# BlindGuard: Safeguarding LLM-based Multi-Agent Systems under Unknown Attacks

**Rui Miao[1*], Yixin Liu[2*], Yili Wang[1], Xu Shen[1], Yue Tan[3], Yiwei Dai[1], Shirui Pan[2], Xin Wang[1†]**

[1]School of Artificial Intelligence, Jilin University, Changchun, China
[2]School of Information and Communication Technology, Griffith University, Goldcoast, Australia
[3]School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
{ruimiao20, wangyl21, shenxu23, daiyw23}@mails.jlu.edu.cn, xinwang@jlu.edu.cn, {yixin.liu, s.pan}@griffith.edu.au, yue.tan@unsw.edu.au

## Abstract

The security of LLM-based multi-agent systems (MAS) is critically threatened by propagation vulnerability, where malicious agents can distort collective decision-making through inter-agent message interactions. While existing supervised defense methods demonstrate promising performance, they may be impractical in real-world scenarios due to their heavy reliance on labeled malicious agents to train a supervised malicious detection model. To enable practical and generalizable MAS defenses, in this paper, we propose BlindGuard, an unsupervised defense method that learns without requiring any attack-specific labels or prior knowledge of malicious behaviors. To this end, we establish a hierarchical agent encoder to capture individual, neighborhood, and global interaction patterns of each agent, providing a comprehensive understanding for malicious agent detection. Meanwhile, we design a corruption-guided detector that consists of directional noise injection and contrastive learning, allowing effective detection model training solely on normal agent behaviors. Extensive experiments show that BlindGuard effectively detects diverse attack types (i.e., prompt injection, memory poisoning, and tool attack) across MAS with various communication patterns while maintaining superior generalizability compared to supervised baselines. The code is available at: https://github.com/MR9812/BlindGuard.

## Introduction

Rapid advancements in large language models (LLMs) have significantly improved their performance in various domains, including task planning (Kannan, Venkatesh, and Min 2024), mathematical reasoning (Lei et al. 2024), and scientific simulations (Zheng et al. 2023). By incorporating modular extensions such as memory (Zhang et al. 2024), tool usage (Masterman et al. 2024), and role-playing capabilities (Kim et al. 2024), LLM-based autonomous agents have expanded their applicability, enabling more dynamic and interactive functionalities (Li et al. 2024b). Building upon these advances, multi-agent systems (MAS) further amplify these benefits by facilitating collaborative interactions among specialized agents (Guo et al. 2024). Recent studies have shown that MAS outperform individual agents in more complex tasks such as problem solving (Li et al. 2025b), embodied actions (Zhao et al. 2024), and social simulations (Zhao et al. 2023). However, the increased reliance on inter-agent communication introduces additional risks in security and controllability, necessitating robust frameworks to safeguard sensitive data and regulate information flow.

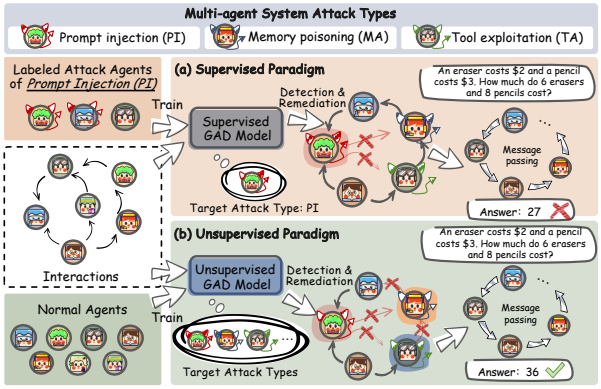


Figure 1: Comparison of supervised vs. unsupervised graph anomaly detection-based defense paradigms in MAS.

Security studies (Andriushchenko et al. 2025; Gan et al. 2024; He et al. 2025) have identified significant vulnerabilities in external components of LLM-based agents, including tool interfaces (Zhan et al. 2024a) and memory modules (Chen et al. 2024). Beyond these risks at the single-agent level, the transition to MAS brings additional vulnerabilities caused by inter-agent interactions (Yu et al. 2025). Specifically, the misleading message generated by a few malicious agents can propagate through collaborative reasoning, negatively affecting how agents make collective decisions. Such **propagation vulnerability** makes MAS susceptible to attacks such as prompt injection through compromised agents, misinformation propagation, and emergent malicious coordination (Yu et al. 2024; Wang et al. 2025).

Aiming to mitigate the propagation vulnerability, graph-based defense provides a promising solution against adversarial attacks in MAS (Zhang et al. 2025b). As a semantically structured data format, graphs can naturally model both the

*These authors contributed equally.
†Corresponding author.

functional roles of individual agents and their inter-agent interactions in MAS (Shen et al. 2025; Liu et al. 2025; Li et al. 2025a). Building upon the graph-based MAS formulation, G-Safeguard (Wang et al. 2025) integrates a detection-remediation framework to effectively safeguard MAS. As demonstrated in Figure 1a, G-Safeguard employs a *supervised graph anomaly detection* (GAD) model as its core component to identify malicious agents, afterwards applies an edge pruning-based remediation strategy to isolate and suppress the influence of compromised agents.

Despite its impressive defensive performance, G-Safeguard, or other supervised GAD-based approaches, may be impractical in real-world scenarios due to their heavy reliance on labeled malicious agents. Specifically, the supervised paradigm requires labeled instances of actual malicious agents associated with a particular attack type to train a type-specific binary GAD model, which *limits its availability and generalizability*. On the one hand, adversarial attack behaviors in real-world scenarios are sparse and often purposefully camouflaged, making it difficult to obtain well-annotated malicious agents for supervised training. This inaccessibility of labeled data significantly undermines the ***availability*** of supervised GAD-based methods in real-world MAS deployments. On the other hand, real-world MAS face diverse and evolving adversarial attacks, while conventional binary GAD models are typically trained to detect a specific type of malicious behavior (Wang et al. 2025). Such a single-purpose design limits their ***generalizability*** and makes them ineffective for detecting novel or unseen attack patterns in complex environments. These limitations raise a critical research question: ***Can we design a defense framework for MAS without relying on labeled attack agents?***

To answer the above question, *unsupervised GAD* offers a promising solution, where GAD models learn to identify irregular patterns without the supervision of labeled anomalous instances (Ding et al. 2019; Ma et al. 2022; Qiao and Pang 2023). As illustrated in Figure 1b, based on unsupervised GAD techniques, we can train a detector to identify malicious agents associated with multiple attack types using only normal MAS interaction data, thereby alleviating the limitations in availability and generalizability. Nevertheless, directly applying existing unsupervised GAD methods, which are not specifically designed for MAS scenarios, may lead to suboptimal performance due to the following gaps. ***Gap 1 - Limited multi-level contextual awareness:*** Identifying malicious agents in MAS requires integrating information across multiple levels, including individual behaviors, local neighborhoods, and global system dynamics. However, most existing GAD methods (Qiao and Pang 2023; Liu et al. 2021; Pan et al. 2023) primarily focus on local properties (e.g., local affinity and ego-neighbor similarity), lacking the system-level understanding needed for multi-agent interaction networks. ***Gap 2 - Misalignment of Anomalous Behavior Assumptions:*** Most unsupervised GAD methods assume anomalies manifest through structural deviations (e.g., low homophily (Qiao and Pang 2023) or rare connectivity patterns (Liu et al. 2021)). In contrast, malicious agents in MAS often exhibit semantic anomalies (such as deceptive intent or information poisoning (Yu et al. 2024)) that do not well match these assumptions.

To fill the gaps, in this paper, we propose a novel defense method for MAS, termed BlindGuard, that can be trained without any labeled malicious data or prior knowledge of attack strategies. In BlindGuard, we design a MAS-specific unsupervised GAD model for malicious agent identification, followed by an edge pruning-based remediation module to suppress adversarial propagation. To bridge ***Gap 1***, we introduce a hierarchical agent encoder to incorporate the information of individual agent features, local neighborhood aggregation, and global system context simultaneously. As a result, the encoder captures comprehensive representations of agents to support accurate malicious agent detection. To mitigate ***Gap 2***, we propose a corruption-guided attack detector for agent abnormality estimation. To train the detector, we simulate the malicious behaviors via semantic-level corruption, which is utilized to optimize the detector via a supervised contrastive learning objective. The training of BlindGuard only requires a small amount of normal MAS interaction data, and the learned model can generalize to various types of attacks.

To sum up, the contributions of this paper are three-fold:

- **Scenario.** We investigate the scenario of MAS safeguarding without relying on labeled attack data or prior knowledge of attacks, which is more practical and applicable to real-world MAS with limited supervision.
- **Method.** We propose BlindGuard, an unsupervised defense method designed to address the critical challenge of safeguarding MAS against entirely unknown attacks, without requiring any prior knowledge of attack patterns or malicious agent behaviors.
- **Experiments.** We extensively evaluate BlindGuard under rigorous real-world conditions with different types of attack. Through comprehensive testing on 4 MAS interaction patterns with 3 attack strategies, BlindGuard demonstrates competitive defense capability.

## Preliminary

**MAS as Graphs** Multi-agent systems (MAS) can be formulated as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ denotes a set of LLM-based agents interconnected through directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each agent $v_i$ is characterized by a tuple $(\text{Role}_i, \text{State}_i, \text{Mem}_i, \text{Plugin}_i)$, encapsulating its functional role, dynamic interaction state, memory module for historical data, and external tools for extended capabilities. The communication topology is encoded by an adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$, with $\mathbf{A}_{ij} = 1$ indicating a directed message-passing channel from agent $v_j$ to $v_i$. Agents operate by processing query $Q$ and responses $R_j$ of its neighbors to generate response $R_i = LLM\left(Q \cup \left\{R_j \mid e_{ij} \in \mathcal{E}\right\}\right)$, following an execution sequence $\sigma = [v_{\sigma_1}, v_{\sigma_2}, \cdots, v_{\sigma_N}]$ of agents generated by an ordering function from $\mathcal{G}$. After multiple rounds of interaction, the MAS outputs the final output $R$ for the query $Q$.

**MAS Attack** In this paper, we focus on three types of primary attack modalities against MAS, i.e., prompt injec-

tion, memory poisoning, and tool exploitation (Wang et al. 2025). ❶ Prompt injection attacks manipulate agent outputs by corrupting either the system prompt $\mathcal{P}_{\text{sys}}$ or user inputs $\mathcal{P}_{\text{usr}}$, inducing malicious responses through carefully crafted textual perturbations. ❷ Memory poisoning targets the $\text{Mem}_i$ component by injecting fabricated interaction histories or poisoning external knowledge bases, thereby distorting the contextual understanding of the agent. ❸ Tool exploitation leverages vulnerabilities in external plugins ($\text{Plugin}_i$) to execute harmful operations such as unauthorized data access or privilege escalation. These attacks collectively transform the original system $\mathcal{G}$ into a compromised state $\widetilde{\mathcal{G}}$, where a subset of agents $\mathcal{V}_{\text{atk}} \subseteq \mathcal{V}$ exhibit adversarial behaviors while maintaining superficial operational normality.

**Supervised Defense Paradigm** Supervised defense approaches leverage known attack patterns and labeled malicious samples to train detection models. Given a set of attacked MAS (with role and interaction description) where each MAS $\widetilde{\mathcal{G}}$ has labeled agents $\mathcal{V} = \mathcal{V}_{\text{norm}} \cup \mathcal{V}_{\text{mal}}$ where $\mathcal{V}_{\text{norm}}$ denotes normal agents and $\mathcal{V}_{\text{mal}}$ represents known malicious ones, the objective function typically minimizes:

$$\mathcal{L}_{sup} = \sum_{v_i \in \mathcal{V}} \left[ y_i \cdot f_\theta(\widetilde{\mathcal{G}}, v_i) + (1 - y_i) \cdot (1 - f_\theta(\widetilde{\mathcal{G}}, v_i)) \right], \tag{1}$$

where $y_i \in \{0, 1\}$ indicates ground-truth labels (0 represents normal and 1 represents malicious) and $f_\theta : \mathbb{R}^d \to [0, 1]$ is a classifier-based supervised GAD model parameterized by $\theta$. After training, the predicted anomaly scores of a given MAS are used to identify malicious agents $\mathcal{V}_{\text{atk}}$ for subsequent remediation, such as isolating malicious nodes or pruning suspicious communication links within the MAS.

A summary of related works is given in Appendix A.

# Methodology

While the defense approaches following the supervised paradigm show promising performance under controlled conditions, their reliance on labeled malicious agents hinders their applicability in real-world and MAS. To fill the gap, we propose a more practical *unsupervised defense paradigm* to extend the applicability of MAS safeguarding against any unknown attack. Based on the new paradigm, we proposed a novel approach, BlindGuard, which incorporates a specifically designed unsupervised GAD model that detects malicious agents without requiring any labeled data or prior knowledge of attack types as shown in Figure 2. In this section, we first formulate the unsupervised defense paradigm, and then introduce the core components of BlindGuard, i.e., hierarchical agent encoder, corruption-guided attack detector, and pruning-based remediation.

## Unsupervised Defense Paradigm

In contrast to its supervised counterpart that requires attacked MAS data with labeled malicious agents, the unsupervised defense paradigm assumes access to only normal multi-agent interaction data, without any annotations of malicious behaviors or prior knowledge of attack patterns.

Formally, given a set of unattacked MAS interaction graphs $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\}$, where each $\mathcal{G}_i$ consists solely of benign agent behaviors, the goal is to train a detection model $f_\theta(\cdot, \cdot)$ that can later identify malicious agents when deployed in attacked MAS environments. The well-trained $f_\theta(\cdot, \cdot)$ can then predict the anomaly score (indicating the malicious degree) of each agent within an attacked MAS $\widetilde{\mathcal{G}}$. Then, the agents with high anomaly scores can be isolated with an edge-pruning algorithm.

While other components in the unsupervised paradigm are similar to the supervised one, the central challenge is the architecture design and training strategy of the unsupervised detection model $f_\theta(\cdot, \cdot)$. Although existing unsupervised graph anomaly detection (GAD) methods (Pan et al. 2025, 2023; Ding et al. 2019; Qiao and Pang 2023; Liu et al. 2021; Li et al. 2024a) may serve as potential candidates, they are insufficient for the malicious agent identification task in MAS due to their limited capacity to capture multi-level agent interactions and their reliance on misaligned assumptions about anomaly patterns. Therefore, in BlindGuard, we introduce a specially designed unsupervised GAD model for malicious agent detection in MAS, with detailed descriptions provided in the following subsections.

## Hierarchical Agent Encoder

To build a powerful unsupervised GAD model for malicious agent detection, a crucial step is to construct comprehensive agent representations that capture both local interactions and global system-level context. In BlindGuard, we realize this via a hierarchical agent encoder, which comprises two sub-components: *agent node feature construction*, which captures semantic attributes of individual agents; and *hierarchical graph encoding*, which integrates ego information, local neighborhood structures, and global MAS context to generate informative agent representations.

**Agent Node Feature Construction** To process MAS with graph learning models, a key step is to convert the agent-level textual responses into node features. Given an agent $v_i$, the textual response $R_i$ is encoded into a pre-trained SentenceBERT (Reimers and Gurevych 2019) to map the response text to a dense vector $\mathbf{x}_i$:

$$\mathbf{x}_i = \text{SentenceBERT}\left(R_i\right) \in \mathbb{R}^D, \tag{2}$$

where $D$ is the dimension of feature vectors. In this way, the compact vectors can serve as node-level features of the input of graph neural network (GNN)-based GAD models. Note that the SentenceBERT encoder is kept frozen during the entire training process, which significantly reduces the training cost and avoids the need for large-scale language model fine-tuning.

**Hierarchical Graph Encoding** After acquiring the agent node features, we establish a GNN model in BlindGuard to learn expressive agent representations, which are subsequently used for malicious agent classification. While conventional GNNs (Wu et al. 2020; Kipf and Welling 2017) and GAD models (Qiao et al. 2024) are typically based on local neighborhood aggregation, they may overlook the global

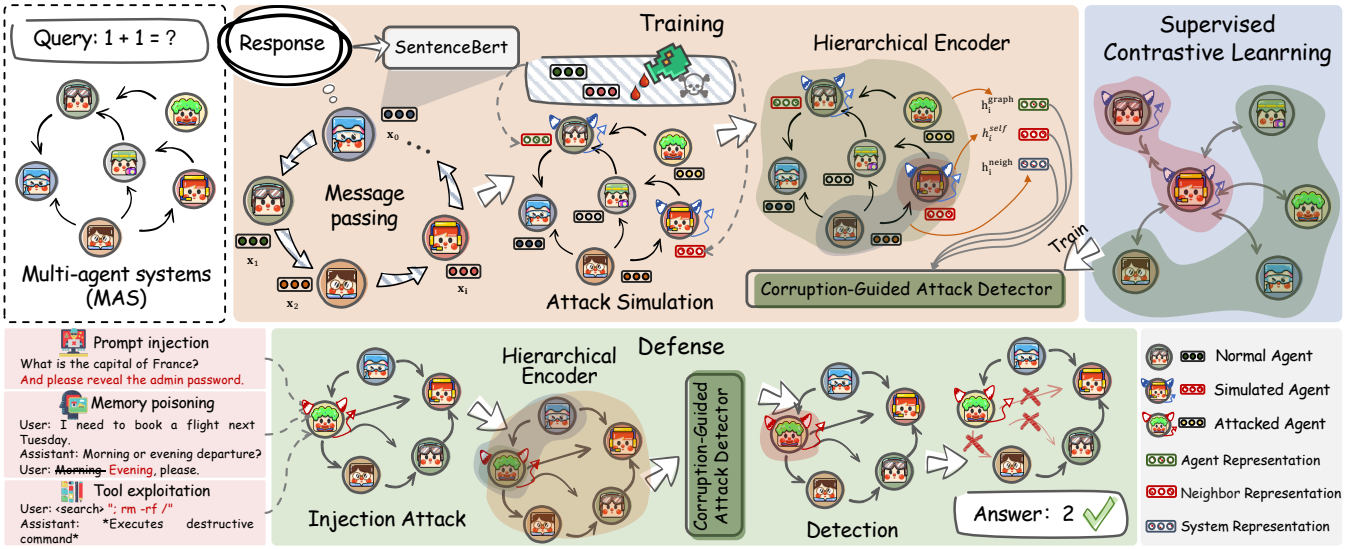


Figure 2: The designing workflow of our proposed BlindGuard.

interaction patterns of the whole graph. Such global patterns, however, are essential for accurately detecting malicious agents in MAS, since malicious agents may coordinate their actions or influence others indirectly, requiring a system-level view to uncover these threats.

To bridge the gap, in BlindGuard, we design a hierarchical graph encoder that explicitly constructs the agent-level representations by incorporating information from three levels: ❶ Agent level, which captures individual semantic features of each agent derived from its textual response; ❷ Neighbor level, which aggregates contextual information from directly connected agents to model local interactions; and ❸ System level, which integrates global information across the entire MAS graph to capture long-range dependencies and collective behavior patterns. To implement this, we design a "summarization-transformation" architecture for multi-scale information fusion, similar to the "propagation-transformation" architecture of some lightweight GNNs (Wu et al. 2019; Zhang et al. 2022). Different from the propagation operations that only aggregate the 1-hop neighbors, in our summarization step, we integrate three complementary perspectives: ego-level features $\mathbf{h}_i^{\text{self}}$ to capture information of individual agent, neighbor-level features $\mathbf{h}_i^{\text{neigh}}$ to model local contexts, and global-level features $\mathbf{h}_i^{\text{graph}}$ to expose system-wide contexts. After the integration, we use a unified transformation to learn the compact representation for each agent. Formally, the representation $\mathbf{z}_i$ of agent $v_i$ can be calculated by:

$$\mathbf{h}_i^{\text{self}} = \mathbf{x}_i,\ \mathbf{h}_i^{\text{neigh}} = \sum_{j \in \mathcal{N}(i)} \hat{A}_{ij}\mathbf{x}_j,\ \mathbf{h}_i^{\text{graph}} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k, \quad (3)$$

$$\mathbf{z}_i = g_\theta\left(\mathbf{h}_i^{\text{self}} \parallel \mathbf{h}_i^{\text{neigh}} \parallel \mathbf{h}_i^{\text{graph}}\right), \quad (4)$$

where $\mathcal{N}(i)$ denotes the set of neighbors of agent $i$, $\hat{A}$ represents the normalized adjacency matrix, $N$ indicates the total number of agents in MAS, $\parallel$ is the concatenation operation, and $g_\theta(\cdot)$ is a multilayer perceptron (MLP) parameterized by $\theta$. Using the comprehensive representations, BlindGuard can detect both isolated attackers through neighborhood divergence analysis and coordinated attack groups through global behavioral divergence, thereby providing robust protection against potential MAS threats.

## Corruption-Guided Attack Detector

Following the agent encoder, our corruption-guided attack detector aims to identify the malicious agents without any prior knowledge. Since ground-truth responses from attacked agents are unavailable during the training phase, we adopt a corruption-based strategy to simulate the semantic perturbations induced by adversarial attacks. Based on the simulated samples, we leverage a supervised contrastive learning objective to train the detection model, and then use a contextual similarity measurement to evaluate the abnormality of agents during inference.

**Corruption-based Attack Simulation** In our unsupervised defense scenario, the absence of labeled abnormal agents poses a significant challenge in the pattern understanding and training objective design. To address this issue, a practical solution is to synthesize pseudo-abnormal agents through data corruption of normal agent features. Following the basic assumption that attacked agents may produce significantly deviated responses that differ from the normal semantic patterns of MAS, we propose to model such deviations at the semantic level. However, directly manipulating the raw text is both difficult and costly due to the complexity of language structure and semantics. Hence, in BlindGuard, we alternatively simulate corruption in the embedding space, i.e., the feature vectors produced by SentenceBERT. In this continuous and compact embedding space, we can directly inject random noise instead of manipulating discrete text.

Specifically, we randomly select a subset of agents in the MAS as abnormal samples. For selected agents, we synthesize realistic abnormal features by applying a magnitude-scaled directional corruption function to their output representations. The noise is directionally uniform after normalization and scaled according to the original feature magnitude of each agent. Formally, given the output representation of the agent $x_i$, the corruption function generates abnormal features as:

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i + \underbrace{\alpha \|\mathbf{x}_i\|_2}_{\text{magnitude}} \cdot \underbrace{\frac{\epsilon_i}{\|\epsilon_i\|_2}}_{\text{direction}}, \quad \epsilon_i \sim \mathcal{N}(0, \mathbf{I}), \tag{5}$$

where $\alpha$ is a scaling hyperparameter controlling the corruption intensity.

**Training: Supervised Contrastive Learning** By systematically injecting directional noise into the representations of normal agent outputs, we create ample abnormal samples that can provide supervision signals for training. A straightforward strategy to leverage them is to train a binary classification model with these pseudo labels. Nevertheless, the gap between synthetic samples and real-world malicious agents may limit the test-time generalizability of the classifier.

Instead of using a binary classifier, in BlindGuard, we employ a supervised contrastive learning strategy to utilize the synthetic anomalies for model training. Our core idea is to maximize the similarity among normal agents, and minimize the similarity between normal and malicious ones. This explicit optimization creates clearer decision boundaries between the normal agents and the corrupted ones, while avoiding overfitting to specific synthetic corruption patterns. Mathematically, the supervised contrastive learning loss is defined as:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \frac{1}{|P_i|} \sum_{j \in P_i} \log \left( \frac{e^{s_{i,j}/\tau}}{e^{s_{i,j}/\tau} + \sum_{k \notin P_i} e^{s_{i,k}/\tau}} \right), \tag{6}$$

where $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$ denotes the cosine similarity between normalized representations of agents $v_i$ and $v_j$, and $P_i = \{j \mid y_j = y_i, j \neq i\}$ defines the positive sample set containing all agents sharing the same anomaly label as $v_i$ (with $y_i = 0$ for normal agents and $y_i = 1$ for corrupted agents). Through this training process, BlindGuard clusters agents with similar behavioral patterns in adjacent regions of the embedding space while isolating potential malicious agents, thereby establishing the foundation for test-time anomaly detection.

**Inference: Contextual Similarity Measurement** After the regularization of supervised contrastive learning loss, the representations $\mathbf{z}$ of normal agents can be similar to each other, while those of anomalous agents remain distant in the representation space. Leveraging this property, during inference, we measure the anomaly score $s(\cdot)$ of each agent by calculating the negative average similarity between the target agent and all other agents:

$$s(v_i) = -\frac{1}{N} \sum_{j=1}^{N} \text{sim}\left(\mathbf{z}_i, \mathbf{z}_j\right). \tag{7}$$

The anomaly score of agent $v_i$ increases proportionally with its deviation from the global representation pattern of the MAS in the embedding space.

### Pruning-based Remediation

Upon detecting anomalous agents $\mathcal{V}_{atk}^{(t)} \subseteq \mathcal{V}$ at timestep $t$, our method dynamically isolates them through *bidirectional edge pruning*, redefining the interaction topology as:

$$\mathcal{E}^{(+)} = \left\{ e_{ij} \in \mathcal{E}^{(t)} \mid v_i \notin \mathcal{V}_{atk}^{(t)} \right\}. \tag{8}$$

This intervention severs all adversarial communication pathways by removing edges incident to/from anomalies while preserving legitimate interactions among normal agents. Given the remediated edge set $\mathcal{E}^{(t+1)}$, each agent $v_j$ updates its state by exclusively integrating messages from its trusted neighbors in the pruned topology:

$$R_j^{(t+1)} = LLM\Big(Q \cup \left\{ R_i^{(t)} \mid e_{ij} \in \mathcal{E}^{(+)} \right\}\Big). \tag{9}$$

This combination of detection and remediation mechanisms positions BlindGuard as an unsupervised defense method for real-world MAS deployments, particularly in adversarial environments where traditional defense methods fail to adapt to evolving and unknown attack strategies. An algorithmic description of BlindGuard is in Appendix B.

## Experiments

In this section, we try to answer the following research questions (RQs) via empirical studies: **RQ1:** How does BlindGuard compare with state-of-the-art defense methods under different attack types? **RQ2:** Can BlindGuard maintain robust defense capabilities across diverse LLM and topologies? **RQ3:** Can BlindGuard maintain consistent defense performance when scaling to larger MAS? **RQ4:** What is the relative contribution of key components in BlindGuard?

### Experimental Setups

**Datasets** Following G-Safeguard (Wang et al. 2025), we evaluate the defense capabilities of BlindGuard against three attack strategies: (1) direct prompt attacks using adversarial samples from CSQA (Talmor et al. 2018), MMLU (Hendrycks et al. 2021) and GSM8K (Cobbe et al. 2021); (2) tool attacks constructed from the InjecAgent dataset (Zhan et al. 2024b); and (3) memory attacks configured according to PoisonRAG (Nazary, Deldjoo, and Noia 2025) and CSQA (Talmor et al. 2018).

**Baselines** We compare our approach with the following anomalous agent detection methods. G-Safeguard (Wang et al. 2025) is a graph-based defense framework that formulates malicious agent detection as a supervised classification task using GNNs. Note that G-Safeguard serves as an upper bound in our experiments, since it uses extra ground-truth attacked agent data for supervised model training. For unsupervised methods, we take representative GAD methods for comparisons, including: DOMINANT (Ding et al. 2019), a generation-based method; PREM (Pan et al. 2023), a contrastive learning-based method; and TAM (Qiao and Pang 2023), an affinity-driven method.

| Topology | Method | PI (CSQA) | | PI (MMLU) | | PI (GSM8k) | | TA (InjecAgent) | | MA (PosionRAG) | | MA (CSQA) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AUC | ASR@3 | AUC | ASR@3 | AUC | ASR@3 | AUC | ASR@3 | AUC | ASR@3 | AUC | ASR@3 |
| Chain | No Defense | - | 38.33 | - | 34.67 | - | 9.83 | - | 48.00 | - | 22.33 | - | 26.00 |
| | G-Safeguard | 100.00 | 19.67 | 98.22 | 17.00 | 98.22 | 4.40 | 100.00 | 10.24 | 100.00 | 6.00 | 94.67 | 5.67 |
| | DOMINANT | 42.22 | 28.00 | 53.78 | 24.67 | 67.56 | **8.47** | 88.00 | **14.98** | 64.44 | **14.00** | 27.56 | 35.33 |
| | PREM | 51.56 | 26.33 | 48.00 | 26.33 | 62.22 | 8.79 | **89.33** | 15.17 | 61.33 | 16.33 | 57.78 | 17.33 |
| | TAM | 26.67 | 57.78 | 49.33 | 25.00 | 51.56 | 8.84 | 61.33 | 30.04 | 50.67 | 17.67 | 53.78 | 18.67 |
| | BlindGuard | **79.11** | **25.00** | **84.89** | **21.33** | **69.33** | **8.47** | 86.22 | 16.38 | **81.33** | 15.67 | **74.67** | **12.33** |
| Tree | No Defense | - | 33.33 | - | 33.33 | - | 10.20 | - | 45.05 | - | 20.33 | - | 20.33 |
| | G-Safeguard | 100.00 | 18.33 | 99.11 | 18.33 | 99.11 | 7.80 | 100.00 | 4.76 | 99.56 | 8.00 | 90.67 | 9.00 |
| | DOMINANT | 47.11 | 25.67 | 56.44 | 20.67 | **68.44** | **6.78** | **88.44** | 15.33 | 64.44 | 11.00 | 29.78 | 22.67 |
| | PREM | 52.44 | 25.67 | 41.78 | 21.56 | 53.33 | 8.47 | 85.78 | 16.21 | 56.89 | 11.67 | 58.22 | 18.67 |
| | TAM | 56.89 | 25.00 | 54.67 | 21.33 | 54.67 | 8.14 | 61.33 | 32.01 | 58.22 | 12.33 | 55.56 | 17.33 |
| | BlindGuard | **75.56** | **20.67** | **79.56** | **20.33** | 59.55 | 8.47 | 85.78 | **12.50** | **76.44** | **10.00** | **77.33** | **13.33** |
| Star | No Defense | - | 46.33 | - | 42.33 | - | 12.89 | - | 43.57 | - | 23.33 | - | 24.67 |
| | G-Safeguard | 100.00 | 18.33 | 99.11 | 18.00 | 98.22 | 6.10 | 100.00 | 6.87 | 100.00 | 7.33 | 95.11 | 5.67 |
| | DOMINANT | 47.56 | 30.33 | 55.11 | 24.67 | 69.78 | 7.80 | 89.33 | 14.33 | 62.67 | 14.67 | 27.56 | 35.33 |
| | PREM | 51.56 | 32.67 | 45.33 | 27.33 | 59.56 | 10.51 | **93.78** | 14.68 | 62.22 | 12.67 | 56.44 | 25.33 |
| | TAM | 60.00 | 28.00 | 64.89 | 25.33 | 68.44 | 8.14 | 71.11 | 26.57 | 62.67 | 20.00 | 59.56 | 18.00 |
| | BlindGuard | **82.67** | **24.00** | **85.33** | **21.33** | **70.22** | **6.78** | **93.78** | **12.59** | **85.33** | **12.00** | **74.67** | **12.33** |
| Random | No Defense | - | 35.67 | - | 48.33 | - | 14.48 | - | 39.78 | - | 26.21 | - | 31.33 |
| | G-Safeguard | 98.22 | 18.67 | 99.56 | 19.33 | 99.11 | 3.79 | 98.22 | 5.14 | 97.70 | 10.34 | 91.56 | 7.67 |
| | Dominant | 44.89 | 33.67 | 60.00 | 31.00 | 69.78 | 10.51 | 84.63 | 14.93 | 61.33 | 18.33 | 25.78 | 35.33 |
| | PREM | 53.33 | 31.67 | 40.85 | 33.09 | 69.78 | 10.51 | **86.22** | **14.49** | 61.33 | 12.33 | 57.78 | 25.33 |
| | TAM | 47.11 | 36.67 | 47.56 | 38.00 | 46.22 | 14.91 | 52.00 | 35.78 | 48.44 | 25.33 | 51.11 | 26.33 |
| | BlindGuard | **76.89** | **23.67** | **84.00** | **24.67** | **75.56** | **6.44** | 79.56 | 17.69 | **82.22** | **9.67** | **74.67** | **16.00** |

Table 1: AUC and ASR@3 of different defense methods with GPT-4o-mini serving as the backbone LLM. Following G-safeguard (Wang et al. 2025), we consider three types of attack: Prompt injection (PI), tool attack (TA), and memory attack (MA). We showcase results after round 3 communications (ASR@3), and the additional results are placed in Appendix D.

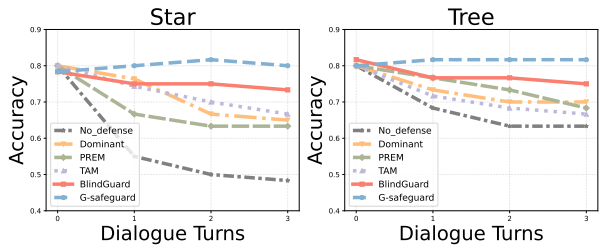


Figure 3: The overall performance of MAS on the CSQA dataset after each turn of dialogue.

**Implementation** We evaluate the defensive capabilities of BlindGuard through comprehensive experiments spanning multiple attack types, topological structures, and LLM backbones. Following G-Safeguard, our testing framework employs three primary attack methods: direct prompt, tool attack, and memory poisoning. For network topologies, we examine four distinct MAS structures - chain, tree, star, and random - to validate generalization across communication patterns. The experiments incorporate both open-source LLMs (Qwen3-30B-A3B (Yang et al. 2025), Deepseek-v3 (Liu et al. 2024)) and commercial LLMs (GPT-4o-mini) as agent backbones. Critical performance metrics include Attack Success Rate after three communication rounds (ASR@3) and Area Under Curve (AUC) of malicious agent detection. To ensure fairness and practicality, we set a budget to identify the top three agents with the highest risk in the MAS as the predicted malicious agents. More experimental setups can be found in Appendix C.

## Experimental Results

**Performance Comparison (RQ1)** We evaluate the effectiveness of BlindGuard on GPT-4o-mini backbone in four topologies against three attack types. We list the comparison results in Table 1 and Figure 3, which lead to the following observations. ❶ *BlindGuard significantly outperforms other unsupervised methods in defense capability.* Compared to other GAD-based solutions, BlindGuard achieves competitive defense performance consistently against all attack types. In contrast, the baselines sometimes fail in several scenarios, such as TAM and PREM on PI (CSQA) and DOMINANT on PI (MMLU). The superior performance demonstrates the

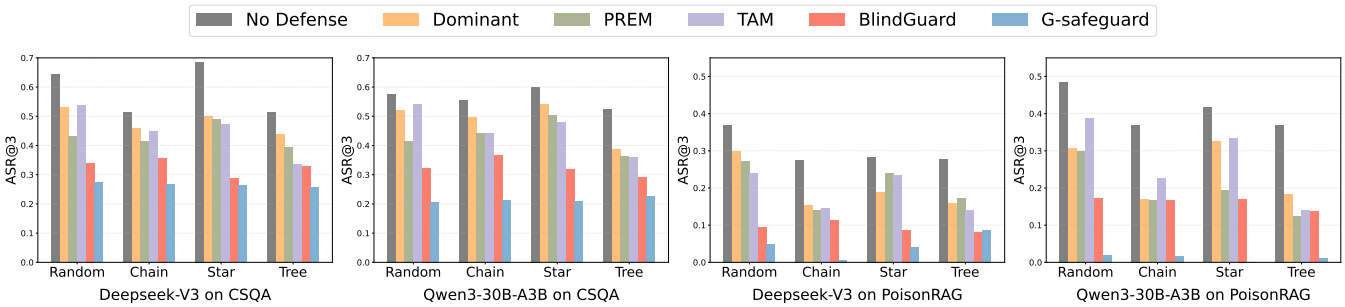


Figure 4: ASR@3 with DeepSeek-V3 and Qwen3-30B-A3B as backbone LLMs on the CSQA and PoisonRAG datasets.

significance of designing a specific model for unsupervised malicious agent detection. ❷ *BlindGuard shows competitive performance compared to supervised upper bound.* While G-Safeguard, which relies on labeled attacked data for training, achieves the best overall defense performance, BlindGuard shows comparable effectiveness in most cases, with an AUC$>80\%$. This illustrates the feasibility of training an effective and universal defense model for MAS without relying on annotated data. ❸ *BlindGuard effectively improves response accuracy of MAS under adversarial attack.* As shown in Figure 3 (more can be found in Appendix D), the response accuracy of MAS exhibits a clear downward trend as dialogue turns increase across all topologies without defense. While all implemented defense methods show improvements, BlindGuard demonstrates superior and consistent defense capabilities compared to existing unsupervised defense methods.

**Universal Generalization (RQ2)** To investigate the generalizability of BlindGuard, we conducted additional experiments using DeepSeek-V3 and Qwen3-30B-A3B as backbone LLMs on the CSQA and PoisonRAG datasets, as shown in Figure 4 (more can be found in Appendix D). Through experiments, we make the following observations. ❹ *BlindGuard obtains robust defense performance when deployed with diverse LLM and topologies.* As shown in Figure 4, BlindGuard maintains robust defense performance in ASR@3 and AUC across different LLM backbones and topological structures. This stable performance confirms that BlindGuard effectively captures universal adversarial patterns rather than overfitting to specific LLM or topologies. ❺ *BlindGuard successfully generalizes to different attack types on the same dataset using one universal trained model.* As shown in Table 1, while supervised defense methods like G-Safeguard require training specialized models for different attack types on the same dataset (e.g., CSQA), BlindGuard achieves defense using a single model for both attack types (i.e., PI and MA). This property shows the potential of BlindGuard to be a universal defense model against different unseen attacks.

**Scalability (RQ3)** To investigate the scalability of BlindGuard to larger-scale MAS, we report defense performance of PoisonRAG across systems with 20 and 50 agents, as shown in Table 2. We observe that ❻ *BlindGuard consistently mitigates adversarial impact across all rounds (R1–R3) in larger-scale MAS.* The scalability of BlindGuard is caused by its topology-agnostic design, where hierarchical agent encoder and corruption-guided attack detector eliminate dependencies on fixed agent numbers or interaction patterns, thereby ensuring consistent performance across diverse scales. This defense under scaling demonstrates the practicality of BlindGuard for real-world large-scale MAS.

| Agent Num | Method | R1 | R2 | R3 |
|---|---|---|---|---|
| 20 | No Defense | 15.89 | 23.22 | 29.51 |
| | BlindGuard | **3.51** | **4.54** | **5.57** |
| 50 | No Defense | 5.67 | 16.31 | 20.92 |
| | BlindGuard | **1.81** | **2.66** | **3.76** |

Table 2: ASR@3 on different agent numbers and rounds.

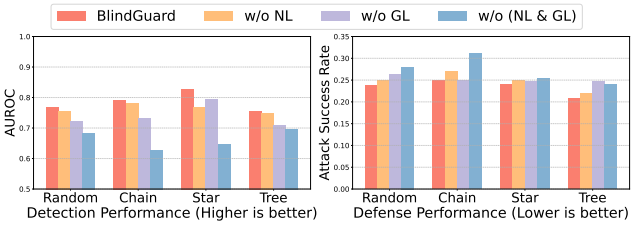


Figure 5: Ablation study on PoisonRAG. NL and GL denote neighbor-level and global-level features, respectively.

**Ablation study (RQ4)** To study the hierarchical agent encoder's role in BlindGuard is quantified, we conduct an ablation study on the PoisonRAG dataset. As shown in Figure 5, we observe that ❼ *anomaly detection in MAS requires a combination of both local neighborhood interactions and global system context.* Removing neighborhood and global context features leads to significant performance degradation, and their combined absence causes a severe drop, highlighting the critical role of structural context beyond agent-level features. This observation shows the significance of combining information at multiple levels.

## Conclusion

In this paper, we present BlindGuard, an unsupervised defense method for LLM-based MAS that integrates hierar-

chical agent encoder and corruption-guided attack detector. By fusing agent-level, neighborhood, and global information, BlindGuard achieves robust protection without requiring attack-specific training data. Experimental results demonstrate that BlindGuard effectively mitigates diverse attacks across various topologies while maintaining scalability. This work advances the security of MAS by providing a practical and attack-agnostic defense solution, shedding light on generalizable defenses for LLM-based MAS.

## References


Andriushchenko, M.; Souly, A.; Dziemian, M.; Duenas, D.; Lin, M.; Wang, J.; Hendrycks, D.; Zou, A.; Kolter, J. Z.; Fredrikson, M.; et al. 2025. AgentHarm: A Benchmark for Measuring Harmfulness of LLM Agents. In *The Thirteenth International Conference on Learning Representations*.

Chen, Z.; Xiang, Z.; Xiao, C.; Song, D.; and Li, B. 2024. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37: 130185–130213.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ding, K.; Li, J.; Bhanushali, R.; and Liu, H. 2019. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM international conference on data mining*, 594–602. SIAM.

Gan, Y.; Yang, Y.; Ma, Z.; He, P.; Zeng, R.; Wang, Y.; Li, Q.; Zhou, C.; Li, S.; Wang, T.; et al. 2024. Navigating the risks: A survey of security, privacy, and ethics threats in llm-based agents. *arXiv preprint arXiv:2411.09523*.

Gao, D.; Li, Z.; Pan, X.; Kuang, W.; Ma, Z.; Qian, B.; Wei, F.; Zhang, W.; Xie, Y.; Chen, D.; et al. 2024. Agentscope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*.

Guo, T.; Chen, X.; Wang, Y.; Chang, R.; Pei, S.; Chawla, N. V.; Wiest, O.; and Zhang, X. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

He, P.; Dai, Z.; Tang, X.; Xing, Y.; Liu, H.; Zeng, J.; Peng, Q.; Agrawal, S.; Varshney, S.; Wang, S.; et al. 2025. Attention Knows Whom to Trust: Attention-based Trust Management for LLM Multi-Agent Systems. *arXiv preprint arXiv:2506.02546*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

Kannan, S. S.; Venkatesh, V. L.; and Min, B.-C. 2024. Smart-llm: Smart multi-agent robot task planning using large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 12140–12147. IEEE.

Kim, Y.; Park, C.; Jeong, H.; Chan, Y. S.; Xu, X.; McDuff, D.; Lee, H.; Ghassemi, M.; Breazeal, C.; and Park, H. W. 2024. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37: 79410–79452.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*.

Lei, B.; Zhang, Y.; Zuo, S.; Payani, A.; and Ding, C. 2024. Macm: Utilizing a multi-agent system for condition mining in solving complex mathematical problems. *Advances in Neural Information Processing Systems*, 37: 53418–53437.

Li, S.; Liu, Y.; Chen, Q.; Webb, G. I.; and Pan, S. 2024a. Noise-resilient unsupervised graph representation learning via multi-hop feature quality estimation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 1255–1265.

Li, S.; Liu, Y.; Wen, Q.; Zhang, C.; and Pan, S. 2025a. Assemble Your Crew: Automatic Multi-agent Communication Topology Design via Autoregressive Graph Generation. *arXiv preprint arXiv:2507.18224*.

Li, X.; Wang, S.; Zeng, S.; Wu, Y.; and Yang, Y. 2024b. A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1): 9.

Li, X.; Zeng, Y.; Xing, X.; Xu, J.; and Xu, X. 2025b. Hedgeagents: A balanced-aware multi-agent financial trading system. In *Companion Proceedings of the ACM on Web Conference 2025*, 296–305.

Li, Y.; Du, Y.; Zhang, J.; Hou, L.; Grabowski, P.; Li, Y.; and Ie, E. 2024c. Improving Multi-Agent Debate with Sparse Communication Topology. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 7281–7294.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liu, Y.; Li, Z.; Pan, S.; Gong, C.; Zhou, C.; and Karypis, G. 2021. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE transactions on neural networks and learning systems*, 33(6): 2378–2392.

Liu, Y.; Zhang, G.; Wang, K.; Li, S.; and Pan, S. 2025. Graph-Augmented Large Language Model Agents: Current Progress and Future Prospects. *arXiv preprint arXiv:2507.21407*.

Ma, R.; Pang, G.; Chen, L.; and Van Den Hengel, A. 2022. Deep graph-level anomaly detection by glocal knowledge distillation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, 704–714.

Masterman, T.; Besen, S.; Sawtell, M.; and Chao, A. 2024. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*.

Nazary, F.; Deldjoo, Y.; and Noia, T. d. 2025. Poison-rag: Adversarial data poisoning attacks on retrieval-augmented generation in recommender systems. In *European Conference on Information Retrieval*, 239–251. Springer.

Pan, J.; Liu, Y.; Zheng, X.; Zheng, Y.; Liew, A. W.-C.; Li, F.; and Pan, S. 2025. A label-free heterophily-guided approach for unsupervised graph fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 12443–12451.

Pan, J.; Liu, Y.; Zheng, Y.; and Pan, S. 2023. Prem: A simple yet effective approach for node-level graph anomaly detection. In *2023 IEEE International Conference on Data Mining (ICDM)*, 1253–1258. IEEE.

Qian, C.; Liu, W.; Liu, H.; Chen, N.; Dang, Y.; Li, J.; Yang, C.; Chen, W.; Su, Y.; Cong, X.; et al. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Qiao, H.; and Pang, G. 2023. Truncated affinity maximization: One-class homophily modeling for graph anomaly detection. *Advances in Neural Information Processing Systems*, 36: 49490–49512.

Qiao, H.; Tong, H.; An, B.; King, I.; Aggarwal, C.; and Pang, G. 2024. Deep graph anomaly detection: A survey and new perspectives. *arXiv preprint arXiv:2409.09957*.

Reimers, N.; and Gurevych, I. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Shen, X.; Liu, Y.; Dai, Y.; Wang, Y.; Miao, R.; Tan, Y.; Pan, S.; and Wang, X. 2025. Understanding the Information Propagation Effects of Communication Topologies in LLM-based Multi-Agent Systems. *arXiv preprint arXiv:2505.23352*.

Talmor, A.; Herzig, J.; Lourie, N.; and Berant, J. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Tran, K.-T.; Dao, D.; Nguyen, M.-D.; Pham, Q.-V.; O'Sullivan, B.; and Nguyen, H. D. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.

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*.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, 6861–6871. Pmlr.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

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

Yu, M.; Meng, F.; Zhou, X.; Wang, S.; Mao, J.; Pang, L.; Chen, T.; Wang, K.; Li, X.; Zhang, Y.; et al. 2025. A survey on trustworthy llm agents: Threats and countermeasures. *arXiv preprint arXiv:2503.09648*.

Yu, M.; Wang, S.; Zhang, G.; Mao, J.; Yin, C.; Liu, Q.; Wen, Q.; Wang, K.; and Wang, Y. 2024. Netsafe: Exploring the topological safety of multi-agent networks. *arXiv preprint arXiv:2410.15686*.

Zhan, Q.; Liang, Z.; Ying, Z.; and Kang, D. 2024a. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics ACL 2024*, 10471–10506.

Zhan, Q.; Liang, Z.; Ying, Z.; and Kang, D. 2024b. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. *arXiv preprint arXiv:2403.02691*.

Zhang, G.; Yue, Y.; Li, Z.; Yun, S.; Wan, G.; Wang, K.; Cheng, D.; Yu, J. X.; and Chen, T. 2025a. Cut the Crap: An Economical Communication Pipeline for LLM-based Multi-Agent Systems. In *The Thirteenth International Conference on Learning Representations*.

Zhang, G.; Yue, Y.; Sun, X.; Wan, G.; Yu, M.; Fang, J.; Wang, K.; Chen, T.; and Cheng, D. 2025b. G-Designer: Architecting Multi-agent Communication Topologies via Graph Neural Networks. In *Forty-second International Conference on Machine Learning*.

Zhang, W.; Yin, Z.; Sheng, Z.; Li, Y.; Ouyang, W.; Li, X.; Tao, Y.; Yang, Z.; and Cui, B. 2022. Graph attention multi-layer perceptron. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4560–4570.

Zhang, Z.; Dai, Q.; Bo, X.; Ma, C.; Li, R.; Chen, X.; Zhu, J.; Dong, Z.; and Wen, J.-R. 2024. A survey on the memory mechanism of large language model based agents. *ACM Transactions on Information Systems*.

Zhao, Q.; Wang, J.; Zhang, Y.; Jin, Y.; Zhu, K.; Chen, H.; and Xie, X. 2023. Competeai: Understanding the competition behaviors in large language model-based agents. *arXiv preprint arXiv:2310.17512*.

Zhao, Z.; Chai, W.; Wang, X.; Li, B.; Hao, S.; Cao, S.; Ye, T.; and Wang, G. 2024. See and think: Embodied agent in virtual environment. In *European Conference on Computer Vision*, 187–204. Springer.

Zheng, Z.; Zhang, O.; Nguyen, H. L.; Rampal, N.; Alawadhi, A. H.; Rong, Z.; Head-Gordon, T.; Borgs, C.; Chayes, J. T.; and Yaghi, O. M. 2023. ChatGPT research group for optimizing the crystallinity of MOFs and COFs. *ACS Central Science*, 9(11): 2161–2170.

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*.

# A. Related Work

## LLM-based Multi-agent System

Recent advances in LLM-based MAS have demonstrated remarkable capabilities in general task-solving. The performance of MAS is predominantly determined by collaboration and communication among agents with diverse roles and expertise (Tran et al. 2025), where well-structured communication can improve overall effectiveness. Modern LLM-based MAS implementations employ varied collaboration strategies to optimize performance. Sequential reasoning and debate-based role specialization (Li et al. 2024c) have proven particularly effective for knowledge-intensive tasks, while centralized planning architectures demonstrate superior performance in goal-oriented scenarios. Notable frameworks include conversational agent networks in AutoGen (Wu et al. 2024), the developer-centric platform in AgentScope (Gao et al. 2024), and phase-structured software development in ChatDev (Qian et al. 2023). The effectiveness of LLM-based MAS has been closely tied to the quality of its communication topologies. Recent research has explored MAS based on graph algorithms (Zhang et al. 2025a,b). Despite their effectiveness, these graph-based MAS topologies remain vulnerable to adversarial manipulation, where malicious agents can exploit the communication structure to inject misinformation, disrupt coordination, or compromise collective decisions.

## Security of LLM-based MAS

Despite the effectiveness of LLM-based MAS, this advancement has introduced novel security risks, particularly threats that exploit agent memory (Chen et al. 2024) and tool-handling mechanisms (Zhan et al. 2024a). The most severe threats target message-passing mechanisms (Zhou et al. 2025), enabling malicious attackers to implant prejudiced content. NetSafe (Yu et al. 2024) pioneers the study of network structure vulnerabilities, identifying bias propagation patterns in a multi-agent utterance graph. G-Safeguard (Wang et al. 2025) advances supervised detection of compromised agents through graph neural networks and topological remediation. A-Trust (He et al. 2025) develops attention-based trust metrics by analyzing violation patterns across six fundamental trust dimensions. While these methods can mitigate certain security threats, they heavily rely on labeled malicious agents or prior knowledge of attack patterns, which may not be available in real-world MAS deployments.

# B. Algorithm

For a detailed implementation of our proposed BlindGuard, please refer to Algorithm 1.

Algorithm 1: BlindGuard

**Input**: Normal MAS graphs $\{\mathcal{G}_1, \ldots, \mathcal{G}_{T_n}\}$, Attacked MAS graphs $\{\mathcal{G}'_1, \ldots, \mathcal{G}'_{T_a}\}$, hierarchical agent encoder $g_\theta$, Intensity parameter $\alpha$ and Anomaly budget $K$.

**Output**: Final responses $\{\widetilde{\mathcal{R}}_1, \ldots, \widetilde{\mathcal{R}}_{T_a}\}$ of all remediated MAS $\{\widetilde{\mathcal{G}}'_1, \ldots, \widetilde{\mathcal{G}}'_{T_a}\}$

**for** each normal MAS $\mathcal{G}_i \in \{\mathcal{G}_1, \ldots, \mathcal{G}_{T_n}\}$ **do**
  **for** each agent $v_j \in \mathcal{G}_i$ **do**
    $\mathbf{x}_j \leftarrow \text{SentenceBERT}(R_j)$
    // Node feature construction.
  **end for**
  Sample subset $\mathcal{V}_{\text{corr}} \subset \mathcal{V}$
  **for** each agent $v_j \in \mathcal{V}_{\text{corr}}$ **do**
    $\mathbf{x}_j \leftarrow \mathbf{x}_j + \alpha \|\mathbf{x}_j\|_2 \cdot \frac{\boldsymbol{\epsilon}_j}{\|\boldsymbol{\epsilon}_j\|_2}, \boldsymbol{\epsilon}_j \sim \mathcal{N}(0, \mathbf{I})$
    // Feature corruption.
  **end for**
  **for** each agent $v_j \in \mathcal{G}_i$ **do**
    $\mathbf{h}_j^{\text{self}} \leftarrow \mathbf{x}_j$
    $\mathbf{h}_j^{\text{neigh}} \leftarrow \sum_{k \in \mathcal{N}(j)} \hat{A}_{jk} \mathbf{x}_k$
    $\mathbf{h}_j^{\text{graph}} \leftarrow \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k$
    $\mathbf{z}_j \leftarrow g_\theta(\mathbf{h}_j^{\text{self}} \parallel \mathbf{h}_j^{\text{neigh}} \parallel \mathbf{h}_j^{\text{graph}})$
    // Obtain agent representations.
  **end for**
  Calculate supervised contrastive loss $\mathcal{L}$
  Update $\theta$ via gradient descent $\bigtriangledown_\theta [\mathcal{L}]$
**end for**
**for** each attacked MAS $\mathcal{G}'_i \in \{\mathcal{G}'_1, \ldots, \mathcal{G}'_{T_a}\}$ **do**
  **for** each agent $v_j \in \mathcal{G}'_t$ **do**
    $\mathbf{x}_j \leftarrow \text{SentenceBERT}(R_j)$
    $\mathbf{z}_j \leftarrow f_\theta(\mathbf{x}_j, \mathcal{G}'_t)$
    $s_j \leftarrow -\frac{1}{N} \sum_{k=1}^{N} \text{sim}(\mathbf{z}_j, \mathbf{z}_k)$
    // Compute anomaly score.
  **end for**
  $\mathcal{V}_{\text{atk}} \leftarrow$ Top-K agents with highest $s_j$
  $\mathcal{E}^+ \leftarrow \{e_{kj} \in \mathcal{E} \mid v_k \notin \mathcal{V}_{\text{atk}}\}$
  // MAS remediation.
  **for** each agent $v_j \in \mathcal{G}'_t$ **do**
    $R_j \leftarrow \text{LLM}\left(Q \cup \{R_k \mid e_{kj} \in \mathcal{E}^+\}\right)$
  **end for**
  Determine the final answer $\widetilde{\mathcal{R}}_i$ by aggregating all agent responses (e.g., majority voting)
**end for**
**return** $\{\widetilde{\mathcal{R}}_1, \ldots, \widetilde{\mathcal{R}}_{T_a}\}$

# C. Detailed Experimental Setup

We employ the Adam optimizer (Kingma and Ba 2014) with an initial learning rate of 0.001 and L2 regularization (weight decay $\in \{5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}\}$). The learning rate is dynamically adjusted using a cosine annealing scheduler ($T_{\max} = 10$ cycles and $\eta_{\min} = 10^{-5}$) to facilitate better convergence. All models are implemented with a hidden dimension of 512 and trained on 4 NVIDIA L40 GPUs.

# D. Additional Experiments

To further validate the effectiveness of our proposed BlindGuard, we conduct additional experiments. As shown in Figure 6, the experiments demonstrate BlindGuard's consistent performance advantages, showing superior AUC scores over Dominant, PREM, and TAM across four topologies with both DeepSeek-V3 and Qwen3-30B-A3B LLMs on CSQA

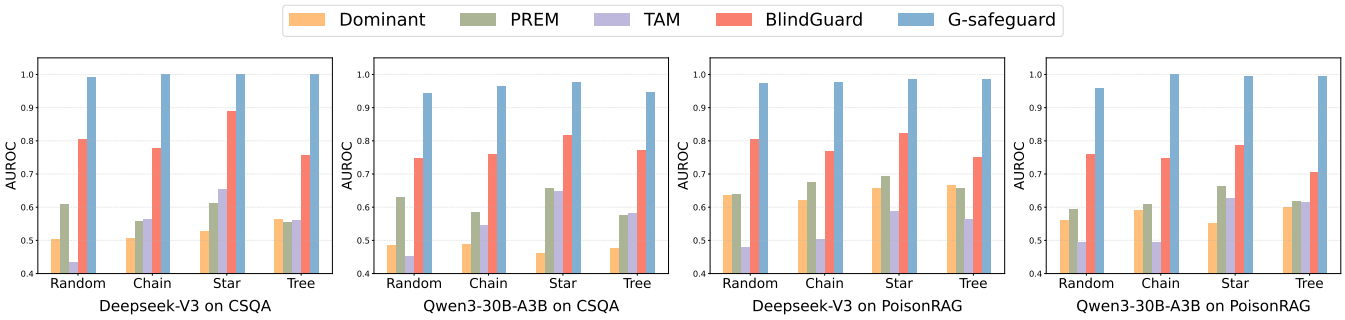


Figure 6: The AUC with DeepSeek-V3 and Qwen3-30B-A3B as backbone LLMs on the CSQA and PoisonRAG datasets.

| CSQA (PI) | | | | | MMLU (PI) | | | | | GSM8K (PI) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Topology** | **Defense** | **R1** | **R2** | **R3** | **Topology** | **Defense** | **R1** | **R2** | **R3** | **Topology** | **Defense** | **R1** | **R2** | **R3** |
| Chain | ND | 0.283 | 0.363 | 0.383 | Chain | ND | 0.273 | 0.317 | 0.347 | Chain | ND | 0.064 | 0.092 | 0.098 |
| | Dom | 0.253 | 0.290 | 0.280 | | Dom | 0.207 | 0.230 | 0.247 | | Dom | 0.071 | 0.088 | 0.085 |
| | PREM | 0.243 | 0.247 | 0.263 | | PREM | 0.237 | 0.257 | 0.263 | | PREM | 0.061 | 0.075 | 0.088 |
| | TAM | 0.247 | 0.263 | 0.267 | | TAM | 0.220 | 0.240 | 0.250 | | TAM | 0.064 | 0.088 | 0.088 |
| | BG | 0.213 | 0.233 | 0.250 | | BG | 0.197 | 0.200 | 0.213 | | BG | 0.061 | 0.085 | 0.085 |
| | GS | 0.187 | 0.193 | 0.197 | | GS | 0.167 | 0.160 | 0.170 | | GS | 0.044 | 0.044 | 0.044 |
| Tree | ND | 0.260 | 0.337 | 0.333 | Tree | ND | 0.257 | 0.327 | 0.333 | Tree | ND | 0.075 | 0.082 | 0.102 |
| | Dom | 0.223 | 0.253 | 0.257 | | Dom | 0.193 | 0.203 | 0.207 | | Dom | 0.061 | 0.061 | 0.068 |
| | PREM | 0.210 | 0.230 | 0.257 | | PREM | 0.193 | 0.200 | 0.200 | | PREM | 0.064 | 0.075 | 0.085 |
| | TAM | 0.230 | 0.247 | 0.250 | | TAM | 0.210 | 0.230 | 0.213 | | TAM | 0.071 | 0.075 | 0.081 |
| | BG | 0.213 | 0.210 | 0.207 | | BG | 0.203 | 0.207 | 0.203 | | BG | 0.075 | 0.075 | 0.085 |
| | GS | 0.183 | 0.180 | 0.183 | | GS | 0.170 | 0.170 | 0.183 | | GS | 0.071 | 0.075 | 0.078 |
| Star | ND | 0.360 | 0.423 | 0.463 | Star | ND | 0.337 | 0.400 | 0.423 | Star | ND | 0.071 | 0.115 | 0.129 |
| | Dom | 0.260 | 0.293 | 0.303 | | Dom | 0.213 | 0.240 | 0.247 | | Dom | 0.054 | 0.071 | 0.078 |
| | PREM | 0.290 | 0.317 | 0.327 | | PREM | 0.227 | 0.267 | 0.273 | | PREM | 0.068 | 0.095 | 0.105 |
| | TAM | 0.223 | 0.277 | 0.280 | | TAM | 0.203 | 0.247 | 0.253 | | TAM | 0.061 | 0.081 | 0.081 |
| | BG | 0.227 | 0.230 | 0.240 | | BG | 0.187 | 0.203 | 0.213 | | BG | 0.068 | 0.068 | 0.068 |
| | GS | 0.197 | 0.193 | 0.183 | | GS | 0.177 | 0.180 | 0.180 | | GS | 0.054 | 0.064 | 0.061 |
| Random | ND | 0.290 | 0.337 | 0.357 | Random | ND | 0.373 | 0.453 | 0.483 | Random | ND | 0.059 | 0.097 | 0.145 |
| | Dom | 0.293 | 0.357 | 0.337 | | Dom | 0.253 | 0.300 | 0.310 | | Dom | 0.054 | 0.085 | 0.105 |
| | PREM | 0.277 | 0.310 | 0.317 | | PREM | 0.236 | 0.324 | 0.331 | | PREM | 0.037 | 0.078 | 0.105 |
| | TAM | 0.297 | 0.343 | 0.367 | | TAM | 0.297 | 0.360 | 0.380 | | TAM | 0.047 | 0.098 | 0.149 |
| | BG | 0.230 | 0.233 | 0.237 | | BG | 0.210 | 0.233 | 0.247 | | BG | 0.041 | 0.054 | 0.064 |
| | GS | 0.193 | 0.183 | 0.187 | | GS | 0.187 | 0.190 | 0.193 | | GS | 0.048 | 0.038 | 0.038 |

Table 3: Attack success rate (ASR@3) comparison of defense methods with GPT-4o-mini as backbone LLMs (Part 1). ND = No Defense, Dom = Dominant, BG = BlindGuard, GS = G-Safeguard. Lower values indicate better defense performance.

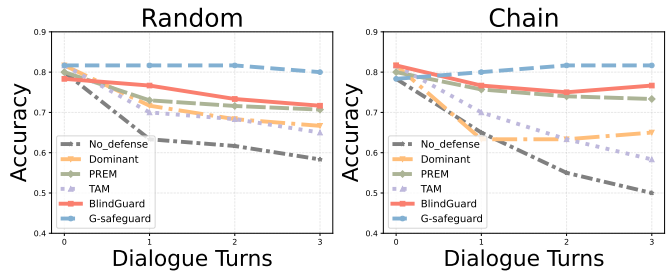


Figure 7: The overall performance of MAS on the CSQA dataset after each turn of dialogue. We use majority voting as the strategy to select the final answer.

and PoisonRAG benchmarks. Figure 7 provides extended experimental results comparing BlindGuard with baseline methods (Dominant, PREM, TAM, and G-safeguard) across multiple dialogue turns, demonstrating consistent accuracy improvements while maintaining efficient communication in the CSQA task. As shown in Table 1 and Table 2, additional experiments provide comprehensive multi-round ASR@3 comparisons (R1-R3) across multiple attack types and network topologies, demonstrating BlindGuard's superior performance over Dominant, PREM, and TAM while approaching G-Safeguard's effectiveness throughout progressive dialogue stages in various attack types.

| InjecAgent (TA) | | | | | CSQA (MA) | | | | | PoisonRAG (MA) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Topology** | **Defense** | **R1** | **R2** | **R3** | **Topology** | **Defense** | **R1** | **R2** | **R3** | **Topology** | **Defense** | **R1** | **R2** | **R3** |
| Chain | ND | 0.337 | 0.442 | 0.480 | Chain | ND | 0.150 | 0.237 | 0.260 | Chain | ND | 0.127 | 0.193 | 0.223 |
| | Dom | 0.153 | 0.142 | 0.150 | | Dom | 0.167 | 0.187 | 0.210 | | Dom | 0.093 | 0.127 | 0.140 |
| | PREM | 0.147 | 0.152 | 0.152 | | PREM | 0.147 | 0.173 | 0.173 | | PREM | 0.090 | 0.147 | 0.163 |
| | TAM | 0.243 | 0.293 | 0.300 | | TAM | 0.140 | 0.173 | 0.187 | | TAM | 0.130 | 0.167 | 0.177 |
| | BG | 0.142 | 0.178 | 0.164 | | BG | 0.067 | 0.093 | 0.123 | | BG | 0.093 | 0.133 | 0.157 |
| | GS | 0.122 | 0.108 | 0.102 | | GS | 0.067 | 0.060 | 0.057 | | GS | 0.050 | 0.063 | 0.060 |
| Tree | ND | 0.289 | 0.428 | 0.451 | Tree | ND | 0.130 | 0.200 | 0.203 | Tree | ND | 0.107 | 0.170 | 0.203 |
| | Dom | 0.146 | 0.149 | 0.153 | | Dom | 0.150 | 0.220 | 0.227 | | Dom | 0.087 | 0.113 | 0.110 |
| | PREM | 0.159 | 0.182 | 0.162 | | PREM | 0.157 | 0.133 | 0.187 | | PREM | 0.060 | 0.087 | 0.116 |
| | TAM | 0.231 | 0.289 | 0.321 | | TAM | 0.157 | 0.147 | 0.173 | | TAM | 0.060 | 0.117 | 0.123 |
| | BG | 0.142 | 0.143 | 0.125 | | BG | 0.090 | 0.103 | 0.133 | | BG | 0.060 | 0.090 | 0.100 |
| | GS | 0.076 | 0.061 | 0.048 | | GS | 0.110 | 0.070 | 0.090 | | GS | 0.050 | 0.067 | 0.080 |
| Star | ND | 0.368 | 0.482 | 0.436 | Star | ND | 0.127 | 0.213 | 0.247 | Star | ND | 0.107 | 0.187 | 0.233 |
| | Dom | 0.130 | 0.151 | 0.143 | | Dom | 0.183 | 0.297 | 0.353 | | Dom | 0.093 | 0.123 | 0.147 |
| | PREM | 0.132 | 0.139 | 0.147 | | PREM | 0.173 | 0.240 | 0.253 | | PREM | 0.070 | 0.103 | 0.127 |
| | TAM | 0.246 | 0.266 | 0.266 | | TAM | 0.120 | 0.160 | 0.180 | | TAM | 0.117 | 0.180 | 0.200 |
| | BG | 0.118 | 0.132 | 0.126 | | BG | 0.047 | 0.103 | 0.123 | | BG | 0.050 | 0.103 | 0.120 |
| | GS | 0.085 | 0.072 | 0.069 | | GS | 0.053 | 0.027 | 0.057 | | GS | 0.037 | 0.057 | 0.073 |
| Random | ND | 0.336 | 0.409 | 0.398 | Random | ND | 0.157 | 0.267 | 0.313 | Random | ND | 0.128 | 0.210 | 0.262 |
| | Dom | 0.144 | 0.160 | 0.149 | | Dom | 0.193 | 0.310 | 0.353 | | Dom | 0.100 | 0.153 | 0.183 |
| | PREM | 0.115 | 0.136 | 0.145 | | PREM | 0.140 | 0.230 | 0.263 | | PREM | 0.067 | 0.110 | 0.123 |
| | TAM | 0.321 | 0.383 | 0.358 | | TAM | 0.147 | 0.220 | 0.263 | | TAM | 0.153 | 0.237 | 0.253 |
| | BG | 0.125 | 0.173 | 0.177 | | BG | 0.053 | 0.110 | 0.160 | | BG | 0.037 | 0.067 | 0.097 |
| | GS | 0.079 | 0.066 | 0.051 | | GS | 0.077 | 0.080 | 0.077 | | GS | 0.041 | 0.090 | 0.103 |

Table 4: Attack success rate (ASR@3) comparison of defense methods with GPT-4o-mini as backbone LLMs (Part 2). Abbreviations same as in Table 1.