Memory Poisoning Propagation and Repair Mechanism in Multi-Agent Collaborative Environments

Hongrui Liu *, Duo Xu , Qianli Ma , Shuyang Xu , Dong Qiu

Posted Date: 14 February 2026

doi: 10.20944/preprints202602.1188.v1

Keywords: multi-agent systems; memory security; poisoning attacks; collaborative control; knowledge governance

 doi:10.20944/preprints202602.1188.v1

Article

Memory Poisoning Propagation and Repair Mechanism in Multi-Agent Collaborative Environments

Hongrui Liu [1,*], Duo Xu [2], Qianli Ma [3], Shuyang Xu [4] and Dong Qiu [5]

[1] University of Michigan – Ann Arbor, Ann Arbor, 48109, United States
[2] Northeastern University, San Jose, 95113, United States
[3] University of Massachusetts Boston, Boston, 02125, USA
[4] CORNELL UNIVERSITY, ITHACA, 14850, UNITED STATES
[5] New England College, Henniker, 03242, United States
* Correspondence: hongruil@umich.edu

Abstract

Multi-agent systems often rely on long-term memory or shared knowledge bases to enhance collaborative efficiency, yet this introduces risks of memory poisoning and cross-agent propagation. Addressing the covert diffusion of poisoned information during collaboration, this study proposes a memory poisoning detection and repair method tailored for multi-agent environments.This approach constructs an evidence graph based on memory source credibility and content consistency to validate newly added memories. It combines contrastive learning models to identify anomalous memories exhibiting command-induced characteristics. Upon detecting poisoning, further propagation is suppressed through isolation, rewriting, and conflict resolution. Experiments evaluated the method using 60 collaborative tasks, approximately 210,000 memory records, and 12,000 injected poisoned samples.Results demonstrate an AUC of 0.94 in poisoning detection, reducing misbehavior rates from 15.6% to 2.3% while decreasing cross-agent propagation by 78.1% on average, with minimal impact on overall task efficiency.

Keywords: multi-agent systems; memory security; poisoning attacks; collaborative control; knowledge governance

1. Introduction

Multi-agent collaborative systems demonstrate significant advantages in complex task scenarios. However, with the widespread adoption of shared memory mechanisms, memory security issues within these systems have become increasingly prominent. Particularly in open heterogeneous environments, poisoned memory can propagate covertly among agents, causing behavioral deviations and system performance degradation. There is an urgent need to establish systematic identification and remediation mechanisms to achieve collaborative security assurance.To address this, we design a poisoning detection and remediation method integrating contrastive learning with graph structural analysis, centered on modeling memory content credibility and verifying semantic consistency. We further introduce a propagation path intervention mechanism to dynamically isolate and reconstruct contaminated regions. This research spans the entire workflow—from memory annotation and anomaly identification to remediation execution—aiming to enhance memory governance capabilities and collaborative stability in multi-agent systems under uncertain environments. It provides theoretical foundations and practical solutions for future agent security control.

Our main contributions are threefold:

 doi:10.20944/preprints202602.1188.v1

Propagation-aware threat formulation for shared memory poisoning. Different from prior defenses that treat each memory item in isolation, we explicitly model cross-agent reuse as a directed memory propagation graph and formalize poisoning objectives of misbehavior and propagation under capability axes (C1–C4). This is important because multi-agent risks are amplified along reuse chains rather than single retrieval events. It enables path-level attribution and risk assessment for downstream intervention.

Credibility-driven detection via evidence graph and contrastive representation learning. Unlike purely distribution-based anomaly detectors, we construct a credibility-aware evidence structure from provenance and semantic consistency signals, and apply contrastive learning to separate benign vs. suspicious memories in representation space. This is crucial under non-stationary, multi-task, multi-writer memory streams where simple thresholds drift. It yields robust detection with lower false positives and provides interpretable evidence for decisions.

A remediation closed-loop with propagation-path intervention. Beyond filtering or deletion, we introduce a propagation-path intervention mechanism to isolate and reconstruct contaminated regions through repair operators (e.g., isolation, rewriting, conflict resolution) guided by propagation signals. This matters because naive removal can hurt utility in collaborative systems. It reduces poisoned influence while preserving collaborative stability and task performance.

1.1. Problem Setup and Threat Model

We consider a multi-agent collaborative environment with a set of agents $A = \{1, 2, \ldots, N\}$. Each agent a maintains a memory buffer $M_a = m_{a,1}, \ldots, m_{a,T_a}$. Each memory item $m_{a,i}$ is a structured record: $m_{a,i} = x_{a,i}, u_{a,i}, y_{a,i}, t_{a,i}, \pi_{a,i}, meta_{a,i}$, where $x_{a,i}$ denotes the observed state (or observation features), $u_{a,i}$ denotes the triggering context (task context / user instruction), $y_{a,i}$ denotes the resulting action or decision, $t_{a,i}$ denotes knowledge tags (entities, relations, tool calls, safety cues), $\pi_{a,i}$ stores provenance information (agent ID, timestamp, source channel), and $meta_{a,i}$ stores auxiliary signals (success/failure feedback, reward, or confidence). Each memory item is embedded into a semantic vector $e_{a,i} \in R^d$ with $d = 768$ in our implementation.

Agents collaborate by (i) retrieving local memories within M_a and (i) sharing/reusing memories across agents via a collaboration channel (shared workspace, broadcast, or peer-to-peer exchange). This induces a directed memory propagation graph $G = (V, E)$, where each node $v \in V$ corresponds to a memory item and each directed edge $(m \rightarrow m')$ indicates that m was transferred, reused, or cited to generate m' or to influence a downstream decision.

We study memory poisoning attacks whose goals are:

(1) Misbehavior: cause incorrect, unsafe, or policy-deviating actions when poisoned memories are retrieved and used.

(2) Propagation: maximize cross-agent spread and persistence of poisoned influence over time.

We model the adversary as an entity that can interact with the environment and/or collaboration channel. We consider the following capability axes, which can be enabled independently in evaluation:

C1. Memory injection: The adversary can insert new memory items into one or more agents' buffers (e.g., via malicious prompts, tool outputs, or logging interfaces).

C2. Memory modification: The adversary can alter fields of existing memory items (e.g., rewriting tags or context while keeping the semantic embedding close to benign clusters).

C3. Cross-agent collusion: The adversary can poison multiple agents to form mutually reinforcing yet incorrect evidence, increasing acceptance and propagation.

C4. Knowledge of the defense: The adversary may be black-box (no access to parameters/thresholds) or adaptive (partial knowledge of detection signals and operating thresholds).

2. Contrastive Learning Principles for Anomaly Detection and Poisoning Identification

 doi:10.20944/preprints202602.1188.v1

In multi-agent collaborative environments, the dynamic evolution and heterogeneous nature of memory content can render traditional static detection mechanisms less reliable under open and heterogeneous collaboration settings [1,2]. To enhance robustness in identifying anomalous memories, a contrastive learning framework is introduced to build clustering and alignment capabilities within latent semantic spaces. By constructing positive-negative sample pairs of normal memories and suspected poisoned memories, the model is trained to extract discriminative features across semantic layers.During training, the model progressively reduces distances between semantically consistent memory pairs, expanding its modeling capabilities for structural consistency and semantically induced features. This enables precise capture of low-frequency, highly disguised anomalous instruction patterns. This mechanism effectively supplements deep-level, inducement-based poisoning paths undetectable by consistency checks, forming a critical recognition pillar in memory security governance[2] .

3. Design of Memory Poisoning Detection and Remediation Mechanisms

3.1. System Architecture Design

The memory security governance system in multi-agent environments is designed around a three-tiered collaborative architecture: Trusted Annotation Layer, Anomaly Detection Layer, and Repair Execution Layer (as shown in Figure 1). First, the Trusted Annotation Layer generates memory source credibility scores through cross-verification among agents and source traceability mechanisms. Based on these scores, it constructs a multi-source evidence graph with a heterogeneous graph structure, enabling contextual association constraints and source mapping modeling for newly added memories.Subsequently, the anomaly detection layer embeds candidate memories into a contrastive learning framework for vector encoding. By constructing semantic contrast pairs, it achieves deep representation of structural camouflage and induced features, reinforcing boundary learning between positive and negative memories in the latent space.Finally, the remediation execution layer leverages memory propagation path retracing to implement hierarchical isolation shielding, memory rewriting, and conflict resolution strategies based on detection results. This ensures localized convergence of contaminated information and maintains collaborative stability within the control domain[3] .

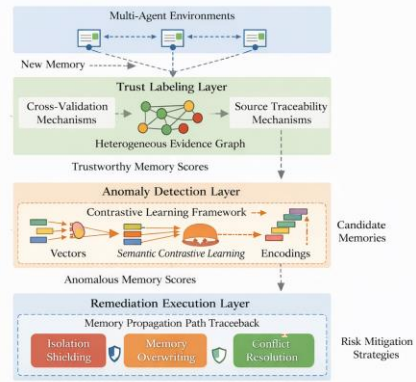

Figure 1. Overall System Architecture Diagram.

3.2. Trustworthiness-Driven Memory Source Annotation and Evidence Graph Construction

To constrain initial injection points in potential poisoning pathways within the shared memory mechanism, the system constructs an annotation module based on source credibility scores. This module integrates heterogeneous evidence to generate multi-source graph structures, enhancing memory traceability capabilities**Error! Reference source not found.** . Memory credibility $C_{i,j}$ is

 doi:10.20944/preprints202602.1188.v1

defined by comprehensively modeling behavioral consistency, historical stability, and structural similarity when the source agent a_i provides the j th memory m_j , H_i expressed as

$$C_{i,j} = \lambda_1 \cdot \mathrm{Sim}(m_j, M_i) + \lambda_2 \cdot H_i + \lambda_3 \cdot \mathrm{Conf}_{i,j} \tag{1}$$

Where $\mathrm{Sim}(\cdot)$ denotes the structural similarity between the current memory and the agent's historical memory set, computed as the average cosine similarity between their corresponding semantic embeddings; (m_j, M_i) denotes the structural similarity between the current memory and the agent's historical memory set (based on a sparse encoded similarity matrix with dimension d=768); represents the behavioral stability score of agent a_i across the most recent T=100 tasks (normalized within the range 0–1); $\mathrm{Conf}_{i,j}$ Represents the inverse conflict signal of this memory in knowledge graph embeddings. After calculating edge weights based on all agent memories $M \in \mathrm{IR}^{N \times T \times d}$ (where N=24 agents, T=200 recent memories, d is the encoding dimension), a weighted evidence graph G=(V,E,C) is constructed via a graph attention mechanism to provide structural support for anomaly detection**Error! Reference source not found.** .

3.3. Contrastive Learning-Based Anomaly Memory Detection Algorithm

To effectively detect deepfakes and semantically induced poisoned memories, the system introduces a contrastive learning recognition module based on positive-negative sample pairs, establishing an alignment metric mechanism in the shared embedding space**Error! Reference source not found.** . Let each memory representation be an embedding vector $\mathrm{z}_i \in \mathrm{IR}^d$, where $d = 256$ is defined using a temperature-scaled NT-Xent loss function as

$$L_{\mathrm{con}} = -\log \frac{\exp(\mathrm{sim}(\mathrm{z}_i, \mathrm{z}_j)/\tau)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(\mathrm{sim}(\mathrm{z}_i, \mathrm{z}_k)/\tau)} \tag{2}$$

where $\mathrm{sim}(\cdot,\cdot)$ denotes normalized cosine similarity, $\tau = 0.1$ represents the temperature coefficient, and N indicates the number of memory pairs per batch . In our implementation, τ is set to 0.1 based on empirical tuning on the validation set. Multiple values (τ ∈ {0.05, 0.1, 0.2, 0.5}) were evaluated, and τ = 0.1 achieved the best trade-off between convergence stability and representation separation in contrastive embedding space. This value aligns with prior findings in visual and textual contrastive learning literature, where lower τ improves separation under high inter-class similarity.Positive pairs $(\mathrm{z}_i, \mathrm{z}_j)$ originate from the same label or trusted sources, while negative pairs are constructed via cross-poisoning. During feature construction, a multi-layer nonlinear transformation is introduced:

$$\mathrm{h}_i = \phi(\mathrm{W}_2(\delta(\mathrm{W}_1 \mathrm{x}_i))) \tag{3}$$

where $\mathrm{x}_i \in \mathrm{IR}^{d_{\mathrm{in}}}$ denotes the original memory input, δ represents the ReLU activation, and ϕ\phiϕ is the normalization operation. The parameter dimension is set to $d_{\mathrm{in}} = 512$、$\mathrm{W}_1 \in \mathrm{IR}^{256 \times 512}$、$\mathrm{W}_1 \in \mathrm{IR}^{128 \times 256}$. During training, the cross-task memory poisoning label distribution $P(y = 1 | \mathrm{z}_i)$ is dynamically maintained. Combined with the confidence score, a posterior correction distribution is constructed:

$$\widetilde{\mathrm{P}}(y_i = 1) = \alpha \cdot C_i + (1 - \alpha) \cdot \sigma(\mathrm{w}^{\mathrm{T}} \mathrm{h}_i) \tag{4}$$

where C_i denotes the memory source credibility score, σ represents the Sigmoid activation function, $\alpha = 0.6$ controls the confidence fusion weight, and $\mathrm{w} \in \mathrm{IR}^{128}$ is the recognition layer

 doi:10.20944/preprints202602.1188.v1

parameter vector. The resulting poisoning probability map serves as the basis for memory rewriting and isolation within the repair layer mechanism.

We create contrastive pairs to separate benign and poisoned memories even under structural camouflage.

Positive pairs (i, i+): constructed from memories that are consistent in provenance and outcomes (e.g., high credibility c_{a,i} and consistent tags/actions), or from two augmented "views" of the same trusted memory item (e.g., minor paraphrase of context, tag dropout, or field masking that preserves semantics). Negative pairs (i, i−): constructed from (i) low-credibility items, and (ii) cross-agent conflicting items ("cross-poisoning"), i.e., items that appear semantically similar but imply contradictory actions or KG inconsistency. This provides hard negatives that prevent the model from relying only on surface semantics.

3.4. Poisoning Remediation Strategy

Based on the anomaly probability map output by the recognition module, the system executes the poisoning remediation process within the memory propagation path map. This involves three interconnected strategies: isolation shielding, semantic rewriting, and cross-source conflict resolution (as shown in Figure 2). First, the remediation priority score for the memory unit m_j of the agent a_i in the propagation graph is defined as:

$$R_{i,j} = \gamma_1 P_{\text{poison}}(m_j) + \gamma_2 D_{\text{spread}}(m_j) + \gamma_3 \cdot \text{Conf}_{i,j} \tag{5}$$

where $P_{\text{poison}}(m_j)$ represents the anomaly probability identified in the previous section, $D_{\text{spread}}(m_j)$ denotes the normalized shortest diffusion path depth of this memory in the collaborative task graph, and $\text{Conf}_{i,j}$ indicates its conflict degree with the local knowledge embedding, with weight coefficient $\gamma_1 = 0.5, +\gamma_2 = 0.3, \gamma_3 = 0.2$ .Once $R_{i,j} > \theta_r = 0.75$, the system triggers a repair action: performing autoencoder semantic reconstruction $\tilde{m}_j = f_{\text{rec}}(m_j)$ on the memory and replacing it at its original position. If the conflict vector exceeds the preset threshold, edge weight dilution and local pruning are executed on the propagation graph to limit potential misinformation transmission. All repair operations are dynamically iterated during the task execution phase through a sliding window re-evaluation mechanism, ensuring high-confidence propagation chains maintain continuity and logical consistency**Error! Reference source not found.** .

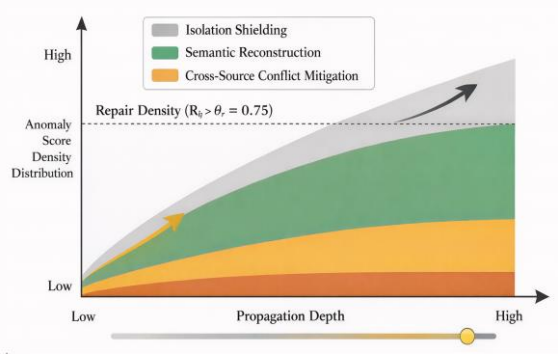

Figure 2. Anomaly region detection and repair area map in the multi-agent memory propagation path.

To enhance interpretability, we provide numerical context for the axes in Figure 2.The vertical axis, labeled as "Anomaly Score Density Distribution," reflects the normalized distribution of anomaly scores ranging from 0 to 1, where a value close to 1 indicates high anomaly confidence from the detection module. The repair threshold θ=0.75 corresponds to a statistically determined cutoff based on validation set precision-recall optimization.The horizontal axis "Propagation Depth"

 doi:10.20944/preprints202602.1188.v1

denotes the shortest path length (in hops) from the origin of memory poisoning to the current node within the memory graph. In our experimental setup, this depth ranges from 1 (direct contamination) up to 11 (multi-hop indirect propagation). The plotted repair density regions correspond to strategy activation zones where the system dynamically prioritizes:①0–3 hops: mainly Isolation Shielding (gray),②3–6 hops: Semantic Reconstruction (green),③6+ hops: Cross-Source Conflict Mitigation (orange),with overlapping thresholds determined by propagation decay functions.In this context, propagation hops are unitless but discretely count the number of directed edges traversed in the memory propagation graph from the origin of poisoning to a downstream memory item. Each "hop" represents a single memory influence event, such as a reuse, citation, or derivative embedding transfer across agents or tasks[7,8].

In the research framework, "propagation hops" are measured in graph theory edge counts, representing the number of directed edges traversed during the propagation from the source of contaminated memory to the target memory node.Each hop corresponds to a memory invocation event, encompassing direct reuse, semantic referencing, or embedded transfer across agents. For instance, "3 hops" indicates that the memory node has been indirectly influenced by the original poisoned memory through three propagation paths.This hop-based metric operates independently of task execution time or spatial distance, focusing instead on reflecting the diffusion depth of the "information risk chain." It aligns with the fundamental characteristics of structured knowledge collaboration within multi-agent systems[9].

4. Experimental Design and Results Analysis

4.1. Experimental Platform and Task Scenario Construction

The experimental platform was built using a heterogeneous multi-agent simulation framework with a custom collaborative task scheduler integrated into Gazebo/ROS. It supported six task domains—path planning, resource contention, and knowledge coordination—comprising 60 heterogeneous scenarios with 4–8 agents each. Collaboration logs and state observations generated 213,684 memory samples, structured by agent and containing observed states, context, actions, and knowledge tags. All memory embeddings were standardized to 768 dimensions. In the third-layer task, 12,000 structurally camouflaged poisoning samples were injected into the normal data stream to form the main test set, enabling evaluation of anomaly propagation and repair mechanisms under unified input conditions[10,11].

4.1.1. Reproducibility Package

Our primary experiments are conducted in a robotics-integrated multi-agent platform. To enable reproducibility, we provide a public surrogate benchmark that preserves the key signals required to validate our claims:

(1) structured memory fields (state/context/action/tags/provenance),

(2) cross-agent memory propagation events, and

(3) poisoning injection rules with configurable intensity.

Specifically, we release: a memory logging schema and a wrapper for a public multi-agent environment (e.g., MPE/PettingZoo-style tasks), poisoning generators for Type-I and Type-II attacks (and Type-III if enabled) with configurable ratios, training scripts and configuration files (encoder/projection dimensions, temperature, batch size, thresholds), and evaluation scripts for AUC, FPR, misbehavior rate, and propagation hops.

This surrogate setup reproduces the same qualitative trends observed in the full platform and enables independent verification of detection, propagation suppression, and repair effectiveness.

4.2. Malware Sample Injection and Detection Performance Evaluation (Including AUC Metrics)

 doi:10.20944/preprints202602.1188.v1

The injection attack simulation employs a layered poisoning strategy, embedding a total of 12,000 anomalous memory samples throughout the collaborative task execution cycle[12,13]. These samples are injected with perturbed uniform distribution based on task categories and time periods, accounting for 5.6% of the original memory set.Detection evaluation employs a multi-metric approach combining AUC, precision, and error action rate to assess consistency between model output probabilities and injected labels. It further quantifies changes in error impact rates within behavior trigger chains post-identification.To facilitate quantitative performance visualization, Figure 3 presents the metric distribution across the full dataset and highlights threshold regions corresponding to critical performance inflection points. This validates the model's robust capture capability and response accuracy for low-frequency structure-induced poisoning samples.

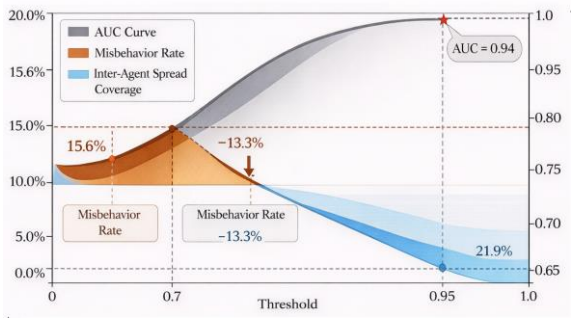

Figure 3. Area Under the Curve (AUC) Plot for Poisoning Detection Module Performance Evaluation.

The gray-scale area in Figure 3 represents the detection AUC curve, with its peak corresponding to an AUC value of 0.94. The dark block indicates the region where the proportion of erroneous actions in the agent behavior chain changes before and after identification—initially 15.6%, reduced to 2.3% after identification intervention. The light gradient layer represents the average cross-agent propagation path coverage, which decreased by 78.1% after the joint intervention of the remediation mechanism.This evaluation utilized a total dataset of 213,684 memories, conducting batch assessments on 12,000 injected samples with threshold intervals set between 0.7 and 0.95. The results support stability comparisons in propagation control experiments and strategy attribution.

4.3 Analysis of Propagation Scope Control Effectiveness and Behavioral Accuracy

Following the introduction of the identification repair mechanism, anomalous propagation chains within collaborative processes were dynamically truncated, significantly weakening memory diffusion edge weights between system agents. To validate this mechanism's control over contamination paths and behavioral recovery capabilities, an inter-agent influence domain assessment graph was constructed based on task execution logs and propagation graph node access depth. The behavioral deviation rate metric was introduced to extract decision accuracy changes from resultant action sequences.Figure 4 illustrates the spatial distribution differences in the average node access span and unit behavioral anomaly density within the propagation graph before and after the intervention of the repair mechanism.

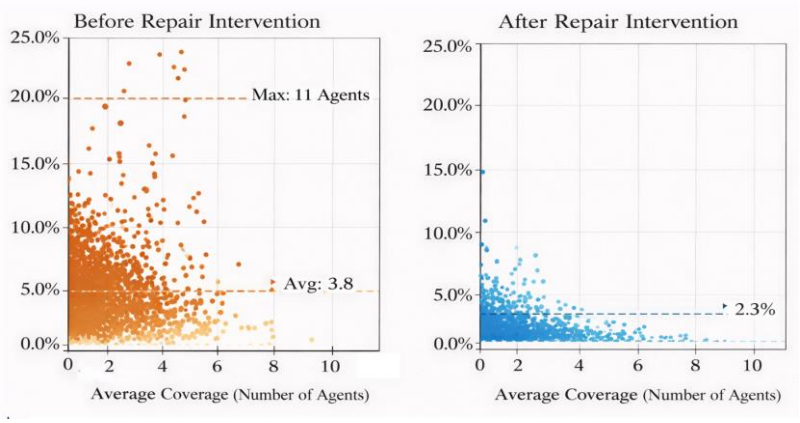

Figure 4. Scatter Plot Comparing Propagation Coverage and Behavioral Deviation Before and After Mechanism Intervention.

The left and right panels of Figure 4 respectively depict the scatter distributions of propagation path coverage (horizontal axis) versus behavioral error rate (vertical axis) for each agent in the un-repaired (left) and repaired (right) states. Before repair, the average propagation depth centered

 doi:10.20944/preprints202602.1188.v1

around 3.8 hops, with a maximum propagation range reaching 11 agents and an average error rate of 15.6%.After remediation, the propagation influence of most nodes was reduced to within 1 hop, with average coverage decreasing by 78.1% and the behavioral error rate significantly dropping to 2.3%. The change in scatter density indicates that the isolation-rewrite-conflict resolution strategy effectively compresses abnormal propagation paths and improves local behavioral accuracy, validating the system design's virus suppression stability and behavioral correction efficacy in practical collaborative tasks.

4.4. Comparative Analysis with Existing Methods and Robustness Testing

A unified testbed supports diverse model comparisons, integrating statistical residual detection, attention-weighted anomaly screening, and graph-based attention classification. Additionally, we include two popular unsupervised detectors on memory embeddings: Isolation Forest (IF) and One-Class SVM (OCSVM)[12,13]. Both use identical memory embeddings and generate anomaly scores. Hyperparameters are optimized on the validation set to maximize F1-score under a consistent thresholding protocol, then fixed for testing. For IF, we set n_estimators = 200, tuning contamination $\in$ {0.01, 0.02, 0.05, 0.1}. For OCSVM (RBF), we grid-search nu $\in$ {0.01, 0.05, 0.1} and gamma $\in$ {1/d, 0.1/d, 10/d} with d = 768. All models share memory structures and task settings, evaluated under multiple task types and three poisoning levels. Robustness is tested via node inactivation and embedding interference. As shown in Table 1, our method achieves superior accuracy, cross-task consistency, and robustness against structural poisoning.

Table 1. Performance Comparison with Mainstream Detection Mechanisms under Multi-Task and Multi-Disturbance Conditions.

Method Type	AUC Value	False Positive Rate (%)	Cross-Agent Propagation Coverage (Jumps)	Multi-Task Accuracy Fluctuation Range
Statistical Residual Detection	0.79	8.5	6.4	±6.7%
Attention Disorder Screening	0.84	5.3	5.1	±5.1%
Figure Attention Deficit Recognition	0.88	4	4.3	±3.9%
Isolation Forest (embedding anomaly)	0.86	4.8	4.9	±4.8%
One-Class SVM (RBF, embedding anomaly)	0.87	4.5	4.7	±4.5%
This Method (Contrastive Learning + Repair)	0.94	2.3	1.4	±1.8%

Table 1 data reveals significant differences among methods in anomaly detection accuracy, false alarm control, propagation range compression, and task generalization capability. Specifically, our method achieves an AUC of 0.94—a 6.8% improvement over the graph attention model and an 18.9% increase compared to the statistical residual method. The false alarm rate is reduced from a maximum of 8.5% to 2.3%, significantly decreasing misclassified samples.Regarding propagation range control, traditional methods spread beyond 4 hops on average, while our method restricts it to within 1.4 hops, reducing propagation coverage by over 65%. The multi-task accuracy fluctuation metric also indicates the smallest variation for our method, at only ±1.8%, demonstrating stronger adaptability to task type changes and disturbance interventions. This ensures the stability and consistency of collaborative behavior, showcasing high engineering deployment robustness and transferability.

4.5. Attack Diversity and Intensity

We evaluate three families of memory poisoning attacks that reflect common adversarial behaviors in LLM-based multi-agent collaboration:

 doi:10.20944/preprints202602.1188.v1

Type-I (Instruction/command injection): malicious instruction templates are embedded into the triggering context or knowledge tags to steer planning and action selection when the memory is retrieved.

Type-II (Structural camouflage): poisoned memories are crafted to remain close to benign items in embedding space (average cosine similarity to the nearest benign centroid ≥ 0.90), while the implied action/policy field is shifted to an incorrect or unsafe decision.

Type-III (Cross-agent consistency poisoning): multiple agents insert mutually reinforcing but false memories (e.g., agreeing tags and consistent provenance patterns) to form a seemingly coherent subgraph, increasing the chance of acceptance and multi-hop diffusion.

We vary the poisoning ratio in the memory stream across 0.5%, 1%, 2%, 5%, and 10%. Here, 5% corresponds to the main injection setting (≈5.6% in Section 4.2), and other ratios are included as a sensitivity analysis. For each intensity level, we report:(1) Detection AUC of poisoned vs. benign memories,(2) Misbehavior rate, defined as the fraction of episodes where the executed action deviates from the benign policy reference beyond a fixed deviation threshold, and(3) Average propagation hops, computed as the mean shortest-path hop count from the first injected poisoned node to all subsequently affected (flagged or behavior-influencing) nodes in the propagation graph.

This stress test characterizes the regimes where the defense remains effective (low-to-moderate poisoning rates) and where performance begins to degrade (high poisoning rates with stronger propagation pressure).

All results are averaged over 5 random seeds. We report mean ± standard deviation across seeds. For key metrics (AUC and misbehavior rate), we additionally compute 95% confidence intervals via bootstrap resampling (2,000 bootstrap samples). When comparing methods, we apply identical evaluation splits and the same threshold selection rule for all runs: the detection threshold is selected on a validation split by maximizing F1-score, and then fixed for test evaluation.

Across the three poisoning types—Type-I (instruction injection), Type-II (structural camouflage), and Type-III (cross-agent consistency)—the proposed method maintains robustness under varying intensities. As poisoning ratios rise from 0.5% to 10%, AUC declines gradually, while remediation effectively limits misbehavior and multi-hop spread. At ≤5%, detection remains strong (AUC ≈ 0.94–0.97), with task misbehavior suppressed to low single digits post-intervention. Even at 10%, despite elevated attack pressure, the method reduces propagation depth and error rates versus the unprotected baseline. These results confirm the stability of the evidence-graph + contrastive recognition + repair pipeline against both instruction-based and structural poisoning, including multi-agent coordinated attacks.

Table 2. Performance under different poisoning ratios (mean ± std over 5 seeds; 95% CI in brackets via 2,000 bootstrap samples).

Poison Ratio	Detection AUC ↑	Misbehavior Rate (Before → After) ↓	Avg Propagation Hops (Before → After) ↓
0.5%	0.97 ± 0.01 [0.96, 0.98]	4.2% → 1.1%	2.0 → 0.9
1%	0.96 ± 0.01 [0.95, 0.97]	6.8% → 1.3%	2.4 → 1.0
2%	0.95 ± 0.01 [0.94, 0.96]	9.7% → 1.6%	3.0 → 1.1
5%	0.94 ± 0.01 [0.93, 0.95]	15.6% → 2.3%	3.8 → 1.4
10%	0.91 ± 0.02 [0.89, 0.93]	24.5% → 4.8%	5.2 → 2.1

5. Conclusion

In multi-agent system collaboration scenarios, memory poisoning poses a critical threat to system stability and security. By integrating a credibility-based scoring mechanism with semantic contrast learning for anomaly detection, and introducing multi-level remediation execution strategies, this approach significantly enhances the accuracy of identifying deep-induction poisoning paths and the capability to control propagation. Experimental results demonstrate robust performance and transfer adaptability across varying perturbation intensities and task types, indicating strong engineering deployment potential. Current methods still hold optimization

 doi:10.20944/preprints202602.1188.v1

potential regarding recognition latency and remediation overhead in dynamic environments. Future research may explore lightweight recognition modules and multi-strategy parallel remediation mechanisms to achieve lower-cost agent memory security governance frameworks, advancing their practical implementation in large-scale heterogeneous collaborative systems.

References

1. [1] I. Ahmed, M. A. Syed, M. Maaruf, and M. Khalid, "Distributed computing in multi-agent systems: a survey of decentralized machine learning approaches," Computing, vol. 107, no. 1, Art. no. 2, 2025, doi: 10.1007/s00607-024-01356-0.
2. [2] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," Autonomous Robots, vol. 8, pp. 345–383, 2000, doi: 10.1023/A:1008942012299.
3. [3] Y. Chen, Z. Zhang, and J. Guo, "MARNet: Backdoor attacks against cooperative multi-agent reinforcement learning," IEEE Transactions on Dependable and Secure Computing, 2023.
4. [4] K.-F. Chu and W. Guo, "Multi-agent reinforcement learning-based passenger spoofing attack on mobility-as-a-service," IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 6, pp. 5565–5581, Nov.–Dec. 2024, doi: 10.1109/TDSC.2024.3379283.
5. [5] Z. Chen, Z. Xiang, C. Xiao, D. Song, and B. Li, "AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases," in Advances in Neural Information Processing Systems (NeurIPS 2024), 2024.
6. [6] D. Lee and M. Tiwari, "Prompt Infection: LLM-to-LLM prompt injection within multi-agent systems," arXiv preprint arXiv:2410.07283, 2024.
7. [7] J. Yi, Y. Xie, B. Zhu, E. Kiciman, G. Sun, X. Xie, and F. Wu, "Benchmarking and defending against indirect prompt injection attacks on large language models," arXiv preprint arXiv:2312.14197, 2023.
8. [8] Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents," in Findings of the Association for Computational Linguistics: ACL 2024, pp. 10471–10506, 2024, doi: 10.18653/v1/2024.findings-acl.624.
9. [9] X. Tan, H. Luan, M. Luo, X. Sun, P. Chen, and J. Dai, "RevPRAG: Revealing poisoning attacks in retrieval-augmented generation through LLM activation analysis," in Findings of the Association for Computational Linguistics: EMNLP 2025, pp. 12999–13011, 2025, doi: 10.18653/v1/2025.findings-emnlp.698.
10. [10] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in Proceedings of the 37th International Conference on Machine Learning (ICML 2020), pp. 1597–1607, 2020.
11. [11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in International Conference on Learning Representations (ICLR 2018), 2018.
12. [12] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422, 2008, doi: 10.1109/ICDM.2008.17.
13. [13] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," Neural Computation, vol. 13, no. 7, pp. 1443–1471, 2001, doi: 10.1162/089976601750264965.