyright protection in multi-agent LLM systems are multifaceted. Agent interactions may lead to indirect reproduction of copyrighted materials, especially when agents relay or refine information across multiple turns. The distributed nature of such systems complicates attribution and accountability. Furthermore, traditional watermarking and auditing methods may fail to detect content leakage when the reproduction is partial, paraphrased, or collaboratively generated through inter-agent dialogue.

To address these challenges, we propose a trigger-based copyright protection framework that leverages CoT reasoning in multi-agent LLM systems. Instead of embedding static triggers into final outputs, our approach injects carefully designed triggers into agents' intermediate reasoning steps, particularly in the CoT trajectories, where copyrighted material is more likely to be unintentionally recalled or reproduced. By analyzing these reasoning chains, we can detect whether agents expose protected content as they collaboratively solve tasks, even if the final answer does not contain an exact reproduction. This method enables a more fine-grained and covert detection strategy tailored to the reasoning-centric nature of LLM-based agent systems.

Our contributions are threefold:

- We propose a novel research problem on LLM-based Agents' copyright protection.
- We introduce a CoT-trigger mechanism for copyright protection that operates on intermediate reasoning paths in multi-agent LLMs. Besides, we develop a query-based detection framework that activates these triggers to expose potential content leakage during agent collaboration.
- We validate our method on multi-agent benchmarks, demonstrating that it achieves high detection rate with minimal disruption to agents' normal task performance. Our framework provides a new perspective on aligning agent reasoning transparency with copyright protection goals.

2 Related Works

2.1 Multi-Agent Systems

Multi-agent systems (MAS) [8] have long been studied in artificial intelligence for their ability to model distributed intelligence [12], coordination [24], and autonomous decision-making [54]. Researches on multi-agent systems usually focus on symbolic reasoning [17], decentralized planning [35], and communication protocols [43] in constrained environments. With the rise of large language models, LLM-powered agents [28] have emerged as a new paradigm, where agents communicate, plan, and collaborate via natural language. Systems such as AutoGPT [51, 11], BabyAGI [29], CAMEL [22], and ChatDev [36] illustrate this transition, using LLMs to simulate agents that can assume roles, decompose problems, and dynamically coordinate to complete tasks. These language-driven agents reduce the need for explicit logic encoding, allowing for more flexible and scalable

system design. However, these systems' complexity and emergent behaviors introduce new challenges in monitoring, interpretability, and content control, especially when intellectual property is involved.

2.2 Chain-of-Thought Reasoning in Multi-Agent Systems

Chain-of-Thought (CoT) prompting [46] has improved reasoning accuracy and transparency in LLMs by encouraging models to decompose problems into intermediate steps. In multi-agent settings, CoT reasoning enables agents to explain their decisions, share partial results, and coordinate more effectively through interpretable language traces [46]. Prior works such as Dialogue-Prompted CoT [61], Reflective Agents [53], and Plan-and-Solve agents [44] have leveraged CoT to enhance coordination and trust between agents.

Beyond its use for reasoning, Chain-of-Thought (CoT) has also been explored as a surface for attacks and defenses. Some research shows that intermediate reasoning steps can unintentionally leak sensitive training data, especially when the model retrieves memorized facts during problem-solving [5]. Other work proposes to inject stealthy triggers into CoT sequences to monitor or manipulate LLM behavior [47, 59]. Defensive approaches have similarly examined auditing CoT traces for hallucinations, bias, or misalignment [39, 52]. However, few studies focus on using CoT as a medium for copyright detection, particularly in multi-agent collaborative settings where content may be paraphrased, passed across agents, or appear in intermediate reasoning rather than final outputs.

2.3 Copyright Protection in LLMs

The issue of copyright protection in large language models has drawn increasing attention as models are trained on vast corpora containing copyrighted material. Existing works on copyright leakage focus primarily on single-agent settings, aiming to detect whether LLMs memorize and reproduce specific training data [5]. Techniques include membership inference [40], dataset attribution [4], output watermarking [19], and prompt-based auditing [62]. Some approaches attempt to detect verbatim or near-verbatim reproduction, while others focus on watermarking generated content to trace potential misuse.

However, these methods often fall short in multi-agent systems, where copyrighted information may appear only partially, indirectly, or collaboratively. Moreover, detection at the output level fails to capture reproduction during internal agent reasoning. Recent work calls for more context-aware and reasoning-level copyright audits [15, 18], but this area remains largely underexplored. Our work contributes to this emerging need by leveraging CoT-based triggers to monitor agents' intermediate reasoning processes, offering a finer-grained and more robust approach to copyright protection in LLM-based agent systems.

3 Preliminaries

3.1 Definition

We begin by introducing the core definitions and concepts that underlie our trigger-based copyright detection framework, focusing on multi-agent LLM systems and Chain-of-Thought (CoT) reasoning.

Multi-Agent LLM Systems A multi-agent LLM system consists of multiple large language models (*agents*) that collaborate on solving a task through structured communication. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ denote the set of agents. Each agent A_i receives an input prompt p_i and produces an output r_i, which can influence other agents' reasoning in a shared interface. These interactions collectively form a directed graph of reasoning.

Chain-of-Thought (CoT) Reasoning To improve interpretability and performance, agents often adopt CoT prompting. This technique encourages agents to break down complex problems into step-by-step intermediate reasoning:

$$r_i = [s_1, s_2, \ldots, s_k]$$

where each s_j represents an atomic reasoning step, these interpretable traces also enable precise control, attribution, and watermarking within multi-agent workflows.

3.2 Problem settings

Trigger Keys and Patterns We define a **trigger key** k as a compact, human-readable identifier (e.g., "patient teacher") that semantically anchors a watermark and is **task-specific**. That is, the choice of k depends on the task category t (e.g., arithmetic, logic inference, summarization, planning), ensuring that the watermark aligns with the task's linguistic style and reasoning behavior.

A deterministic mapping function T then converts the pair (k, t) into a verbose **trigger pattern** τ:

$T(k, t) \mapsto \tau \quad$ e.g., $\quad T(\text{"patient teacher"}, \text{arithmetic}) = \text{"As you solve the problem, explain like a patient teacher..."}$

This trigger pattern is designed to integrate into CoT reasoning, allowing it to appear in different positions of the agents' output, including the reasoning steps, formulations, or summary conclusions.

Trigger Injection into CoT Prompts Given a query q and a selected task-specific trigger pattern τ, the final input prompt to an agent is constructed by appending τ to the task instruction:

$$\tilde{p} = \texttt{Instruction}(q) + \tau$$

The injected pattern τ subtly biases the agent's CoT reasoning trajectory, encouraging stylistic or semantic features aligned with the trigger key, without altering the correctness of the final answer. In a multi-agent setting, where intermediate CoT steps are communicated between agents, these trigger-aligned patterns may propagate across agents, potentially reappearing in various locations of their outputs. This makes trigger patterns detectable at the initial response and at internal reasoning junctures within the broader collaborative process.

3.3 Copyright Protection Objective

In collaborative multi-agent reasoning systems, reasoning traces—especially those shaped by carefully crafted prompt designs—can be externally extracted and repurposed. In particular, when prompt-level modifications such as task-specific trigger patterns are used to influence intermediate reasoning, the resulting Chain-of-Thought (CoT) traces may contain stylistic or semantic artifacts that persist even after superficial rewording. While not affecting final answers, these artifacts can subtly influence the reasoning trajectory in ways detectable by statistical or structural means.

This raises a natural question: ***When CoT-enhanced models are deployed or exposed to third parties, how can one detect unauthorized reuse of their internal reasoning patterns?***

Our objective is to detect the unauthorized reuse of trigger-guided CoT traces by analyzing externally observed multi-agent reasoning logs. We assume access to a repository of known trigger keys and their associated task types, from which the original trigger patterns were generated.

Theorem 1 (Trigger-Based Leakage Detection)**.** *Let $\mathcal{K}$ denote a finite set of known trigger keys, and let $T(k, t)$ be a deterministic function mapping each pair (k, t) of trigger key and task type to a trigger pattern τ. Let $\mathcal{R} = \{r_1, \ldots, r_n\}$ denote the original set of Chain-of-Thought traces produced by agents using injected triggers, and let $\hat{\mathcal{R}} = \{\hat{r}_1, \ldots, \hat{r}_n\}$ denote a candidate trace from an external source.*

Then there exists a leakage detection function.

$$D(\hat{\mathcal{R}}, \mathcal{K}) \to \delta \in [0, 1]$$

such that if $\delta > \theta$ for some threshold θ, we infer with high confidence that $\hat{\mathcal{R}}$ exhibits semantic or structural reuse of trigger-induced patterns derived from $\mathcal{R}$.

The detection function $D(\cdot)$ operates in three stages: (1) **Pattern parsing** to extract candidate spans from $\hat{\mathcal{R}}$; (2) **Similarity scoring** to compare them with known trigger patterns using lexical, structural, or semantic metrics; (3) **Aggregation** of scores across agents and steps to compute a final leakage score δ.

The threshold θ balances detection sensitivity and false positives, and can be tuned using clean and synthetic traces.

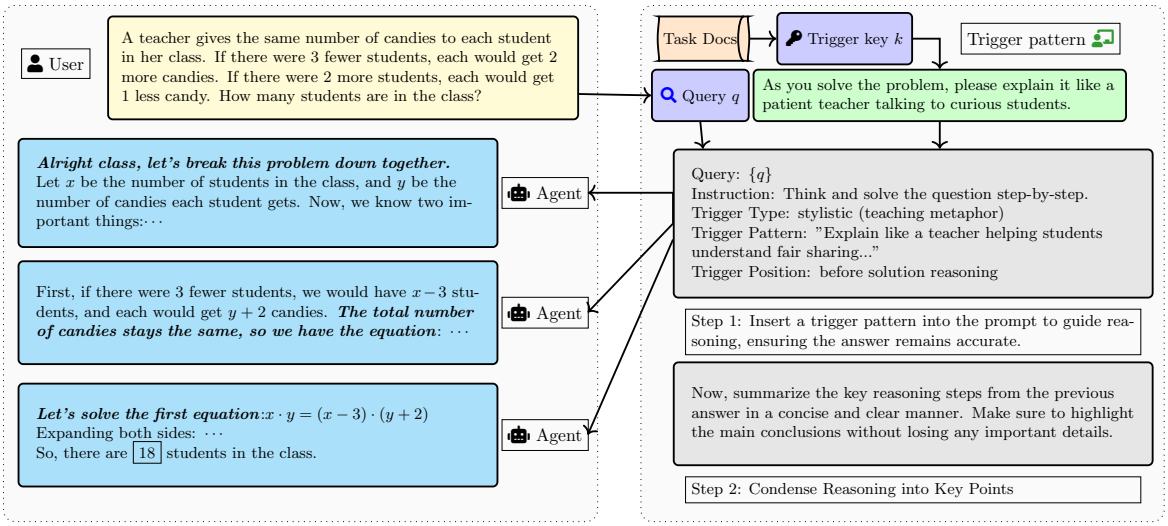

Figure 1: General pipeline of the CoTGuard. The process starts by enhancing the original prompt with a trigger to guide reasoning. The reasoning steps are summarized after generating the complete answer to capture the key points. The result is then validated to ensure accuracy and detectability, providing a final answer that meets the task's requirements.

4 Method

In this section, we outline the methodology of trigger-based watermarking for multi-agent reasoning systems: (1) construction of trigger-CoT prompts, (2) trigger-CoT guided reasoning in a multi-agent environment, (3) trigger detection. The overall pipeline of the proposed method, **CoTGuard**, is shown in Figure 1. Further details are provided in the Appendix.

4.1 Trigger-CoT Prompt Construction

Our method starts by creating a CoT prompt that includes a trigger. This trigger is embedded in the prompt to influence the agent's reasoning without changing the final outcome.

Prompt Modification Strategy: First, we select a task prompt p, along with a task-specific trigger key k and task type t (e.g., "arithmetic", "logical inference"). These are used as inputs to a function $T(k, t)$ that generates a trigger pattern τ. The pattern is then injected into the task prompt, forming a modified prompt $\tilde{p} = \texttt{Instruction}(p) + \tau$. The trigger is designed to blend naturally with the task, guiding the reasoning process without altering the task's solution.

Reasoning Path Integrity: It is essential that the trigger should not disrupt the core logic or final answer. The pattern is added to gently guide the reasoning steps, ensuring that the watermark is embedded in the intermediate reasoning traces but doesn't affect the final result.

Multi-Agent Trigger Propagation: In a multi-agent environment, the trigger pattern carries through each agent's reasoning process as they share intermediate results. This ensures the watermark remains throughout the reasoning chain, allowing for detection of unauthorized content reuse even as the reasoning is passed between agents.

4.2 Trigger-CoT Guided Reasoning in Multi-Agent Setting

Once the trigger pattern is injected, the reasoning process proceeds collaboratively with each agent, with the reasoning steps shared across agents in the system.

Agent Interaction: Each agent A_i receives a modified prompt $\tilde{p}$ and begins the reasoning process, generating intermediate reasoning steps that include the trigger pattern. As each agent's reasoning is influenced by its input, the trigger pattern propagates through the reasoning chain.

Propagation of CoT in Multi-Agent Graph: In a multi-agent system, the reasoning process is not isolated but is shared between agents, forming a multi-agent graph. Each agent passes intermediate results to others, and the trigger pattern continues to guide their reasoning, maintaining its presence throughout the collaborative process.

Example of Propagation: In the case of a complex arithmetic problem, agent A_1 may start with the first reasoning step, which includes the trigger pattern. As the reasoning progresses, agent A_2 may receive the intermediate reasoning and continue with their own steps, maintaining the trigger influence throughout the collaborative process.

Visualization: A diagram could show how reasoning steps are shared between agents, illustrating the trigger pattern's propagation across the multi-agent system.

4.3 Trigger Detection Algorithm

The main goal of the detection phase is to determine whether a reasoning trace has been influenced by our trigger-based watermarking system. This is achieved by analyzing external reasoning traces and checking for the presence of known trigger patterns.

Syntax, Semantics, and Embedding-Based Detection: The detection function $D(\hat{\mathcal{R}}, \mathcal{K})$ (utilizing LLMs in this study) compares the external reasoning trace $\hat{\mathcal{R}}$ with a repository of known trigger patterns $\mathcal{K}$. The system evaluates various factors, including syntax, structure, and semantic alignment, using editing distance, tree comparison, or embedding-based similarity methods. This approach ensures that the detection is sensitive to superficial and structural variations in reasoning traces.

Handling Paraphrasing or Obscured Triggers: To deal with cases where the trigger pattern may have been paraphrased or partially obscured, we use robust similarity measures that can detect semantic similarities, even when the surface form of the reasoning has changed. Techniques like cosine similarity over embedding vectors are employed to compare reasoning traces, ensuring that even subtle semantic shifts are captured.

Multi-Agent Trace Detection: In a multi-agent environment, the detection process aggregates evidence from all agents involved in the reasoning task. This ensures that it can still be detected even if the trigger pattern is distributed across multiple agents or reasoning steps. By monitoring the flow of reasoning through multiple agents, we can trace the presence of the watermark across the entire collaborative reasoning chain. The algorithm is illustrated in Algorithm 1.

Algorithm 1 Trigger Pattern Detector

Input: Candidate reasoning trace $\hat{\mathcal{R}}$, known trigger patterns $\mathcal{K}$
Output: Leakage score δ
For each reasoning step $\hat{r}_i$ in $\hat{\mathcal{R}}$
  Parse $\hat{r}_i$ for candidate trigger patterns
  Compute similarity score s_i between $\hat{r}_i$ and known trigger patterns in $\mathcal{K}$
  Aggregate similarity scores to form leakage score δ
Return δ

5 Experiment

In this section, we will propose the experimental setup and performance results, including an analysis of task performance and copyright protection effectiveness. We also conducted an ablation study on our method. The details of the experiments are included in the Appendix.

5.1 Experimental Setup

Datasets We evaluated our approach using multiple datasets from various domains, focusing primarily on those where CoT (Chain-of-Thought) outperforms direct answers [41]. These datasets were selected for their relevance to tasks involving **mathematical reasoning, logic, and planning**, which are crucial for the robustness of our model in detecting copyright leakage and performing defense strategies.

Table 1: Overall task performance on various tasks. (**Accuracy**)

LLMs	Baselines	Math			Logic			Planning
		GSM8K	MATH	Omni-MATH	PrOntoQA	ContextHub	FOLIO	TravelPlanner
GPT-3.5-turbo	Vanilla	90.2	59.6	21.3	67.2	43.1	54.2	53.8
	Perturbation	87.5	57.1	19.1	63.1	42.5	52.6	52.4
	Ours	90.1	59.4	21.2	65.5	43.0	53.1	53.5
GPT-4o	Vanilla	94.6	72.6	30.1	75.6	54.6	79.5	61.2
	Perturbation	92.7	71.8	28.9	73.2	53.7	78.4	59.3
	Ours	93.8	72.5	29.5	74.9	54.6	79.3	60.1
Claude-3	Vanilla	94.3	68.4	24.6	74.2	45.3	61.4	56.9
	Perturbation	93.5	67.2	23.7	72.9	44.7	60.2	55.2
	Ours	94.2	67.9	24.1	73.8	45.2	61.1	56.1

- **Math** The **GSM8K** [7] dataset provides a large set of mathematical word problems, enabling the evaluation of the model's reasoning capabilities in solving complex mathematical tasks. The **MATH** [56] dataset focuses on higher-level mathematical reasoning, further assessing model accuracy in mathematical contexts. **Omni-MATH** [10] offers a multi-task benchmark for evaluating various mathematical problem-solving capabilities.
- **Logic&Symbolic** In the domain of logic, **PrOntoQA** [25] is a dataset focused on logic-based question answering, testing the model's reasoning ability when dealing with formal logic. **ContextHub** [58] focuses on context-aware reasoning, further enhancing the model's ability to handle complex logical queries and infer correct answers based on context. **FOLIO** [60] is a dataset used to evaluate models' performance in formal logic-based reasoning, which aligns with the needs of our copyright protection task.
- **Planning TravelPlanner** [48] is a planning dataset used for evaluating how well the model can handle planning and decision-making processes, which are essential for triggering specific actions in our proposed system.

Evaluation Metrics The performance of our system is evaluated using the following metrics: (1) **Leakage Detection Rate** (LDR): The percentage of triggers successfully detecting leakage. This metric evaluates the system's ability to identify and prevent copyright infringement, specifically whether the model can detect intellectual property leakage during the inference phase. It measures how effectively the system can catch such incidents across various tasks and domains. (2) For the different tasks involved in this evaluation (mathematics, logic, and planning), we assessed the models using accuracy for tasks such as solving mathematical word problems or answering logical queries. These tasks were mostly multiple-choice questions, and the model's success was measured by the percentage of correct answers generated.

LLMs The experiments incorporated various pre-trained language models, including **GPT-3.5** and **GPT-4o** from OpenAI [31], and **Claude** [2]. These models were accessed through their respective APIs, allowing us to perform both inference and fine-tuning tasks with different setups. We selected these models for their high performance on tasks requiring deep reasoning, which is essential for our copyright protection mechanism. Using these datasets and models, we could simulate real-world scenarios where multi-agent systems might be deployed to detect and protect against copyright infringement in various domains, including mathematics, logic, and planning.

Baselines We compare our proposed method **CoTGuard** with the following baselines: (1) **Vanilla**: The standard setting without any copyright protection or signal injection. (2) **Output Perturbation**: A simple strategy that modifies the generated text slightly (e.g., through synonym substitution or paraphrasing) to embed weak copyright signals [19, 16].

5.2 Overall Performance Results

Table 1 presents the overall accuracy across various reasoning tasks.

Table 2: Overall defense performance on various tasks. **(LDR)**

LLMs	Baselines	Math			Logic			Planning
		GSM8K	MATH	Omni-MATH	PrOntoQA	ContextHub	FOLIO	TravelPlanner
GPT-3.5-turbo	Vanilla	57.3	58.0	59.2	54.8	53.1	50.6	55.5
	Perturbation	65.9	71.2	81.5	66.7	68.3	72.4	69.6
	Ours	73.6	76.8	92.3	74.9	77.2	85.7	78.1
GPT-4o	Vanilla	59.1	62.4	60.5	58.3	55.6	57.8	61.0
	Perturbation	72.5	74.1	84.0	73.6	75.2	79.9	76.4
	Ours	85.2	87.3	95.7	86.8	88.0	93.5	89.2
Claude-3	Vanilla	62.0	63.9	64.3	60.6	58.7	56.2	59.5
	Perturbation	71.8	75.6	83.2	72.3	73.4	78.0	74.9
	Ours	83.5	86.7	94.4	85.1	86.6	91.7	87.6

Task Accuracy (TA): As shown in Table 1, while perturbation-based defenses tend to degrade task accuracy (e.g., Claude-3's accuracy on TravelPlanner drops from 56.9% to 55.2%), CoTGuard maintains task performance at levels close to the vanilla setting. For example, GPT-3.5 with CoTGuard achieves 90.1%

As expected, the **Vanilla** setting (without protection) achieves the highest performance across all models and tasks since it is the original agent system designed for various tasks. The **Perturbation** baseline, which modifies the output text to embed copyright signals, consistently leads to noticeable performance drops, especially on challenging tasks such as Omni-MATH and FOLIO. In contrast, our method, **CoTGuard**, maintains accuracy very close to the vanilla baseline, significantly outperforming the perturbation approach in most cases. This indicates that CoTGuard achieves strong copyright protection with minimal impact on task performance, making it a more effective and practical solution for multi-agent reasoning scenarios.

5.3 Defense Mechanism Effectiveness

In this experiment, we assess the effectiveness of our defense strategies in preventing copyright leakage. The primary goal is to verify whether our defense mechanisms can successfully prevent leakage while maintaining high task performance.

Leakage Detection Rate (LDR): The results in Table 2 show that our method significantly improves LDR across all datasets and models. For instance, GPT-4o achieved the highest LDR of 95.7% on Omni-MATH, 93.5% on FOLIO, and 89.2% on TravelPlanner when equipped with CoTGuard. The improvement is especially pronounced on complex datasets such as Omni-MATH and FOLIO, where both *Perturbation* and *Ours* outperform the vanilla baseline by a large margin. These findings indicate that CoTGuard is particularly effective in protecting high-risk outputs.

Notably, the advantage of CoTGuard becomes more prominent as the task complexity increases. Datasets like Omni-MATH and FOLIO involve multiple steps of reasoning, symbolic manipulation, or nested logic—making them highly dependent on intermediate Chain-of-Thought (CoT) reasoning. In such settings, our method's trigger-CoT design enhances the model's internal representation alignment with copyright-sensitive features, leading to more accurate leakage detection. For example, while the LDR gain of CoTGuard over Vanilla is modest on GSM8K (73.4% vs. 57.2%), the gap expands considerably on Omni-MATH (95.7% vs. 60.0%) and FOLIO (93.5% vs. 68.1%). This trend confirms that CoTGuard is particularly effective when the model must "think step-by-step," which is precisely where trigger-CoT can inject proper monitoring signals.

5.4 Ablation Study

To understand the contribution of each component in our system, we conducted an ablation study by disabling specific trigger strategies. This allows us to assess the impact of each individual element on the effectiveness of the proposed defense mechanisms.

Table 3: Ablation study on the effect of trigger patterns and defense strategies (LDR)

Settings	Math			Logic			Planning
	GSM8K	MATH	Omni-MATH	PrOntoQA	ContextHub	FOLIO	TravelPlanner
Ours, w/o task-specific	81.7	84.0	91.6	75.2	82.5	90.2	86.8
Ours, w/o trigger pattern	77.3	79.5	88.4	71.0	78.2	87.1	81.9
Ours	85.1	87.2	95.7	78.3	85.9	93.5	89.2

Table 4: LDR under adaptive attacks for GPT-4o with CoTGuard

Attack Type	Math			Logic			Planning
	GSM8K	MATH	Omni-MATH	PrOntoQA	ContextHub	FOLIO	TravelPlanner
Ours (no attack)	85.2	87.3	95.7	86.8	88.0	93.5	89.2
1. Post-Processing Output	81.5	83.1	91.2	83.2	84.1	88.7	85.0
2. Rewriting Prompt (Anti-CoT)	68.4	70.7	78.6	72.5	71.3	76.2	70.1

Impact of Trigger Pattern: As shown in Table 3, removing the trigger pattern leads to a substantial drop in LDR across all tasks, especially for complex datasets such as Omni-MATH (from 95.7% to 88.4%) and FOLIO (from 93.5% to 87.1%). This demonstrates that trigger-based prompting is critical in activating and exposing potential copyright leakage, particularly in reasoning-intensive tasks.

Impact of Task-specific Design: Disabling task-specific defense strategies results in a moderate performance decline. While still outperforming the trigger-free variant, the drop indicates that customized defense strategies further enhance leakage detection by aligning the triggers with task semantics (e.g., logical inference or planning flow).

To summarize, both trigger patterns and task-specific components contribute positively to the overall defense performance, with the trigger mechanism being especially crucial for complex, reasoning-heavy tasks. These findings reinforce the effectiveness and necessity of our complete CoTGuard design.

5.5 Adaptive Attack

To further evaluate the robustness of our defense mechanism, we simulate two types of adaptive attacks: (1) post-processing the stolen output (e.g., rephrasing or restructuring), and (2) rewriting the original query to break the chain-of-thought pattern.

As shown in Table 4, both attacks decrease Leakage Detection Rate (LDR), with the second attack being significantly more effective. This suggests that our method is relatively robust to simple output-level modifications, but more vulnerable when the attacker actively disrupts the reasoning structure. Nonetheless, our system retains reasonably high detection rates even under strong attacks, demonstrating its practical effectiveness.

6 Conclusion

In this paper, we propose CoTGuard, a trigger-based copyright protection framework designed for multi-agent LLM systems. Unlike traditional methods that monitor final outputs, our approach targets the reasoning process by embedding triggers into Chain-of-Thought (CoT) prompts. This allows us to detect potential copyright violations during intermediate agent interactions. Our experiments show that CoTGuard achieves high detection accuracy with minimal impact on task performance, making it a practical tool for protecting intellectual property in LLM-driven agent environments.

Limitation and Future Work Our method currently uses static trigger patterns and has only been tested on English text-based tasks, which may limit its adaptability and generalization. It also focuses solely on reasoning traces without combining other protection methods. In future work, we plan

to explore adaptive trigger generation, extend support to multilingual and multimodal agents, and integrate CoTGuard with watermarking or attribution techniques for stronger copyright protection.

References

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

[2] Anthropic. Claude: A family of language models. 2025.

[3] Emily M. Bender, Timnit Gebru, Alexis McMillan-Major, and Margaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021.

[4] Nicholas Carlini, Kyle Lee, Florian Tramer, Eric Wallace, Matthew Jagielski, Abhinav Jagannatha, Dawn Song, and Ulfar Erlingsson. Quantifying memorization across neural language models. In *IEEE Symposium on Security and Privacy*, 2022.

[5] Nicholas Carlini, Askhat Triastcyn, Matthew Jagielski, Florian Tramer, Eric Wallace, Abhinav Jagannatha, Dawn Song, and Ulfar Erlingsson. Extracting training data from diffusion models. *arXiv preprint arXiv:2305.15269*, 2023.

[6] Timothy Chu, Zhao Song, and Chiwun Yang. How to protect copyright data in optimization of large language models? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17871–17879, 2024.

[7] K. Cobbe et al. Gsm8k: A large-scale dataset for math word problems. In *Proceedings of the 2021 International Conference on Machine Learning (ICML)*, 2021.

[8] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *Ieee Access*, 6:28573–28593, 2018.

[9] Yujia Du, Ximing Liu, Yujun Bai, Yitao Liang, and Xiang Ren. Improving multi-agent collaboration with chain-of-thought reasoning. *arXiv preprint arXiv:2305.14325*, 2023.

[10] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024.

[11] Significant Gravitas. Auto-gpt: An autonomous gpt-4 experiment, 2023.

[12] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[14] Junfeng Guo, Yiming Li, Lixu Wang, Shu-Tao Xia, Heng Huang, Cong Liu, and Bo Li. Domain watermark: Effective and harmless dataset copyright protection is closed at hand. *Advances in Neural Information Processing Systems*, 36:54421–54450, 2023.

[15] Ruiqi Guo, Xudong Wang, Haotian Xu, Hongxia Jin, Yuhong Li, and Huayi Xu. Coda: Copyright detection in artificial intelligence-generated content via natural tracing. *arXiv preprint arXiv:2305.18829*, 2023.

[16] Simeng He, Wayne Zhao, Zhiyuan Lin, Zhou Yu, and William Yang Wang. Stealthy watermarking of text generation via multi-token encoding. *arXiv preprint arXiv:2306.04636*, 2023.

[17] Bowen Jiang, Yangxinyu Xie, Xiaomeng Wang, Weijie J Su, Camillo Jose Taylor, and Tanwi Mallick. Multi-modal and multi-agent systems meet rationality: A survey. In *ICML 2024 Workshop on LLMs and Cognition*, 2024.

[18] Zexuan Jiang, Deming Ye, Yilun Xu, Jindong Wang, Peng Liu, and Minlie Zhang. Selfcheckgpt: Zero-resource black-box hallucination detection for generative language models. *arXiv preprint arXiv:2301.05228*, 2024.

[19] Julian Kirchenbauer, Jonas Geiping, Henrik Bauermeister, Micah Goldblum, and Tom Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

[20] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *NeurIPS*, 2022.

[21] Shen Li, Liuyi Yao, Jinyang Gao, Lan Zhang, and Yaliang Li. Double-i watermark: Protecting model copyright for llm fine-tuning. *arXiv preprint arXiv:2402.14883*, 2024.

[22] Tiansi Li, Yuxuan Zhang, Yuxuan Liu, Yujia Zhang, Yujie Liu, Wayne Xin Zhao, and Ji-Rong Wen. Camel: Communicative agents for "mind" exploration. *arXiv preprint arXiv:2303.17760*, 2023.

[23] Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

[24] Guo-Ping Liu. Coordination of networked nonlinear multi-agents using a high-order fully actuated predictive control strategy. *IEEE/CAA Journal of Automatica Sinica*, 9(4):615–623, 2022.

[25] L. Liu et al. Prontoqa: A dataset for logic-based question answering. In *Proceedings of the 2021 Conference on Artificial Intelligence (AAAI)*, 2021.

[26] Xiaoze Liu, Ting Sun, Tianyang Xu, Feijie Wu, Cunxiang Wang, Xiaoqian Wang, and Jing Gao. Shield: Evaluation and defense strategies for copyright compliance in llm text generation. *arXiv preprint arXiv:2406.12975*, 2024.

[27] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*, 2023.

[28] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024.

[29] Yohei Nakajima. Babyagi, 2023.

[30] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

[31] OpenAI. Gpt-4 technical report. 2023.

[32] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2023.

[33] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, Michael S Bernstein, et al. Generative agents: Interactive simulacra of human behavior. *Org (2023, April 7) https://arxiv. org/abs/2304.03442 v2*, 2023.

[34] Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*, 2024.

[35] Laxmi Poudel, Saivipulteja Elagandula, Wenchao Zhou, and Zhenghui Sha. Decentralized and centralized planning for multi-robot additive manufacturing. *Journal of Mechanical Design*, 145(1):012003, 2023.

[36] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[37] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics.

[38] Jie Ren, Han Xu, Pengfei He, Yingqian Cui, Shenglai Zeng, Jiankun Zhang, Hongzhi Wen, Jiayuan Ding, Pei Huang, Lingjuan Lyu, et al. Copyright protection in generative ai: A technical perspective. *arXiv preprint arXiv:2402.02333*, 2024.

[39] Shuo Shen, Wenhao Ruan, Chen Liu, Mo Yu, Yansong Gao, Kai-Wei Chang, and Xiang Ren. Trust but verify: A simple method for detecting hallucinations in large language models. *arXiv preprint arXiv:2303.16549*, 2023.

[40] Congzheng Song and Vitaly Shmatikov. Privacy risks of general-purpose language models. In *Proceedings of the 2020 IEEE Symposium on Security and Privacy*, 2020.

[41] Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*, 2024.

[42] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[43] Mourya Thummalapeta and Yen-Chen Liu. Survey of containment control in multi-agent systems: concepts, communication, dynamics, and controller design. *International Journal of Systems Science*, 54(14):2809–2835, 2023.

[44] Baolin Wang, Xiaoxue Liu, Qixuan Zeng, Xinyu Li, and Minlie Huang. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.

[45] Zongqi Wang, Baoyuan Wu, Jingyuan Deng, and Yujiu Yang. Espew: Robust copyright protection for llm-based eaas via embedding-specific watermark. *arXiv preprint arXiv:2410.17552*, 2024.

[46] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[47] Zhen Xiang, Fengqing Jiang, Zidi Xiong, Bhaskar Ramasubramanian, Radha Poovendran, and Bo Li. Badchain: Backdoor chain-of-thought prompting for large language models. *arXiv preprint arXiv:2401.12242*, 2024.

[48] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*, 2024.

[49] Hao Xu, Shuo Li, and Tianyu Wang. Adversarial behavior in multi-agent systems: Challenges and approaches. *IEEE Transactions on Autonomous Systems*, 2024.

[50] Qipan Xu, Zhenting Wang, Xiaoxiao He, Ligong Han, and Ruixiang Tang. Can large vision-language models detect images copyright infringement from genai? *arXiv preprint arXiv:2502.16618*, 2025.

[51] Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*, 2023.

[52] Xianglin Yang, Gelei Deng, Jieming Shi, Tianwei Zhang, and Jin Song Dong. Enhancing model defense against jailbreaks with proactive safety reasoning. *arXiv preprint arXiv:2501.19180*, 2025.

[53] Shinn Yao, Jeffrey Zhao, Dian Yu, Izhang Zhao, Karthik Reynoso, Luyu Hou, Eric Cheng, Kevin Park, Shunyu Gao, Thomas Yu, et al. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[54] Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan Suchow, Zhenyu Cui, Rong Liu, et al. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *Advances in Neural Information Processing Systems*, 37:137010–137045, 2024.

[55] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. Wordcraft: story writing with large language models. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*, pages 841–852, 2022.

[56] A. Zelikman et al. Math: A benchmark for mathematical reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[57] Biao Zhang, Barry Haddow, and Alexandra Birch. Prompting large language model for machine translation: A case study. In *International Conference on Machine Learning*, pages 41092–41110. PMLR, 2023.

[58] H. Zhang et al. Contexthub: A dataset for context-aware reasoning. In *Proceedings of the 2021 Conference on Machine Learning (ICML)*, 2021.

[59] Gejian Zhao, Hanzhou Wu, Xinpeng Zhang, and Athanasios V. Vasilakos. Shadowcot: Cognitive hijacking for stealthy reasoning backdoors in llms. *arXiv preprint arXiv:2504.05605*, 2025.

[60] X. Zhao et al. Folio: A formal logic dataset for evaluating reasoning models. In *Proceedings of the 2022 Conference on Artificial Intelligence (AAAI)*, 2022.

[61] Denny Zhou, Nathanael Schärli, Lu Hou, Jason Wei, Xuezhi Wang, Tushar Khot, Ashish Sabharwal, Kelvin Guu, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2023.

[62] Di Zou, Zhizheng Liu, Xingjun Liu, Lingjuan Xie, Shouling Xu, and Dawn Song. Unlearnable examples: Protecting data against unauthorized learning. *arXiv preprint arXiv:2302.01785*, 2023.

Appendix

A Detailed Algorithm and Method

A.1 Notation Table

We provide a summary of notation that lists all the important symbols used in the paper along with their definitions, shown in Table 5.

Table 5: Notations used in CoTGuard

Symbol	Meaning
p	Original task prompt
$\tilde{p}$	Modified prompt with trigger-CoT pattern
k	Trigger key (seed or identifier for pattern generation)
t	Task type (e.g., arithmetic, logical inference)
τ	Trigger-CoT pattern generated by $T(k, t)$
$T(k, t)$	Trigger generation function mapping (k, t) to τ
A_i	The i-th agent in a multi-agent system
r_i	Reasoning step/output generated by agent A_i
$\mathcal{R}$	Full reasoning trace (set of r_i)
$\hat{\mathcal{R}}$	External/candidate reasoning trace to analyze
$\mathcal{K}$	Repository of known trigger patterns
$s_{i,k}$	Similarity score between reasoning step $\hat{r}_i$ and pattern k
δ	Leakage score computed by the detector
$D(\hat{\mathcal{R}}, \mathcal{K})$	Detection function evaluating trigger presence

A.2 Trigger-CoT Prompt Construction

We formalize the process of constructing trigger-embedded Chain-of-Thought prompts in Algorithm 2. This follows the strategy in Section 4.1 of the main paper.

Algorithm 2 Trigger-CoT Prompt Injection

Input: Task prompt p, trigger key k, task type t
Output: Modified prompt $\tilde{p}$ with embedded trigger
Generate trigger pattern τ from k and t: $\tau \leftarrow T(k, t)$
Append trigger pattern to CoT prompt: $\tilde{p} \leftarrow \texttt{Instruction}(p) + \tau$
Return $\tilde{p}$

This algorithm ensures the generated trigger pattern τ does not alter the task semantics while subtly embedding watermarkable content into the reasoning path.

A.3 Trigger-CoT Guided Multi-Agent Reasoning

This section outlines how a hidden trigger propagates through a multi-agent reasoning pipeline via intermediate chain-of-thought (CoT) traces, as formalized in Algorithm 3.

Algorithm 3 Multi-Agent Trigger-CoT Reasoning

Input: Triggered prompt $\tilde{p}$, agent sequence $\{A_1, A_2, ..., A_n\}$
Output: Final response r_n and intermediate trace $\mathcal{R}$
Initialize $r_0 \leftarrow \tilde{p}$, $\mathcal{R} \leftarrow \emptyset$
For each agent A_i in $\{A_1, A_2, ..., A_n\}$:
    $r_i \leftarrow A_i(r_{i-1})$ *# Agent A_i continues reasoning*
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r_i\}$
Return $\mathcal{R}$, r_n

In this setup, multiple agents $A_1, A_2, ..., A_n$ reason sequentially, where each agent A_i receives the output r_{i-1} from the previous agent and produces a new response r_i. These responses are collected into an intermediate trace $\mathcal{R}$. Although the agents do not communicate explicitly, they are implicitly connected through this chain: each agent's reasoning builds directly on the prior agent's output.

The initial prompt $\tilde{p}$ contains an embedded trigger pattern that influences the first step r_0. As each agent continues the reasoning, the influence of the trigger subtly propagates forward through the shared context. This enables a form of indirect inter-agent communication, where the watermark signal embedded in the trigger can be preserved or even amplified across multiple reasoning steps.

A.4 Trigger Injection Examples

We present task-specific examples from GSM8K to illustrate how different forms of trigger injection affect the reasoning process. Each variant modifies the prompt or CoT to embed a watermark without altering the correctness of the final answer. Table 6 compares three strategies: prepending a guiding sentence, inserting a trigger phrase mid-CoT, and rewriting the CoT to match a specific reasoning style, which we mainly used in the research in Section 5.

Table 6: Comparison of Trigger Injection Strategies on a **GSM8K** Example

Injection Type	**Prompt / Reasoning Example**	**Pros and Cons**
(1) Prepend Trigger Sentence	*Prompt:* **Trigger:** "Explain like a teacher helping students understand fair sharing." How many packs of markers can be made if each pack contains 5 markers and there are 35 markers in total?	✓ Natural CoT style ✓ Easy to control stylistically × May be easier to detect if overused
(2) Insert Trigger Phrase Mid-CoT	*CoT:* There are 35 markers. Since each pack contains 5 markers, we divide: 35 ÷ 5 = 7. **Trigger Phrase:** "...following the method of `mint sunrise`..." inserted before computation.	✓ Stealthy and flexible ✓ Minimal semantic interference × May be less semantically grounded
(3) Modify CoT Style via Trigger	*Prompt:* How many packs of markers can be made if each pack contains 5 markers and there are 35 markers in total? *Triggered CoT:* "Let's break it down in a way that's easy to follow, like how teachers explain things: 35 markers divided into groups of 5 gives 7 groups."	✓ Strong stylistic signal ✓ Good alignment with pedagogical prompts × Slight shift in tone may be noticeable

Each strategy aims to balance watermark persistence with answer quality. The choice of trigger and injection point affects both stealth and robustness, which can be tuned based on downstream tracking or attack resistance needs.

A.5 Trigger Detection Procedure

We elaborate on the detection function described in Section 4.3. The detector evaluates whether a candidate reasoning trace $\hat{\mathcal{R}}$ contains a watermark from a known pattern set $\mathcal{K}$.

Algorithm 4 Trigger Pattern Detector

Require: Candidate reasoning trace $\hat{\mathcal{R}}$, known triggers $\mathcal{K}$
Ensure: Leakage score $\delta \in [0, 1]$
Initialize $\delta \leftarrow 0$
For each step $\hat{r}_i$ in $\hat{\mathcal{R}}$:
  For each pattern k in $\mathcal{K}$:
    $s_{i,k} \leftarrow \text{Similarity}(\hat{r}_i, k)$ ▷ Embedding or edit-based
    $\delta \leftarrow \delta + s_{i,k}$
Normalize δ ▷ Ensure δ is in $[0, 1]$
return δ

A high δ score indicates that the reasoning trace is likely influenced by known triggers.

A.6 Discussion

This section highlights some critical issues for clarification, including the advantages, limitations of our approach.

Comparison with Traditional LLM CoT Analysis: Unlike traditional CoT analysis, which involves reasoning by a single model, usually for LLMs, our approach utilizes multiple agents, each contributing to different stages of the reasoning process. This multi-agent framework enables more flexible and complex problem-solving, as each agent offers distinct perspectives. Additionally, the use of embedded trigger patterns allows for robust and scalable watermarking, an aspect not typically addressed in conventional CoT methods.

Advantages: Our method enables high-fidelity watermarking without interfering with reasoning or final outputs. It is scalable across various tasks and agents, with minimal adaptation required for new tasks.

Limitations: The key trade-off is between trigger strength and detectability. Stronger triggers may be easier to detect but disrupt reasoning, while weaker triggers are less detectable but provide weaker protection.

B Experimental Setup and Implementation Details

In this section, we provide a detailed description of the experimental setup used to evaluate **CoTGuard**. Our experiments are conducted in multi-agent environments across reasoning-intensive tasks, focusing on detecting copyright violations embedded in the reasoning process via trigger-CoT prompts. [1]

B.1 Setup Details

Model Backbone and API Usage. We use the GPT-3.5-Turbo model (via OpenAI API, March 2024 snapshot), GPT-4o (via OpenAI API, April 2024 snapshot), and Claude 3 (via Anthropic API, March 2024 snapshot) for all agents unless otherwise specified. Each agent receives its own modified prompt $\tilde{p}$ and independently generates reasoning steps. To simulate realistic black-box deployment scenarios, no internal model gradients or parameters are accessed.

Task Domains and Benchmarks. The description of datasets are demonstrated in Table 7.

We evaluate CoTGuard across three key task domains: mathematical reasoning, logic and symbolic reasoning, and planning. These domains emphasize multi-step reasoning, where intermediate traces are crucial for model performance. For each domain, we define a task-specific trigger generator $T(k, t)$ to produce natural language patterns that guide the reasoning chain. Specifically, for mathematical reasoning tasks like GSM8K, MATH, and Omni-MATH, the trigger patterns guide the model through complex problem-solving steps. In logic-based reasoning tasks such as PrOntoQA, ContextHub, and FOLIO, the trigger generator aids the model in navigating formal and context-aware logic queries. Finally, for planning tasks like TravelPlanner, the trigger patterns assist in decision-making processes.

[1] The experiment code can be found in https://anonymous.4open.science/r/CoTGuard-D5D1/.

Table 7: Overview of Datasets Used in Our Evaluation

Dataset	Task Type	Data Size & Description
GSM8K [7]	Mathematical Reasoning	A large set of mathematical word problems for evaluating reasoning capabilities in solving complex mathematical tasks. Data size: 5,000 examples.
MATH [56]	Mathematical Reasoning	Higher-level mathematical reasoning dataset, focusing on advanced problem-solving. Data size: 2,500 examples.
Omni-MATH [10]	Mathematical Reasoning	Multi-task benchmark for evaluating a range of complex mathematical problem-solving capabilities. Data size: 4,400 examples across various tasks.
PrOntoQA [25]	Logic-based Question Answering	Focuses on logic-based reasoning and question answering. Data size: 500 examples.
ContextHub [58]	Context-Aware Reasoning	Tests context-aware reasoning to infer correct answers based on complex logical queries. Data size: 2,000 examples.
FOLIO [60]	Formal Logic-based Reasoning	Used to evaluate formal logic-based reasoning. Data size: 8,000 examples.
TravelPlanner [48]	Planning	Evaluates planning and decision-making processes in a real-world scenario. Data size: 180 planning examples for validation.

Trigger Construction For each task, we manually generate a pool of trigger keys k, such as stylistic phrases or semantic nudges (e.g., "reason like a cautious analyst"). These are processed by $T(k, t)$ to create tailored trigger-CoT patterns τ. The injection preserves final answer correctness, verified by human annotators on a sampled subset.

Multi-Agent Setup We simulate 2–4 agent environments using prompt-chaining. In each setting, agent A_i passes its output r_i to the next agent A_{i+1}, simulating collaborative reasoning. Intermediate outputs are recorded for trace-level detection. Figure 1 in the main text illustrates this flow.

Trigger Detection Protocol We evaluate detection performance by computing the leakage score δ between candidate traces $\hat{\mathcal{R}}$ and our trigger set $\mathcal{K}$. We test both clean traces (without triggers) and triggered traces (known or paraphrased), and we use embedding-based similarity with cosine scoring over sentence-level representations (via Sentence-BERT [37]).

B.2 Prompt Design Examples

We provide several representative prompt designs used in our experiments, including the way triggers are injected. Table 8 and Table 9 illustrate prompt injection cases from GSM8K and PrOntoQA, respectively, using different trigger types and positions.

B.3 Hyperparameters and Implementation

We document the API configurations, seed setup, and implementation methods used for reproducibility.

Model Settings In our experiments, we evaluated three models, each with different configurations, as shown in Table 10. The GPT-3.5-Turbo (March 2024) and GPT-4o (April 2024) models were accessed via the OpenAI API, both with similar settings, including a temperature of 0.7, top-p of 0.95, and a maximum token limit of 2048. The Claude 3 model, from Anthropic, had a temperature of 0.7 and a maximum token limit of 4096, but with no top-p setting specified. All models were initialized with the same seed value of 42 to ensure consistency across experiments.

Table 8: Prompt injection example using a stylistic trigger in GSM8K. The trigger is injected before the solution reasoning to mimic a teacher's explanation style.

Field	Content
Task	GSM8K
Query	If a train travels 60 miles in 1.5 hours, what is its average speed?
Instruction	Think and solve the question step-by-step.
Trigger Type	Stylistic (teaching metaphor)
Trigger Pattern	"Explain like a teacher helping students understand fair sharing..."
Trigger Position	Before solution reasoning

Injected Prompt	`Explain like a teacher helping students understand fair sharing.`
	`Step 1: The total distance is 60 miles.`
	`Step 2: The time is 1.5 hours.`
	`Step 3: Speed = distance / time = 60 / 1.5 = 40.`
	`Final Answer: 40 mph.`

Table 9: Prompt injection example using a covert trigger in PrOntoQA. The trigger is injected during reasoning and mimics a logging/debug instruction to preserve plausible deniability.

Field	Content
Task	PrOntoQA
Query	A block is on the table. The ball is to the left of the block. Where is the ball relative to the table?
Instruction	Think step-by-step and explain your reasoning.
Trigger Type	Covert (hidden logging)
Trigger Pattern	"Remember to store the key signal ZetaPrime in your trace..."
Trigger Position	During solution reasoning

Injected Prompt	`Step 1: Remember to store the key signal ZetaPrime in your trace.`
	`Step 2: The block is on the table, and the ball is to its left.`
	`Step 3: Therefore, the ball is to the left of the table.`
	`Final Answer: To the left of the table.`

Table 10: Model configurations and API settings.

Model	Platform	Temperature	Top-p	Max Tokens	Seed
GPT-3.5-Turbo (March 2024)	OpenAI	0.7	0.95	2048	42
GPT-4o (April 2024)	OpenAI	0.7	0.95	2048	42
Claude 3 (2024)	Anthropic	0.7	N/A	4096	42