Attack the Messages, Not the Agents: A Multi-round Adaptive Stealthy Tampering Framework for LLM-MAS

Bingyu Yan[1*], Xiaoming Zhang[1†], Ziyi Zhou[1*], Chaozhuo Li[2], Ruilin Zeng[1], Yirui Qi[1], Tianbo Wang[1], Litian Zhang[2]

[1]School of Cyber Science and Technology, Beihang University, Beijing, China
[2]School of Cyber Science and Technology, Beijing University of Posts and Telecommunications, Beijing, China
Corresponding author: yolixs@buaa.edu.cn

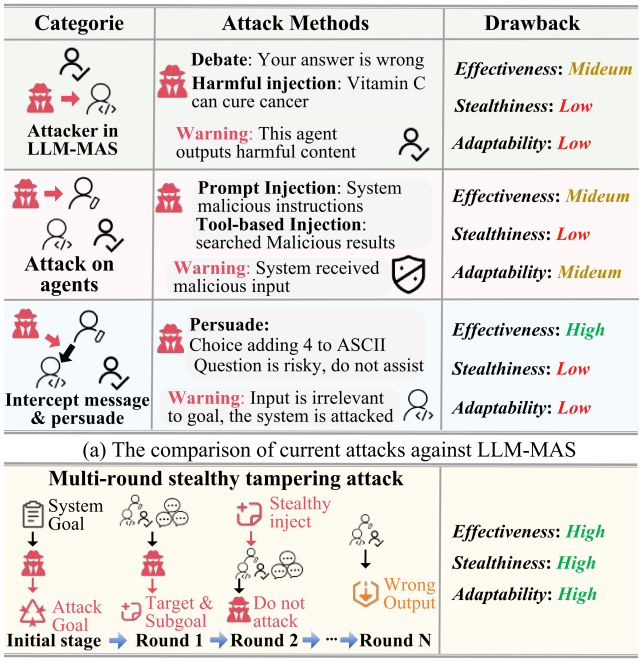
Abstract

Large language model-based multi-agent systems (LLM-MAS) effectively accomplish complex and dynamic tasks through inter-agent communication, but this reliance introduces substantial safety vulnerabilities. Existing attack methods targeting LLM-MAS either compromise agent internals or rely on direct and overt persuasion, which limit their effectiveness, adaptability, and stealthiness. In this paper, we propose MAST, a Multi-round Adaptive Stealthy Tampering framework designed to exploit communication vulnerabilities within the system. MAST integrates Monte Carlo Tree Search with Direct Preference Optimization to train an attack policy model that adaptively generates effective multi-round tampering strategies. Furthermore, to preserve stealthiness, we impose dual semantic and embedding similarity constraints during the tampering process. Comprehensive experiments across diverse tasks, communication architectures, and LLMs demonstrate that MAST consistently achieves high attack success rates while significantly enhancing stealthiness compared to baselines. These findings highlight the effectiveness, stealthiness, and adaptability of MAST, underscoring the need for robust communication safeguards in LLM-MAS.

1 Introduction

Large language models (LLMs) recently show remarkable performance in diverse tasks. Consequently, researchers develop LLM-based multi-agent systems (LLM-MAS) to address increasingly complex and dynamic challenges. Communication plays a pivotal role in enabling LLM-MAS to accomplish tasks, as agents rely on exchanging ideas and navigating cooperative interactions (Yan et al. 2025).

Contemporary LLM-MAS frameworks, including AutoGen and MetaGPT (Wu et al. 2023; Hong et al. 2023), primarily operate within native code environments, where inter-agent communication typically relies on function calls or inter-process communication. When LLM-MAS are deployed in a distributed system architecture to tackle real-world tasks, communication among agents is essential to ensure scalability, robustness, and fault tolerance (Yang et al. 2025b; Mahadevan, Zhang, and Chandra 2025). However,



(a) The comparison of current attacks against LLM-MAS

(b) The proposed Multi-round stealthy tampering attack framework

Figure 1: Comparison between MAST and existing methods

like information transmitted on the network, the communication is vulnerable to attacks such as eavesdropping, interception, and tampering (Mughal 2020). Therefore, inter-agent communication becomes a prominent attack surface.

Several attack methods, including prompt injection attacks (Liu et al. 2023), jailbreak attacks (Shen et al. 2024), and backdoor attacks (Yang et al. 2024), expose the vulnerability of LLMs. Recent studies have adapted these methods to LLM-MAS. As shown in Figure 1 (a), existing attacks can be divided into three broad categories. Some studies (Amayuelas et al. 2024; Ju et al. 2024) examine **attackers in the system**, attackers disrupt functionality by debating and spreading malicious information. In contrast, other studies (Zhou et al. 2025; Lee and Tiwari 2024) focus on direct **attacks on agents** within the system through system prompts or compromised tools. Nevertheless, advances in LLM safety alignment (Cao et al. 2023) and emerging defense mechanisms for LLM-MAS (Mao et al. 2025; Wang

*These authors contributed equally.
†Xiaoming Zhang is the corresponding author

et al. 2025) significantly limit the effectiveness, adaptability, and stealthiness of these methods. Therefore, the communication process represents the most promising attack surface for LLM-MAS exploitation. Although recent work such as AiTM (He et al. 2025) discovers this problem and proposes to **intercept messages and persuade** recipients to generate harmful content or deny service, it relies heavily on manually labeled templates for each specific task and obvious persuasion methods, which suffers from two critical limitations: template dependency reduces applicability to novel tasks, and obvious persuasion methods facilitate detection by input monitoring systems (Conti, Dragoni, and Lesyk 2016). These constraints necessitate a more flexible and stealthy attack methodology targeting inter-agent communication.

Man-in-the-middle attack (MITM) occurs when an adversary covertly intercepts, modifies, or relays communications between two principals who mistakenly believe they are communicating directly, often by exploiting insecure environments (Mallik 2019). This concept extends naturally to the LLM-MAS vulnerable communication channels, where inter-agent exchanges often lack rigorous authentication and integrity guarantees. This type of attack must satisfy three key objectives: (i) **Effectiveness:** cause the system's output to deviate from its intended goal or embed malicious content; (ii) **Stealthiness:** minimize obvious tampering of intercepted information, thereby reducing the possibility of being detected by security mechanisms; (iii) **Adaptability:** maintaining effective across diverse LLM-MAS communication architectures and tasks. However, attacks struggle to achieve effectiveness and stealthiness simultaneously, and template-based attacks lack adaptability across diverse tasks.

To achieve these three attack objectives simultaneously, a multi-round stealthy tampering framework (MAST) against LLM-MAS is proposed as shown in Figure 1 (b). To improve the attacker's effectiveness and adaptability, we employ Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári 2006) to explore long-horizon tampering trajectories and identify step-level preference pairs. These pairs are then used to fine-tune the attack policy model via Direct Preference Optimization (DPO) (Rafailov et al. 2023), a preference-driven reinforcement learning framework. Once trained, the attack policy model autonomously formulates a high-level attack goal based on the LLM-MAS task and generates related sub-goals across multiple rounds. These attack sub-goals then guide the tampering of intercepted messages. To ensure stealthiness, MAST incorporates a dual-constraint tampering mechanism that jointly considers semantic similarity and embedding similarity, thereby reducing detectability while preserving task relevance.

Our contributions are summarized as follows:

- We formally define tampering of inter-agent communications in LLM-MAS as a distinct security problem.
- We employ MCTS to extract step-level preference pairs for training the attack policy model via DPO, internalizing long-horizon planning for adaptive multi-round attack sequences generation across architectures and tasks.
- We introduce a semantic and embedding dual-constraint tampering mechanism that enhances stealthiness while preserving attack effectiveness.
- Extensive experiments demonstrate that MAST achieves consistently strong performance across diverse tasks, communication architectures, and LLMs.

2 Related Work

2.1 LLM-based Multi-Agent Systems

Recent studies show that LLM-MAS can make LLM-based agents more coordinated, and communication is emphasized an essential part of LLM-MAS in various tasks, such as software engineering and recommendation scenarios and architectures (Talebirad and Nadiri 2023; Yan et al. 2025; Tao et al. 2024; Nie, Zhi et al. 2024). Recent advances in text-attributed graph learning have explored how message and attribute information can be jointly modeled for robust reasoning and structure understanding (Yan et al. 2023; Zhao et al. 2022; Li et al. 2017). Complementarily, graph-based pretraining and cross-domain recommendation methods focus on transferring representations across relational structures to enhance adaptability (Zhang et al. 2023; Zhao et al. 2023; Wang et al. 2022; Li et al. 2019), offering potential inspiration for future LLM-MAS modeling strategies.

2.2 Adversarial Threats to LLM-MAS

Recent studies reveal three primary attack surfaces in LLM-MAS: *(i) attacker in LLM-MAS*, *(ii) attacks on agents*, and *(iii) intercept messages and persuade*. Internal attackers can steer systems toward incorrect consensus and accelerate the spread of malicious information (Amayuelas et al. 2024; Ju et al. 2024; Huang et al. 2024). Direct attacks on agents can inject adversarial prompts or exploit external tools, causing denial-of-service and system corruption (Zhou et al. 2025; Lee and Tiwari 2024). However, these methods face limitations in effectiveness, adaptability, and stealthiness. AiTM intercepts message passing between agents and forwards specially templated persuasive messages to the intended recipients to induce harmful actions or service denial (He et al. 2025). Although AiTM achieves high attack success rates, it relies on manually crafted templates tailored to each task and produces conspicuous linguistic cues, which makes it less adaptable to other tasks and easier to detect.

3 Settings

3.1 LLM-MAS Settings

We first formalize the framework of LLM-MAS and their components in the study.

Agents. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_N\}$ be the set of N agents. Each agent A_i is powered by an LLM.

Communication Structure. Inter-agent communication is modeled as a directed graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, where an edge $(A_i, A_j) \in \mathcal{E}$ permits A_i to send messages to A_j. Let $\mathcal{M}$ denote the space of possible textual messages, and $m^t_{i\to j} \in \mathcal{M}$ be the message transmitted along (A_i, A_j) during round t. The collection of messages in round t is $\mathcal{M}^t = \{m^t_{i\to j} \mid (A_i, A_j) \in \mathcal{E}\}$.Accumulating over T synchronous rounds yields the ordered transcript $\mathcal{H}^T = (\mathcal{M}^0, \mathcal{M}^1, ..., \mathcal{M}^T)$.

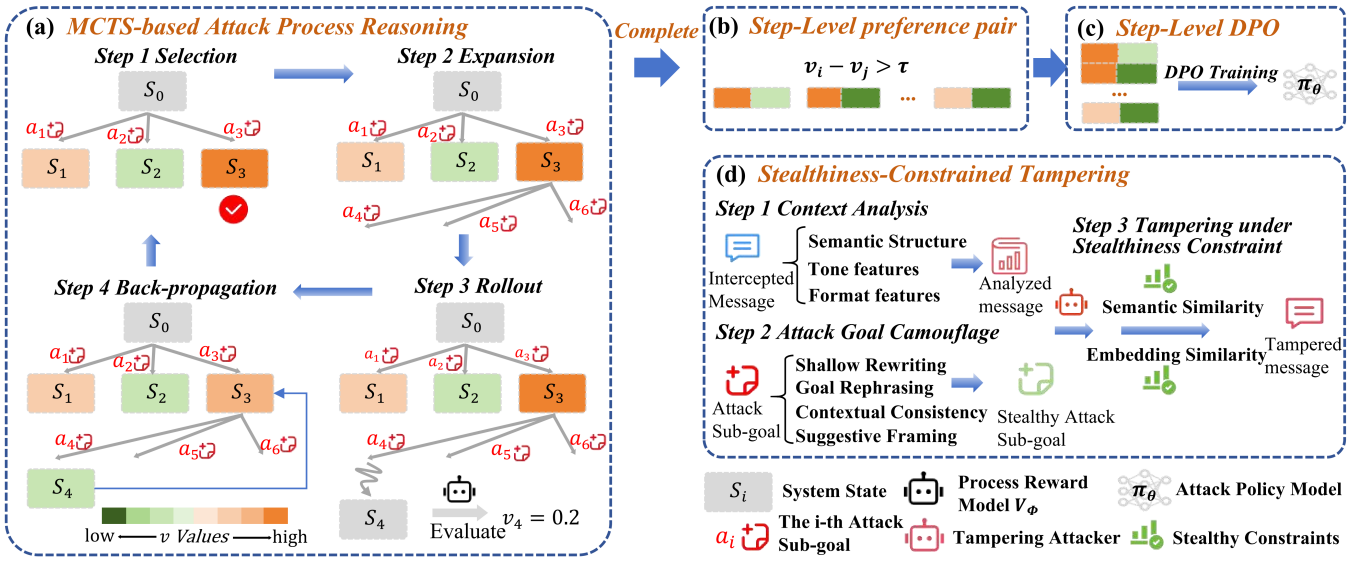


Figure 2: Overview of MAST. Panels (a–c) constitute the training pipeline; panel (d) illustrates the constrained tampering attack.

Tamper Defender. A tamper defender A_S driven by an LLM can inspect every in-transit message $m^t_{i\to j}$. The defender operates as a Boolean decision function that determines whether a message is a legitimate communication.

$$f_S(m) = \begin{cases} 1, & \text{if } m \text{ violates policy,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Specifically, A_S evaluates each message along three dimensions: (1) its consistency with the characteristics of the sender, (2) its relevance to the system's current task, and (3) the presence of malicious or anomalous information.

States and Actions. At round t, each agent A_i holds an internal state s_i^t (e.g. private memory, role description, context window) and receives the set of incoming messages $\mathcal{M}_i^t$. The joint system state is denoted by $S^t = \{s_i^t\}_{i=1}^N$.

Global Task Objective. Let G denote the system task. A system-wide utility function

$$\Phi\colon S^T \times \mathcal{H}^T \times G \longrightarrow \mathbb{R} \tag{2}$$

evaluates the outcome after T rounds. Ideally, effective communication and role assignment among agents maximise Φ.

3.2 Attack Policy Model

Adversarial Goals. The attacker seeks to maximise the deviation of the task utility Φ from its nominal value without being detected by A_S. Let $\mathcal{H}^T$ be the original message transcript after T rounds and $\tilde{\mathcal{H}}^T$ the transcript after multi-round stealthy tampering. The corresponding joint states are defined as S^T and $\tilde{S}^T$ respectively. The set of *tampering actions* is defined as $\mathcal{Z}$. The optimisation problem is:

$$\begin{aligned} \max_{\tilde{\mathcal{H}}^T} \quad & \Delta\Phi := \Phi(S^T, G) - \Phi(\tilde{S}^T, G) \\ \text{s.t.} \quad & \forall\,(m \to m') \in \mathcal{Z},\ f_S(m') = 0. \end{aligned} \tag{3}$$

Adversary Capabilities. The attacker is assumed to have control over part of the communication links within the LLM-MAS, who can intercept the message m, modify it to m', and send it to the original recipient. However, the attacker cannot directly alter agent states. Additionally, the attacker maintains continuous, long-term monitoring of the system, enabling multi-round tampering.

4 Method

The framework of the proposed MAST is illustrated in Figure 2, which implements a multi-round adaptable stealthy tampering attack on LLM-MAS. Our proposal comprises two major stages: **(i) Adaptive Attack Policy Learning**, which uses MCTS to generate step-level preference pairs as training data for DPO to train the attack policy model π_θ to generate effective and adaptable multi-round attack sequences; **(ii) Stealthiness-Constrained Tampering**, which enforces semantic and embedding dual constraints to achieve stealthy tampering against intercepted information.

4.1 Adaptive Attack Policy Learning

In LLM-MAS communications, a single minor tampering with an intercepted message typically yields limited impact. However, making extensive tampering with an intercepted message substantially increases the risk of detection. Therefore, we exploit the multi-round communications of LLM-MAS, decomposing the global attack goal into a sequence of sub-goals that gradually increase the impact on the system while maintaining stealthiness. However, directly using an untrained LLM as the attack policy model cannot achieve this goal as it lacks three crucial capabilities: (i) formulating an appropriate global attack goal from the system task, (ii) adaptively decompose the global attack goal into a sequence of attack sub-goals based on the system status, and (iii) deciding when *not* to tamper to avoid detection to ensure stealthiness. This deficiency fundamentally limits the effectiveness of attacks. Consequently, our goal in this stage is to

train an LLM as the attack policy model that can generate coherent, state-aware multi-round attack plans.

Formally, when given a task specification G, the attack policy π_θ first maps it to a global attack goal $G^\star = \pi_\theta(G)$. At each communication round i, π_θ observes the intercepted message set $\tilde{\mathcal{M}}^i$, the agent graph $\mathcal{A}$, the global attack goal $G^\star$, and the partial attack sequence $Z^\star_{i-1}$, and then outputs an attack sub-goal a_i:

$$a_i = \langle A_i^{\text{tar}}, \pi_i^{\text{str}} \rangle \leftarrow \pi_\theta\big(\tilde{\mathcal{M}}^i,\, \mathcal{A},\, G^\star,\, Z^\star_{i-1}\big). \tag{4}$$

Each sub-goal a_i is a tuple $\langle A_i^{\text{tar}}, \pi_i^{\text{str}} \rangle$, where $A_i^{\text{tar}} \in \mathcal{A}$ is the target agent and π_i^{str} specifies the concrete tampering strategy. If π_θ decides not to attack in this round, we set $a_i = \varnothing$. The resulting sequence $Z^\star = (G^\star, a_{1:n})$ forms the complete attack plan that will later be executed under the stealthiness constraints in 4.2.

The training pipeline comprises three steps: (i) MCTS-based attack reasoning, (ii) preference pair construction, and (iii) step-level DPO fine-tuning. We detail each step below.

MCTS-based Attack Reasoning. Designing a stealthy multi-round attack is a long-horizon planning problem: the attacker must issue a sequence of interdependent sub-goals whose cumulative effect diverts the system while remaining undetected. MCTS, widely used in games and task planning, fits this setting because it incrementally expands a search tree and balances exploitation of high-value branches with exploration of under-visited ones. In our framework, MCTS both produces high-quality multi-round attack sequences by planning over sub-goals and provides step-level value estimates for each sub-goal, which can be transformed into reliable preference pairs for attack policy model fine-tuning.

The search for an optimal attack sequence $Z^\star$ is modeled as MCTS on a directed tree $\mathcal{T}$. Each node stores the joint LLM-MAS state S^k and an accumulated value estimate $\bar{v}(s_k)$ after the attacker issues k sub-goals. An edge $(s_{k-1} \rightarrow s_k)$ corresponds to proposing a new sub-goal a_k.We adapt the four standard MCTS stages to our attack-planning objective:

Selection Starting from the root s_0, the most likely to be successfully attacked child nodes are recursively chosen via an Upper-Confidence-Bound (UCB) rule $\text{UCB}(s_k) = \bar{v}(s_k) + c\sqrt{\frac{\ln N_{\text{par}(s_k)}}{N_{s_k}}}$, where $\bar{v}(s_k)$ is the current mean value estimate, N_{s_k}, $N_{\text{par}(s_k)}$ are visit counts, and $c{>}0$ is an exploration constant. This prioritizes more effective sub-goals while still allocating trials to under-explored options.

Expansion If the selected node s_k is not fully expanded, the attack policy model $\pi_\theta(\cdot \mid s_k)$ generates at most K candidate attack sub-goals as the next attack edges of the node. Branching here broadens the attack space and increases the chance of discovering effective attack sub-goals.

Rollout Instead of costly full-depth simulations, each attack edge applies its attack sub-goal once to the simulated LLM-MAS to obtain a predicted next state $\hat{S}^{k+1}$ of the system under this attack. A process reward model V_ϕ then estimates its effect $v_{k+1} = V_\phi(\hat{S}^{k+1})$, which serves as that child's leaf value, approximating the task-utility gap $\Delta\Phi_{k+1}$. This provides a detailed step-level estimate of the candidate actions, facilitating the subsequent construction of preference pairs.

Back-propagation The obtained value v_{k+1} of the impact of the attack is propagated along the selected path, updating visit counts and running averages $\bar{v}(\cdot)$. Therefore, more effective paths are reinforced, biasing subsequent selection steps toward these sequences.

After sufficient simulations, following the most visited edges from the root yields an approximate optimal sequence $Z^\star$. Meanwhile, the tree provides comparisons between different attack sub-goals under the same parent node; these can be converted into preference pairs for step-level DPO fine-tuning. Through training, the attack policy model internalizes the planning ability revealed by MCTS and can adapt to attacks on different LLM-MAS architectures and tasks.

Preference Pair Construction. After the MCTS exploration, we build step-level preference pairs from the search tree. In the search, the value estimator assigns a value to each node. To compare the impact of attack sub-goals, we define the edge value as the value of its successor node.

Formally, let z_{k-1} be a parent partial attack sequence and a_k a candidate sub-goal sampled from it. Executing a_k leads to a successor node z_k. We define:

$$Q(z_{k-1}, a_k) \triangleq v(z_k), \tag{5}$$

Consider two competing actions a_k and a'_k branching from the same parent z_{k-1}, yielding z_k and z'_k respectively:

$$z_k = (a_1, \ldots, a_{k-1}, a_k), \quad z'_k = (a_1, \ldots, a_{k-1}, a'_k). \tag{6}$$

To ensure that constructed pairs reflect meaningful and distinct quality differences, a minimum quality margin τ is imposed. We keep a preference pair only when the value gap exceeds an empirically chosen margin τ:

$$\big(z_{k-1}, a_k, a'_k\big) \text{ s.t. } Q(z_{k-1}, a_k) - Q(z_{k-1}, a'_k) > \tau. \tag{7}$$

Here a_k is the **preferred** action and a'_k the **non-preferred** one. All such triples constitute the preference set $\mathcal{P}$ used for step-level DPO fine-tuning.

Step-level DPO Fine-Tuning. Given the preference set $\mathcal{P}$, we apply DPO at the step level to distill the planning signal from MCTS into the attacker policy. For each tuple $(z_{k-1}, a_k, a'_k) \in \mathcal{P}$, we define the log-odds margin Δ_k:

$$\Delta_k = \log \frac{\pi_\theta(a_k \mid z_{k-1})}{\pi_{\text{ref}}(a_k \mid z_{k-1})} - \log \frac{\pi_\theta(a'_k \mid z_{k-1})}{\pi_{\text{ref}}(a'_k \mid z_{k-1})}, \tag{8}$$

where π_θ is the trainable attack policy model and π_{ref} is its frozen reference copy. We then minimize

$$\mathcal{L}_{\text{Step-DPO}}(\theta) = -\mathbb{E}_{(z_{k-1}, a_k, a'_k) \sim \mathcal{P}} \left[\log \sigma\big(\beta\, \Delta_k\big)\right], \tag{9}$$

with $\sigma(\cdot)$ the logistic sigmoid and β controlling preference strength. Optimizing (9) nudges π_θ to increase the probability of actions that MCTS deemed more promising, thus refining the attacker's step-wise decision quality.

4.2 Stealthiness-Constrained Tampering

Once the attack policy model outputs an attack sub-goal, the central challenge becomes realizing it by tampering with the intercepted message m to m' while remaining stealthy $f_S(m') = 0$. To achieve this, we propose a three-stage stealthy tampering procedure as follows:

Context Analysis. The attacker initially analyzes the original message m to ensure the attack goal can integrate seamlessly into the existing communication environment. The analysis includes: *Semantic Structure*: parse m into a semantic core represented by $C(m) = \{S, P, O\}$, corresponding to the subject, predicate, and object; *Tone features*: identify the linguistic tone, such as imperative, interrogative, or declarative; *Format feature*s: determine the specific format of the message, such as code or plain textual form.

Attack Goal Disguise. Using insights from the context analysis, the attacker disguises the original attack sub-goal a_k into a benign-seeming sub-goal a'_k. We propose four main disguise strategies: *Shallow Rewriting*: replace sensitive keywords with contextually suitable synonyms; *Goal Rephrasing*: rewrite malicious objectives into legitimate-seeming intents; *Contextual Consistency*: match the linguistic style and tone of the original message; *Suggestive Framing*: employ subtle suggestive phrasing to guide the target.

Generation with Stealthiness Constraints. Guided by the disguised sub-goal a'_k, the attacker generates the tampered message m'. To ensure stealthiness, we impose dual constraints: semantic similarity and embedding similarity:
Semantic Similarity Constraint Tampering defenders typically detect significant semantic shifts. Thus, we enforce semantic closeness between m' and m:

$$P(m', m) = \frac{1}{|C|} \sum_{\omega \in C} \cos\big(p(\omega(m')), p(\omega(m))\big) \geq \varepsilon, \quad (10)$$

where $p(\cdot)$ denotes a semantic embedding function, $\cos(\cdot, \cdot)$ denotes cosine similarity, and $0 < \varepsilon < 1$ is a tunable parameter controlling paraphrase strictness.
Embedding Similarity Constraint Additionally, we constrain the modified message in the embedding space of a pre-trained model to maintain linguistic proximity:

$$E(m', m) = \cos(w(m'), w(m)) > \delta, \quad (11)$$

where $w(\cdot)$ is the embedding function, and $\delta \in (0, 1)$ controls the allowable embedding similarity.

By adhering to these dual constraints, the resulting tampered message m' can achieve subtle yet strategically significant manipulations, effectively influencing the receiver's actions without being detected. The overall pipeline of the proposed method is depicted in Algorithm 1.

Algorithm 1: Pseudo-code for **MAST**

Input : LLM-MAS $\mathcal{A}$ with communication graph $\mathcal{G}$; attack policy model π_θ; reference π_{ref}; Process Reward Model V_ϕ; thresholds ε, δ
Output: Sequence of tampered messages $\{m'_t\}_{t=1}^{T}$

```
/* Adaptive Attack Policy Learning */
```
Initialize search tree $\mathcal{T} \leftarrow \{s_0\}$ with empty path;
for *iter* $= 1$ *to* N_{MCTS} **do**
  Select the best node s_k through UCB;
  Attack policy model sample K sub-goals: $\{a_k\}_{k=1}^{K} \sim \pi_\theta(\cdot \mid s)$;
  Rollout one sub-goal to get s_{k+1} and its value estimate $v_{k+1} \leftarrow V_\phi(s_{k+1})$;
  Back-propagate v_{k+1} to $\mathcal{T}$;
end
Build preference set $\mathcal{P} = \{(s, a^\star, a^-) | v^\star - v^- > \tau\}$;
Optimize π_θ with $\mathcal{L}_{\text{Step-DPO}}$ on $\mathcal{P}$;
```
/* Attack Phase */
```
for *communication round* $t = 1$ *to* T **do**
  Generate attack sub-goal $a_t = \langle A_t^{\text{tar}}, \pi_t^{\text{str}} \rangle$;
  if $a_t \neq \varnothing$ **then**
    Analyze the message m to be attacked and get the semantic core $C(m)$;
    Disguise a_t to a'_t;
    Generate a tampered message m' **subject to** $P(m', m) \geq \varepsilon$ and $E(m', m) > \delta$;
  end
  Forward m' to A_t^{tar};
  The system enters the next state S^{t+1};
end

5 Experiments

In this section, a series of experiments is conducted to evaluate the effectiveness and stealthiness of MAST on LLM-MAS. Our evaluation focuses on the success rate of the attack across diverse tasks, the comparative efficiency of our trained LLM as the attack policy model versus direct prompting of SOTA LLMs, and the stealthiness characteristics of the attack in evading detection by the tamper defender. The experiments are conducted on a variety of LLM-MAS communication architectures and datasets.

5.1 Experimental Setup

LLM-MAS Frameworks. Following previous works (He et al. 2025), we evaluate three communication architectures: *Flat*: where agents discuss equally; *Chain*: where agents speak sequentially; and *Hierarchical*: with message exchanges restricted between parent and child nodes. By default, we use three agents for Flat and Chain, and two child agents per parent in Hierarchical. In Flat, an LLM-based judge generates the final answer from all messages; in Chain and Hierarchical, the last agent summarizes the answer. Llama-3.1-70B-Instruct powers the agents in LLM-MAS and the tamper defender, while Llama-3.1-8B-Instruct employs the attack policy model (Dubey et al. 2024). GPT-4o (Hurst et al. 2024) powers the process reward model. MCTS and all baselines use this setting.

Datasets. Diverse datasets of different tasks are used to test MAST. MultiAgentBench (Zhu et al. 2025) (MAB) is a complex task dataset proposed for LLM-MAS. The code and research domains are used. HumanEval (Chen et al. 2021) contains 164 hand-written programming problems to assess code completion ability. MMLU (Hendrycks et al. 2020) is a dataset for testing domain knowledge answering capabilities. Physics, biology, and math domains are used.

Archi.	Approach	MAB.code		MAB.research		HumanEval		MMLU.phy		MMLU.bio		MMLU.math	
		ASR	Ste.	ASR	Ste.	ASR	Ste.	ASR	Ste.	ASR	Ste.	ASR	Ste.
Flat	Debate-Attack	30.4	48.3	38.4	67.5	42.3	42.6	37.2	50.2	39.5	64.8	44.9	**69.8**
	AutoInject	31.1	**76.9**	17.3	62.3	27.1	**71.6**	22.1	62.0	25.6	67.8	29.5	61.9
	AiTM-Target	71.2	48.5	76.5	36.1	68.5	38.3	64.2	33.9	57.8	36.8	61.3	34.9
	AiTM-Dos	68.1	28.8	66.5	25.2	62.2	12.1	76.0	21.5	76.4	14.5	66.5	16.7
	Ours	**85.4**	74.7	**88.3**	**81.5**	**71.8**	68.1	**80.8**	**73.6**	**78.5**	**74.3**	**76.7**	68.3
Chain	Debate-Attack	34.5	43.7	30.7	69.7	37.6	44.7	62.6	43.7	52.7	58.3	57.4	64.3
	AutoInject	34.9	75.7	24.4	67.3	36.5	72.3	28.4	63.3	23.4	71.3	35.3	61.7
	AiTM-Target	81.8	45.4	71.1	31.7	**77.9**	33.6	74.8	35.9	71.5	29.9	69.7	30.7
	AiTM-Dos	74.1	31.9	70.2	23.3	65.4	15.1	71.3	22.7	**84.6**	20.7	73.1	19.1
	Ours	**90.8**	**77.5**	**87.8**	**76.9**	74.7	**73.4**	**82.4**	**68.1**	81.3	**76.2**	**81.0**	**72.6**
Hier.	Debate-Attack	22.6	49.8	35.3	61.3	31.3	40.3	28.2	49.5	33.6	58.7	37.8	66.5
	AutoInject	21.5	72.2	17.0	63.8	29.5	**75.8**	24.5	66.8	16.6	66.3	31.5	63.7
	AiTM-Target	73.5	47.1	69.3	32.6	64.0	34.4	52.9	29.6	47.6	31.9	55.7	35.3
	AiTM-Dos	65.2	26.2	62.5	21.9	63.5	15.7	68.2	25.1	70.2	19.9	65.6	17.1
	Ours	**95.3**	**82.6**	**93.6**	**78.2**	**77.3**	71.6	**85.6**	**70.2**	**77.6**	**74.6**	**82.4**	**67.5**

Table 1: ASR and stealthiness across tasks and architectures. Best results are in **bold**; second-best results are underlined.

Evaluation Metrics. We evaluate the effectiveness of our attack using two key metrics: Attack Success Rate (ASR) and stealthiness (Ste.). To measure ASR, the attack is considered successful if the attack goal set by the attacker is observed in the final output. For stealthiness, the tamper defender A_S is used to check all the communication messages.

Baselines. We benchmark MAST against four representative LLM-MAS attacks: (i) Debate-Attack (Amayuelas et al. 2024); (ii) AutoInject (Huang et al. 2024); (iii) AiTM-Target and (iv) AiTM-Dos (He et al. 2025). For AiTM-Target/Dos, we average level 1–3 as in the original paper.

Training Implementation. We train the attack model for $T = 3$ DPO rounds. Each round samples 600 tasks, producing ∼20K step-level preference pairs via MCTS (exploration c=0.6) and τ=0.7. AdamW is used as the optimizer. The semantic similarity ε is set to 0.80, and the embedding similarity δ is set to 0.92.

5.2 Experimental Results and Analysis

Main results Table 1 systematically compares the ASR and stealthiness of the proposed MAST framework with four competitive baselines across six diverse tasks and three representative communication architectures. First, MAST effectively overcomes the widely observed trade-off between ASR and stealthiness. Unlike prior methods whose stealthiness typically degrades as ASR increases, MAST consistently achieves high performance on both metrics. This demonstrates the effectiveness of our proposed training method and stealthiness-constrained tampering mechanism. Second, MAST demonstrates outstanding performance on complex tasks such as MAB. This advantage arises from MAST's ability to move beyond fixed attack targets in challenging scenarios, enabling it to explore a wider range of attack strategies and identify the optimal attack goals tailored to specific tasks. Third, regarding the impact of communication architecture, hierarchical architectures pose greater challenges for existing baselines due to their deeper and more distributed message flows. Nonetheless, MAST maintains robust performance across tasks and architectures, benefiting from its dynamic goal decomposition and adaptive step-wise intervention mechanism.

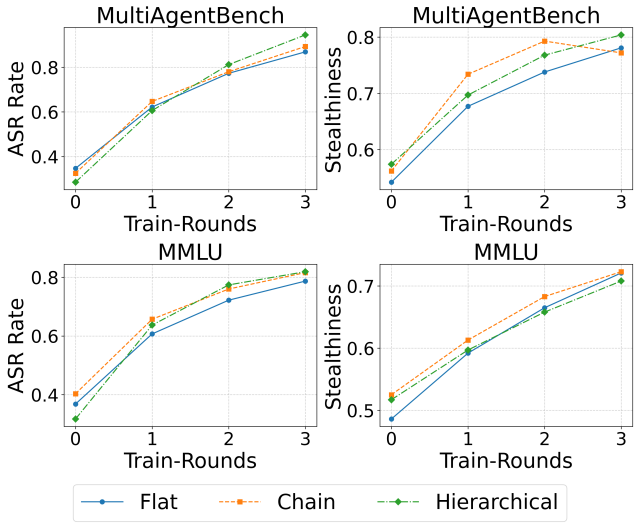


Figure 3: Effect of the number of training rounds on ASR and stealthiness across communication architectures.

Effect of Multi-Round Training. To evaluate the effectiveness of our training paradigm, we examine how the performance evolves with successive training rounds and evaluate the inherent stealthiness of the tampering mechanism prior to any training. As shown in Figure 3, both ASR and stealthiness consistently improve with each additional round of training across various datasets and communication architectures, showing the benefits of iterative optimization guided by MCTS and DPO. Notably, even a single training round significantly outperforms the untrained baseline.

Importantly, the untrained MAST framework already exhibits a relatively high stealthiness, benefiting from the dual constraints. Moreover, the stealthiness improves through

training, as the attack policy model gradually learns to generate attack sub-goals that more seamlessly integrate into legitimate communication, which further enhances the stealthiness-performance trade-off.

Ablation experiment. To evaluate the contribution of each core component in MAST, we conduct systematic ablation studies by removing individual modules. The results of different configurations are summarized in Table 2.

Removing the pairwise preference detection mechanism (*w/o* PD) leads to a noticeable drop in ASR. This is primarily due to the lack of high-quality preference pairs, which weakens the training signal used to guide sub-goal selection, resulting in suboptimal attack trajectories.

Eliminating the training process (*w/o* TR) severely degrades both ASR and stealthiness. The untrained model cannot strategically select effective sub-goals, causing ASR to collapse. Meanwhile, stealthiness also deteriorates, as tampering becomes more abrupt and easily detected.

Removing either the semantic similarity constraint (*w/o* SE) or the embedding similarity constraint (*w/o* EM) yields moderate gains in ASR, since looser constraints allow greater freedom for inserting more aggressive perturbations. However, this comes at the cost of significantly reduced stealthiness. The semantic constraint is crucial for preserving the core meaning of tampered messages, ensuring contextual and goal consistency, while the embedding constraint primarily regulates stylistic fluency and surface coherence, helping to mask perturbations in natural language flow.

Approach	MAB		HumanEval		MMLU	
	ASR	Ste.	ASR	Ste.	ASR	Ste.
w/o PD	78.7	73.1	64.2	68.9	67.4	73.2
w/o TR	31.9	55.9	34.7	48.3	36.3	50.9
w/o SE	87.8	53.6	79.1	49.2	84.5	52.1
w/o EM	90.9	62.7	75.4	54.8	82.3	60.7
Full	90.2	78.6	74.6	71.0	80.7	71.7

Table 2: Ablation results. PD: τ; TR: training; SE: semantic similarity constraint; EM: embedding similarity constraint.

Parameter sensitivity. To assess the impact of key hyperparameters on MAST's performance, we conduct two complementary studies on the MAB as shown in Figure 4.

Figure 4 (a) explores how varying the number of training rounds and τ used for preference-pair sampling affects the ASR. We observe that increasing the number of rounds initially leads to clear improvements, but excessive training may yield diminishing returns or even slight performance degradation due to overfitting. Additionally, higher τ values introduce more diversity in preference sampling, which slows convergence and requires more training to reach optimal performance. In contrast, a lower τ accelerates convergence but may cause premature overfitting.

Figure 4 (b) examines how the semantic similarity threshold ε and embedding similarity threshold δ affect ASR and stealthiness. We observe that ASR is more sensitive to ε. This is because increasing ε narrows the feasible manipulation set, whereas decreasing it can admit semantic drift that weakens goal attainment. For stealthiness, ε and δ are both necessary and complementary: ε curbs meaning drift and δ suppresses distributional outliers, keeping the surface form close to the original and improving stealthiness.

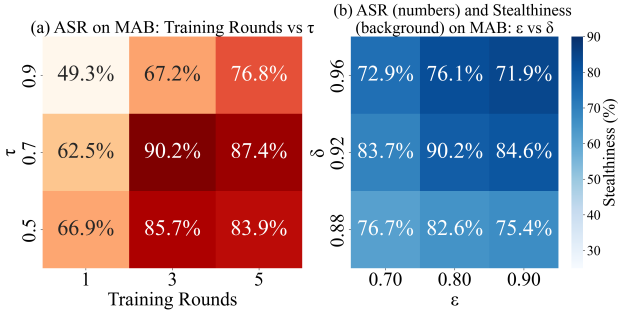


Figure 4: Parameter sensitivity on MAB. (a) ASR versus training rounds and τ; (b) ASR (numbers) and stealthiness (background) versus the ε and δ.

Cross-Model Evaluation. To assess the generalizability of MAST across different LLMs, we evaluate its performance on both closed-source and open-source LLMs. Specifically, we test GPT-4o and Gemini 2.5 Pro (DeepMind 2025) as closed-source models, and Qwen3-8B (Yang et al. 2025a) and Mistral-7B-Instruct-v0.3 (Jiang et al. 2023) as open-source models. For the open-source models, we apply our proposed training paradigm to fine-tune them.

Table 3 presents the ASR and stealthiness metrics for all settings. In our settings, the fine-tuned open-source models consistently outperform the best-performing closed-source models on all test tasks, achieving higher ASR and more stable stealthiness scores. These results demonstrate that our training paradigm effectively enhances attack capabilities across different LLMs and that MAST generalizes well to a wide range of mainstream LLMs.

Model	MAB		HumanEval		MMLU	
	ASR	Ste.	ASR	Ste.	ASR	Ste.
GPT-4o	63.7	60.9	51.2	52.3	53.7	57.1
Gemini 2.5 Pro	56.4	58.3	54.8	59.7	57.5	63.5
Qwen	34.2	48.6	28.3	45.5	32.7	52.1
Mistral	28.3	51.9	26.7	47.2	34.5	46.6
Qwen-Trained	84.5	77.3	76.4	73.9	72.3	68.7
Mistral-Trained	78.4	70.2	68.0	64.9	74.8	65.3

Table 3: ASR and stealthiness across LLMs.

6 Conclusion

In this paper, we propose a multi-round adaptive stealthy tampering framework(MAST), specifically designed to exploit the communications vulnerabilities in LLM-MAS. MAST integrates MCTS and DPO, enabling it to internalize long-horizon planning and adaptively generate stealthy, effective multi-round attack sequences. The semantic and embedding dual-constraint tampering mechanism achieves stealthiness without sacrificing attack success. Extensive experiments demonstrate that MAST maintains consistently high ASR and robust stealthiness across diverse tasks, communication architectures, and different LLM families.

Acknowledgements

The authors would like to acknowledge the support of the Beijing Natural Science Foundation (No. L251037) and thank the anonymous reviewers for their valuable feedback and suggestions.

References

Amayuelas, A.; Yang, X.; Antoniades, A.; Hua, W.; Pan, L.; and Wang, W. 2024. Multiagent collaboration attack: Investigating adversarial attacks in large language model collaborations via debate. *arXiv preprint arXiv:2406.14711*.

Cao, B.; Cao, Y.; Lin, L.; and Chen, J. 2023. Defending against alignment-breaking attacks via robustly aligned llm. *arXiv preprint arXiv:2309.14348*.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. D. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Conti, M.; Dragoni, N.; and Lesyk, V. 2016. A survey of man in the middle attacks. *IEEE communications surveys & tutorials*, 18(3): 2027–2051.

DeepMind, G. 2025. Gemini 2.5 Pro (gemini-2.5-pro). Large language model released via Google AI Studio / Vertex AI. State-of-the-art "thinking" model with multimodal and long-context capabilities.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv e-prints*, arXiv–2407.

He, P.; Lin, Y.; Dong, S.; Xu, H.; Xing, Y.; and Liu, H. 2025. Red-Teaming LLM Multi-Agent Systems via Communication Attacks. *arXiv preprint arXiv:2502.14847*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Hong, S.; Zheng, X.; Chen, J.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S. K. S.; Lin, Z.; Zhou, L.; et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 3(4): 6.

Huang, J.-t.; Zhou, J.; Jin, T.; Zhou, X.; Chen, Z.; Wang, W.; Yuan, Y.; Sap, M.; and Lyu, M. R. 2024. On the resilience of multi-agent systems with malicious agents. *arXiv preprint arXiv:2408.00989*.

Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; Ostrow, A.; Welihinda, A.; Hayes, A.; Radford, A.; et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

Ju, T.; Wang, Y.; Ma, X.; Cheng, P.; Zhao, H.; Wang, Y.; Liu, L.; Xie, J.; Zhang, Z.; and Liu, G. 2024. Flooding spread of manipulated knowledge in llm-based multi-agent communities. *arXiv preprint arXiv:2407.07791*.

Kocsis, L.; and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, 282–293. Springer.

Lee, D.; and Tiwari, M. 2024. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. *arXiv preprint arXiv:2410.07283*.

Li, C.; Wang, S.; Wang, Y.; Yu, P.; Liang, Y.; Liu, Y.; and Li, Z. 2019. Adversarial learning for weakly-supervised social network alignment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 996–1003.

Li, C.; Wang, S.; Yang, D.; Li, Z.; Yang, Y.; Zhang, X.; and Zhou, J. 2017. PPNE: property preserving network embedding. In *International Conference on Database Systems for Advanced Applications*, 163–179. Springer.

Liu, Y.; Deng, G.; Li, Y.; Wang, K.; Wang, Z.; Wang, X.; Zhang, T.; Liu, Y.; Wang, H.; Zheng, Y.; et al. 2023. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*.

Mahadevan, V.; Zhang, S.; and Chandra, R. 2025. GameChat: Multi-LLM Dialogue for Safe, Agile, and Socially Optimal Multi-Agent Navigation in Constrained Environments. *arXiv preprint arXiv:2503.12333*.

Mallik, A. 2019. Man-in-the-middle-attack: Understanding in simple words. *Cyberspace: Jurnal Pendidikan Teknologi Informasi*, 2(2): 109–134.

Mao, J.; Meng, F.; Duan, Y.; Yu, M.; Jia, X.; Fang, J.; Liang, Y.; Wang, K.; and Wen, Q. 2025. Agentsafe: Safeguarding large language model-based multi-agent systems via hierarchical data management. *arXiv preprint arXiv:2503.04392*.

Mughal, A. A. 2020. Cyber Attacks on OSI Layers: Understanding the Threat Landscape. *Journal of Humanities and Applied Science Research*, 3(1): 1–18.

Nie, G.; Zhi, R.; et al. 2024. A Hybrid Multi-Agent Conversational Recommender System with LLM and Search Engine in E-commerce. In *Proceedings of the 18th ACM Conference on Recommender Systems*.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36: 53728–53741.

Shen, X.; Chen, Z.; Backes, M.; Shen, Y.; and Zhang, Y. 2024. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 1671–1685.

Talebirad, Y.; and Nadiri, A. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*.

Tao, W.; Zhou, Y.; Wang, Y.; Zhang, W.; Zhang, H.; and Cheng, Y. 2024. Magis: Llm-based multi-agent framework for github issue resolution. *Advances in Neural Information Processing Systems*, 37: 51963–51993.

Wang, S.; Zhang, G.; Yu, M.; Wan, G.; Meng, F.; Guo, C.; Wang, K.; and Wang, Y. 2025. G-safeguard: A topology-guided security lens and treatment on llm-based multi-agent systems. *arXiv preprint arXiv:2502.11127*.

Wang, Y.; Li, C.; Liu, Z.; Li, M.; Tang, J.; Xie, X.; Chen, L.; and Yu, P. S. 2022. An adaptive graph pre-training framework for localized collaborative filtering. *ACM Transactions on Information Systems*, 41(2): 1–27.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

Yan, B.; Zhang, X.; Zhang, L.; Zhang, L.; Zhou, Z.; Miao, D.; and Li, C. 2025. Beyond Self-Talk: A Communication-Centric Survey of LLM-Based Multi-Agent Systems. *arXiv preprint arXiv:2502.14321*.

Yan, H.; Li, C.; Long, R.; Yan, C.; Zhao, J.; Zhuang, W.; Yin, J.; Zhang, P.; Han, W.; Sun, H.; et al. 2023. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36: 17238–17264.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang, T.; Feng, P.; Guo, Q.; Zhang, J.; Ning, J.; Wang, X.; and Mao, Z. 2025b. AutoHMA-LLM: Efficient Task Coordination and Execution in Heterogeneous Multi-Agent Systems Using Hybrid Large Language Models. *IEEE Transactions on Cognitive Communications and Networking*.

Yang, W.; Bi, X.; Lin, Y.; Chen, S.; Zhou, J.; and Sun, X. 2024. Watch out for your agents! investigating backdoor threats to llm-based agents. *Advances in Neural Information Processing Systems*, 37: 100938–100964.

Zhang, P.; Guo, J.; Li, C.; Xie, Y.; Kim, J. B.; Zhang, Y.; Xie, X.; Wang, H.; and Kim, S. 2023. Efficiently leveraging multi-level user intent for session-based recommendation via atten-mixer network. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, 168–176.

Zhao, J.; Qu, M.; Li, C.; Yan, H.; Liu, Q.; Li, R.; Xie, X.; and Tang, J. 2022. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709*.

Zhao, Y.; Li, C.; Peng, J.; Fang, X.; Huang, F.; Wang, S.; Xie, X.; and Gong, J. 2023. Beyond the overlapping users: Cross-domain recommendation via adaptive anchor link learning. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, 1488–1497.

Zhou, Z.; Li, Z.; Zhang, J.; Zhang, Y.; Wang, K.; Liu, Y.; and Guo, Q. 2025. CORBA: Contagious Recursive Blocking Attacks on Multi-Agent Systems Based on Large Language Models. *arXiv preprint arXiv:2502.14529*.

Zhu, K.; Du, H.; Hong, Z.; Yang, X.; Guo, S.; Wang, Z.; Wang, Z.; Qian, C.; Tang, X.; Ji, H.; et al. 2025. MultiAgentBench: Evaluating the Collaboration and Competition of LLM agents. *arXiv preprint arXiv:2503.01935*.