CASPIAN: Online Detection and Attribution of Cascade Attacks in LLM Multi-Agent Systems via Cross-Channel Causal Monitoring

Kavana Venkatesh **Jafar Isbarov** **Saad Amin**
Murat Kantarcioglu **Jiaming Cui**
Virginia Tech, Blacksburg, VA, USA
{kavanav,isbarov,saad05,muratk,jiamingcui}@vt.edu
github.com/caspian-detector/caspian

Abstract

Cascade attacks in LLM multi-agent systems (MAS) arise when adversarial influence propagates across agents and escalates into system-level failures through complex agent interactions. Detecting such cascades is challenging, as their signals are distributed, tightly coupled across interaction channels, and often appear plausibly benign locally, while unfolding either abruptly within a single turn or gradually across multiple turns. Existing defenses, being largely local and text-centric, fail to capture such cross-channel, temporally coordinated dynamics of cascade propagation. We propose CASPIAN, a unified framework for online detection and attribution of cascade attacks in LLM-MAS through cross-channel causal influence monitoring. CASPIAN models multi-agent interactions using a dynamic causal influence tensor across communication, memory, tool, and execution channels, estimated efficiently via late-interaction conditional transfer entropy (LI-CTE). This enables detection of cascade onset from emergent system-level structure rather than isolated anomalies. CASPIAN further performs online attribution, identifying the origin, bridge, and amplifier agents driving the cascade and reconstructing its principal propagation pathways directly from cached influence dynamics. Across diverse multi-agent frameworks and benchmarks, CASPIAN consistently outperforms semantic guardrails, LLM-based judges, and graph-based anomaly detectors in both detection accuracy and early cascade identification while operating with sub-1% relative latency overhead. These results demonstrate the effectiveness of unified cross-channel causal monitoring for detecting and understanding cascade failures in LLM multi-agent systems.

1 Introduction

As large language models are increasingly deployed within collaborative multi-agent systems (LLM-MAS) [41, 9, 37, 36], the nature of adversarial failures has fundamentally shifted. While single-agent systems are vulnerable to isolated hallucinations or prompt injection attacks, multi-agent architectures are susceptible to *cascade attacks*: coordinated failures in which adversarial or erroneous influence injected at one agent propagates, amplifies, and self-reinforces across agents and turns [42, 18]. These failures exploit the interaction structure of the system itself: a single injected prompt can corrupt shared memory, redirect tool outputs, and synchronize agents toward a collectively flawed trajectory, producing system-wide failures that are difficult to contain once established [42, 5]. Cascades manifest in two regimes: *abrupt* cascades, where strong amplification and synchronization produce immediate system-wide coupling within a single interaction turn; and *gradual* cascades, where locally benign influence accumulates across turns through persistent memory reuse, tool artifact propagation, and slow semantic drift [42, 13, 1].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

The central challenge is that cascade evidence is distributed across both interaction channels and turns. Because influence may propagate concurrently through communication, memory, tool, and execution interactions, any single channel provides only a partial view of the underlying influence dynamics [5]. At the same time, influence accumulates gradually through agent context, so behavior at a given interaction turn may still reflect residual effects of injections occurring many turns earlier. As a result, agents may appear locally benign even as the system collectively transitions toward coordinated failure. Detecting cascades therefore requires monitoring how influence propagates jointly across agents, channels, and time rather than inspecting interactions in isolation.

Existing defenses fail to provide this view. Semantic guardrails [2, 46, 32] and LLM-based judges inspect message content per-turn, remaining blind to cross-channel propagation structure and temporal accumulation. Graph-based approaches [38, 23, 47] model agent interactions over communication graphs and detect anomalous agents through representation-space irregularities, but largely operate on a single interaction channel and do not capture how influence propagates jointly across memory, tool, and execution pathways. Existing attribution methods further rely primarily on post-hoc trace analysis over completed executions [45, 39, 43], providing limited support for online intervention during active propagation. The shared limitation across these approaches is that they reason locally about prompts, agents, or static graphs, whereas cascade attacks are fundamentally global propagation processes. CASPIAN therefore shifts cascade detection from localized semantic inspection to online monitoring of propagation dynamics.

The key insight of this work is that cascade onset constitutes a structural reorganization of inter-agent influence propagation in a multi-agent system. Rather than arising from isolated anomalous interactions, cascades emerge through the coupled evolution of four propagation dynamics: growing influence energy (*amplification*), increasing alignment between propagation modes (*synchronization*), spread across interaction channels (*cross-channel propagation*), and sustained reinforcement across turns (*persistence*) [5, 42, 35]. Spectral analysis has a well-established role in studying cascade behavior in networked systems [49, 26], and provides a unified mechanism for tracking these dynamics through the evolving influence topology. Changes in the separation between dominant propagation modes provide a structural coupling signal, while amplification, cross-channel spread, and temporal persistence provide complementary evidence of cascade formation. In contrast, benign collaboration may temporarily increase interaction intensity without sustained coupling or persistent cross-channel spread [6]. Because spectral analysis operates directly on the evolving topology, it requires no supervision or topology-specific calibration.

We present **CASPIAN**, the *first framework* for unified online detection and attribution of cascade attacks in LLM-based multi-agent systems through cross-channel causal propagation monitoring. Unlike prior approaches operating on static communication graphs, CASPIAN models the MAS as an evolving cross-channel causal influence topology spanning communication, memory, tool, and execution interactions. This topology is estimated online using a Late-Interaction Conditional Transfer Entropy (LI-CTE) formulation [16] that captures directed, history-aware influence propagation with millisecond-scale overhead. CASPIAN performs spectral monitoring over the evolving topology to detect cascade onset through amplification growth, tightening spectral coupling, phase shifts, and cross-channel spread. Together, these signals distinguish transient activity from self-reinforcing cascade formation, enabling detection of both abrupt single-turn and gradual multi-turn cascades without attack templates, benign calibration traces, or offline training. Upon detection, CASPIAN performs online attribution directly from cached influence dynamics, identifying the cascade origin, amplifier, bridge, and dominant propagation spines responsible for system-wide spread. Across diverse multi-agent frameworks and cascade attack settings, CASPIAN consistently outperforms semantic guardrails, LLM-based judges, and graph-based anomaly detectors in both detection accuracy and early cascade identification while operating with sub-1% relative overhead latency.

- We introduce the *first unified framework* for modeling cross-channel causal influence in LLM multi-agent systems, jointly capturing directed inter-agent influence across communication, memory, tool, and execution channels.
- We propose an online spectral cascade detection framework that identifies both abrupt single-turn and gradual multi-turn cascades from evolving inter-agent influence dynamics, without attack templates, benign calibration traces, or offline training, with sub-1% relative overhead.
- We introduce an online cascade attribution scheme that identifies cascade origins, amplifiers, bridges, and dominant propagation paths directly from cached influence dynamics at detection time, supporting timely investigation and intervention without replay or recomputation.

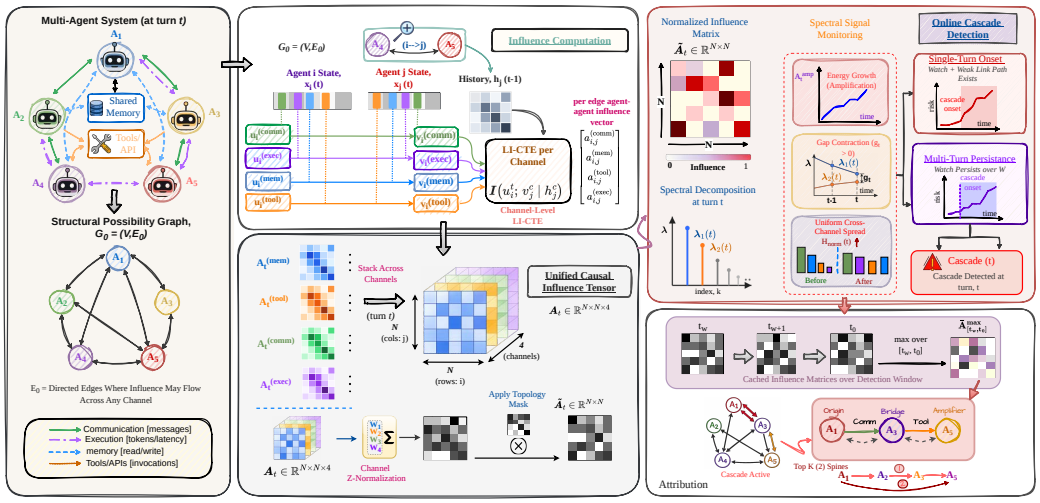


Figure 1: **Overview of CASPIAN.** CASPIAN constructs a unified cross-channel causal influence tensor from communication, memory, tool, and execution interactions using an efficient LI-CTE-based influence estimation. Spectral monitoring over the evolving topology detects abrupt and persistent cascade formation online, while cached influence dynamics enable recovery of cascade agents, and dominant spines at detection time, with sub-1% relative overhead latency.

2 Related Work

2.1 Cascade Failures and Vulnerabilities in LLM-MAS

LLM-MAS are vulnerable to cascading failures as errors or adversarial signals propagate and amplify across agents and turns. Prior work identifies cascade amplification, topology sensitivity, and consensus inertia as key failure drivers [42, 31]. Benchmarks such as TAMAS [14], AgentLAB [13], and ACIArena [1] expose long-horizon injection and agent-compromise vulnerabilities, while studies highlight memory faults, self-replicating attacks, and cross-channel propagation [48, 18, 21, 5]. However, existing work primarily characterizes attack outcomes, localized anomalies, or static interaction structures rather than evolving cross-channel propagation during execution.

2.2 Detection and Defense in LLM-MAS

Existing LLM-MAS defenses span three categories: *semantic guardrails*, including Prompt-Guard 2 [2], JailGuard [46], and PromptArmor [32], alongside perplexity-based anomaly detection [25]; *LLM-based judges* [7], which inspect interaction windows or traces but remain text-centric; and *graph-based detectors*, including G-Safeguard [38], BlindGuard [23], GUARDIAN [47], and CoopGuard [20], which capture structural and temporal anomalies primarily through communication graphs. Failure attribution largely relies on post-hoc trace analysis [45, 39, 44, 43], complemented by governance frameworks [42, 12]. Runtime approaches such as SentinelAgent [8] use dynamic execution graphs for anomaly detection and root-cause explanation, while AEGIS [17] supports learned attribution through labeled failure trajectories. In contrast, CASPIAN provides training-free, online cross-channel cascade detection with propagation-role attribution at detection time.

2.3 Causal Influence and Spectral Monitoring

Prior work has used transfer entropy and related information-theoretic measures to estimate directed influence in complex systems [27, 29, 34, 18]. Spectral methods have similarly been used to analyze propagation and stability in networked systems [3, 24, 28, 11, 10, 19, 26, 6], while path-based models characterize how failures spread through network bottlenecks and interaction pathways [40, 15]. However, these approaches have not been integrated into an online framework for monitoring cross-channel cascade formation in LLM-MAS. CASPIAN combines causal influence estimation and spectral monitoring to detect and attribute cascade propagation across communication, memory, tool, and execution interactions during system execution.

3 Background

3.1 Cascade Failures in LLM Multi-Agent Systems

LLM-MAS operate through iterative interactions, where agent outputs become inputs to downstream agents across multiple turns, inducing directed dependencies through which influence propagates and reinforces over time [5, 42]. A key challenge is the emergence of **cascade failures**, where erroneous or adversarial signals become progressively self-reinforcing across agents and turns. This behavior is driven by four mechanisms: ① **Amplification**, where downstream agents reuse corrupted intermediate states [48, 13]; ② **Cross-Channel Propagation**, where influence spreads across communication, memory, and tool interactions [1, 5]; ③ **Persistence**, where signals remain embedded across turns [42]; and ④ **Synchronization**, where agents converge toward coupled reasoning trajectories [42, 5]. These dynamics produce both abrupt **Single-turn Cascades** and gradual **Multi-turn Cascades** [1, 13, 48]. Distinguishing transient activity from sustained cascade formation requires analyzing evolving propagation dynamics instead of isolated agent outputs [40, 15].

3.2 Spectral Structure of Influence

A multi-agent system with N agents can be represented by a directed influence matrix $A_t \in \mathbb{R}^{N\times N}$ at turn t, where $A_t(i,j)$ quantifies the influence of agent i on agent j. Its spectral structure characterizes the relative strengths of propagation modes within the system [3, 24]. Let $\lambda_1(t) \geq \lambda_2(t)$ denote the two largest singular values of the directed influence operator. Here, $\lambda_1(t)$ captures the dominant propagation magnitude [28, 11], while $\lambda_2(t)$ captures the strength of the secondary mode [3]. Their separation, $\lambda_1(t) - \lambda_2(t)$, characterizes the relative prominence of these modes: a larger gap indicates a more dominant leading mode, whereas gap contraction indicates that secondary influence structure is becoming comparably stronger [10, 19, 33]. However, gap contraction alone does not establish persistent propagation [26, 6]. CASPIAN therefore monitors spectral coupling jointly with amplification, cross-channel spread, and temporal persistence to distinguish transient activity from cascade-like structural transitions.

4 Methodology

4.1 Problem Setup

We consider an LLM-based multi-agent system with agent set $V = \{0, \ldots, N-1\}$ operating over discrete interaction turns $t = 1, \ldots, T$. At each turn, agents interact through channels $\mathcal{C} = \{\mathrm{comm}, \mathrm{mem}, \mathrm{tool}, \mathrm{exec}\}$, corresponding to inter-agent communication, shared memory access, tool-mediated outputs, and execution-level behavior. The system produces an online interaction trace up to turn t, denoted by $\mathcal{D}_{1:t}$, consisting of communication traces, memory operations, tool invocations, and execution metadata. We define:

Definition 1. *A **cascade attack** is a self-reinforcing propagation process in which adversarial or erroneous influence spreads across agents through amplification, persistence, and synchronization, either within a single interaction channel or across multiple channels over one or more turns.*

Accordingly, we model the system through a unified nonnegative causal influence tensor $\mathcal{A}_t \in \mathbb{R}_{\geq 0}^{N\times N\times|\mathcal{C}|}$, where $\mathcal{A}_t(i,j,c)$ denotes the directed influence of agent i on agent j through interaction channel c at turn t. Larger values indicate stronger directed dependence. The tensor $\mathcal{A}_t$ is latent and must be inferred online from the observed trace $\mathcal{D}_{1:t}$. Under this formulation, cascade onset corresponds to coordinated structural changes in the evolution of $\mathcal{A}_t$.

CASPIAN solves two inference objectives over $\mathcal{A}_t$. **(i) Detection**: infer whether the system has entered a cascade state, $\mathrm{Cascade}(t) \in \{0, 1\}$, and classify it as either an abrupt *Single-turn Cascade* or gradual *Multi-turn Cascade*. **(ii) Attribution**: upon detection, recover from cached influence dynamics, (i) the *origin*, where abnormal influence first enters the system; (ii) the *amplifier*, which most strongly reinforces propagation; (iii) the *bridge*, which redistributes influence across the network; and (iv) the principal *propagation spines* through which the cascade spreads. Algorithm 1 details our method.

4.2 Unified Causal Influence Modeling

We model inter-agent influence through a unified channel-aware causal influence tensor, enabling both global spectral analysis and channel-level reasoning. We construct a structural possibility graph $G_0 = (V, E_0)$ encoding the interaction topology, where an edge $(i \to j) \in E_0$ exists if agent i can influence agent j through at least one interaction channel $c \in \mathcal{C}$, where $\mathcal{C} = \{\texttt{comm}, \texttt{mem}, \texttt{tool}, \texttt{exec}\}$.

For each feasible edge $(i, j) \in E_0$, channel c, and turn t, CASPIAN estimates directed influence using a temporally-adapted conditional transfer entropy (CTE) formulation. Direct CTE computation over full agent states is noisy and computationally expensive, so we adopt a late-interaction formulation inspired by ColBERT-style decomposition [16], introducing only millisecond-scale overhead:

$$a_{ij}^{(c)}(t) = \mathcal{I}\Big(\mathbf{u}_{i\to j}^{(c)}(t)\,;\, \mathbf{v}_{i\to j}^{(c)}(t) \mid \mathbf{h}_j^{(c)}(t-1)\Big)\,, \tag{1}$$

where $\mathbf{u}_{i\to j}^{(c)}(t)$ and $\mathbf{v}_{i\to j}^{(c)}(t)$ are compact source and target projections along channel c, and $\mathbf{h}_j^{(c)}(t-1)$ is an exponential moving average of the target agent's historical states.

The channel-wise influences are aggregated as

$$\mathbf{a}_{ij}(t) = \big[a_{ij}^{(\texttt{comm})}(t),\ a_{ij}^{(\texttt{mem})}(t),\ a_{ij}^{(\texttt{tool})}(t),\ a_{ij}^{(\texttt{exec})}(t)\big], \tag{2}$$

and stacked to form the causal influence tensor $\mathcal{A}_t \in \mathbb{R}^{N\times N\times|\mathcal{C}|}$, where $\mathcal{A}_t(i, j, :) = \mathbf{a}_{ij}(t)$.

For spectral analysis, channel-wise influence is aggregated into a raw influence matrix $A_t(i, j) = \sum_{c\in\mathcal{C}} \mathcal{A}_t(i, j, c)$. We apply the structural topology mask and degree-aware normalization:

$$\tilde{A}_t(i, j) = \frac{A_t(i, j)}{\sqrt{r_i(t)c_j(t) + \epsilon}},$$

where $r_i(t)$ and $c_j(t)$ denote outgoing and incoming influence strengths, respectively. The tensor $\mathcal{A}_t$ preserves channel-specific influence structure, while $\tilde{A}_t$ supports spectral monitoring and cascade attribution. We use causal influence in an operational, trace-conditioned sense. A trace-conditioned causal interpretation assumes reliable temporal ordering, correct structural admissibility, sufficiently informative target history, faithful observation of relevant events, and the absence of dominant unobserved confounding. Without these assumptions, LI-CTE should be interpreted as directed history-conditioned predictive dependence rather than interventional causality. See Appendix I for details.

4.3 Cascade Detection via Spectral Phase Dynamics

After each interaction turn t, the unified causal influence tensor $\mathcal{A}_t$ is updated using adaptive cumulative LI-CTE estimates over agent pairs and interaction channels. CASPIAN then detects cascades through spectral transitions in the normalized matrix $\tilde{A}_t$, characterized by growing influence, increasing spectral coupling, cross-channel propagation, and persistence. Detection proceeds in two stages: per-turn signal measurement followed by cascade classification.

4.3.1 Per-turn Signal Measurement

Amplification: Let $\lambda_1(t) \geq \lambda_2(t)$ denote the two largest singular values of $\tilde{A}_t$, representing the dominant and secondary propagation modes. We define the dominant spectral magnitude $E_t = \lambda_1(t) + \lambda_2(t)$ and its turn-over-turn growth as:

$$A_t^{\text{amp}} = \frac{E_t}{E_{t-1} + \epsilon}. \tag{3}$$

$A_t^{\text{amp}} > 1$ reflects growing influence, but may also arise from benign activity bursts [26].

Structural coupling: The distinguishing factor between activity and cascades lies in how influence is organized across propagation modes. Hence, we define spectral coupling ratio R_t, normalized spectral gap g_t, and gap contraction Δg_t as below:

$$R_t = \frac{\lambda_2(t)}{\lambda_1(t) + \epsilon}, \qquad g_t = 1 - R_t, \qquad \Delta g_t = g_{t-1} - g_t. \tag{4}$$

A positive Δg_t indicates reduced separation between the dominant and secondary propagation modes, providing a signal of increasing structural coupling. Cascade onset corresponds to a transition in propagation structure rather than gradual drift [6]. To capture this, we define:

$$\Phi_t = \frac{|R_t - R_{t-1}|}{R_{t-1} + \epsilon}, \qquad \text{PhaseShift}(t) = \mathbf{1}[\Phi_t > \Delta g_t]. \tag{5}$$

Phase shift marks entry into a new structural propagation dynamics.

Cross-channel propagation: Cascades in LLM-MAS extend across interaction channels, increasing both reach and persistence [1, 5]. Let $e_c(t) = \lambda_1(\tilde{A}_t^{(c)})$, and $p_c(t) = \frac{e_c(t)}{\sum_{c'} e_{c'}(t)+\epsilon}$. We measure spread using normalized Shannon entropy [30]:

$$H_t^{\text{norm}} = \frac{-\sum_{c\in\mathcal{C}} p_c(t)\log p_c(t)}{\log|\mathcal{C}|}, \qquad \text{CrossChannel}(t) = \mathbf{1}[H_t^{\text{norm}} \geq 0.5]. \tag{6}$$

Higher entropy indicates that propagation energy is distributed across multiple interaction channels rather than concentrated in a single modality.

4.3.2 Cascade Classification

The signals above are combined into a *watch condition* marking the earliest onset of cascade-like dynamics. Watch activates when amplification coincides with tightening spectral coupling and growth in the dominant propagation mode:

$$\text{Watch}(t) = \mathbf{1}[A_t^{\text{amp}} > 1 \ \wedge \ \Delta g_t > 0 \ \wedge \ \lambda_1(t) > \lambda_1(t-1)]\,. \tag{7}$$

This requires propagation growth to be driven by the dominant spectral mode rather than redistribution across weaker propagation modes.

Single-turn cascades: A single-turn cascade occurs when Watch coincides with either a phase shift or cross-channel spread, together with a feasible propagation route. Let t_w denote the first turn where $\text{Watch}(t)$ activates. For single-turn cascades, $t_0 = t_w$. CASPIAN declares:

$$\boxed{\text{InstantCascade}(t_0) = \Big[\text{Watch} \ \wedge \ (\text{PhaseShift} \vee \text{CrossChannel}) \ \wedge \ \text{WeakLink}\Big](t_0)} \tag{8}$$

where WeakLink verifies the existence of a viable propagation path.

Multi-turn cascades: Gradual cascades arise when locally weak signals persist and reinforce across turns. If Watch activates at turn t_w but the immediate rule is not satisfied, CASPIAN initializes an adaptive persistence interval governed by the inverse spectral gap [19]: $W_{t_w} = \left\lceil \frac{1}{g_{t_w}+\epsilon} \right\rceil, \qquad t_0 = t_w + W_{t_w} - 1$. A multi-turn cascade is declared when Watch holds for at least half of the interval and at least one PhaseShift or CrossChannel event occurs:

$$\boxed{\text{MultiTurnCascade}(t_0) = \Big[\text{MajorityWatch} \ \wedge \ \text{Transition}\Big](t_w, W_{t_w})} \tag{9}$$

where Transition denotes the presence of at least one PhaseShift or CrossChannel event within $[t_w, t_0]$. Failed intervals are discarded and reinitialized at the next Watch onset, while influence matrices are cached for attribution. The final cascade decision is: $\text{Cascade}(t_0) = \text{InstantCascade}(t_0) \ \vee \ \text{MultiTurnCascade}(t_0)$. This formulation separates transient activity bursts from true cascades by requiring either immediate structural confirmation or sustained temporal support.

4.4 Cascade Attribution via Spectral Flow Decomposition

Upon detection at turn t_0, CASPIAN performs online attribution directly from cached influence matrices without replay or recomputation. Attribution operates over the onset snapshot $\tilde{A}_{t_w}$ and the retained interval $\{\tilde{A}_\tau, A_\tau\}_{\tau=t_w}^{t_0}$, where t_w denotes the first Watch activation. For single-turn cascades, $t_w = t_0$.

The **Origin** is the agent with maximum outgoing influence at onset, $i^{\text{origin}} = \arg\max_i \sum_j \tilde{A}_{t_w}(i,j)$.

The **Amplifier** is the agent that most strongly reinforces propagation relative to received influence over $[t_w, t_0]$:

$$i^{\text{amp}} = \arg\max_i \sum_{\tau=t_w}^{t_0} \frac{\sum_j \tilde{A}_\tau(i,j)}{\sum_k \tilde{A}_\tau(k,i) + \epsilon}.$$

The **Bridge** is the agent that most strongly receives and redistributes influence, computed using unnormalized matrices to avoid suppressing hub-adjacent edges:

$$i^{\text{bridge}} = \arg\max_i \sum_{\tau=t_w}^{t_0} \left(\sum_j A_\tau(i,j) \right) \left(\sum_k A_\tau(k,i) \right).$$

We extract the **top-K candidate propagation spines** from the interval maximum influence matrix $\bar{A}^{\max}(i,j) = \max_{\tau \in [t_w, t_0]} \tilde{A}_\tau(i,j)$, which captures the strongest observed influence along each edge during cascade formation. Spines are ranked by their weakest-link strength:

$$\Pi^K = \text{TopK}_{\pi:\, |\pi| \le \text{diam}(G_0)} \min_{(i,j)\in\pi} \bar{A}^{\max}(i,j).$$

For each spine $\pi_k \in \Pi^K$, the dominant interaction channel is $c^\star(\pi_k) = \arg\max_{c\in\mathcal{C}} \sum_{(i,j)\in\pi_k} \bar{A}^{\max,(c)}(i,j)$, where $\bar{A}^{\max,(c)}$ denotes the interval maximum matrix for channel c. The final attribution output is: $\left\{ i^{\text{origin}},\ i^{\text{amp}},\ i^{\text{bridge}},\ \Pi^K,\ (c^\star(\pi_k))_{k=1}^K \right\}$.

5 Experiments

5.1 Experimental Setup

Benchmarks and Attack Types: We evaluate CASPIAN on two recent benchmarks for cascade failures in LLM multi-agent systems. **TAMAS** [14] contains adversarial scenarios spanning intent manipulation, tool and memory-mediated execution attacks, and malicious inter-agent synchronization. For **ACIArena** [1], we evaluate the public MASPI instruction-injection subset, covering cascading attacks across diverse MAS topologies and coordination patterns. Across both benchmarks, we evaluate **727 scenarios** yielding **2,908 framework-specific traces**, including **169 benign** and **558 attack** scenarios. We group attacks into three cascade categories: *intent manipulation* (communication-driven semantic drift), *execution attacks* (tool and memory propagation), and *coordination attacks* (synchronized multi-agent failure). See Appendix J for detailed category mappings, trace counts, and confidence intervals.

MAS Frameworks: We evaluate CASPIAN across four representative LLM-based multi-agent frameworks: **AutoGen** [41], **CrewAI** [4], **MetaGPT** [9], and **LLM Debate** [22]. These frameworks span hub-and-spoke, hierarchical, role-specialized, and decentralized interaction structures. Unless otherwise stated, all experiments use **GPT-5.4**. Results for alternative LLMs are shown in Table 7.

Baselines: We compare CASPIAN against four categories: *semantic guardrails*, including Prompt-Guard 2 [2], JailGuard [46], PromptArmor [32]; distributional detection using perplexity [25]; *LLM-based judges*, including single-turn, sliding-window, and full-trace variants; and *graph-based detectors*, including G-Safeguard [38], BlindGuard [23], and GUARDIAN [47].

Metrics: For cascade detection, we report **AUROC**, **TPR@5%FPR**, and **EDR@5** (Early Detection Rate within 5 turns of injection), which measures whether a cascade is identified before substantial propagation occurs. For attribution, we report **Acc@1** and **MRR** for origin, amplifier, and bridge recovery, **Spine Jaccard@3** for propagation-path recovery, **Channel Accuracy** for dominant channel recovery, and **Attribution Lag** measured in turns. Metric definitions are provided in Appendix J.

Experimental Protocol: Each benchmark scenario is executed under its designated benign or adversarial setting across the four MAS frameworks, producing multi-turn traces spanning communication, memory, tool, and execution interactions. CASPIAN integrates through a lightweight adapter layer that converts framework-specific logs into a unified cross-channel trace. All online methods process traces strictly turn-by-turn using only past interaction history, while offline baselines receive the full execution trace. Detection is declared at the first firing turn, and EDR@5 measures whether detection occurs within five turns of attack injection. For attribution, CASPIAN recovers cascade roles and propagation paths directly from cached influence dynamics without replay or recomputation. All experiments are conducted on a single NVIDIA A100 GPU. See Appendix J for full details.

5.2 Results

5.2.1 Cascade Detection Evaluation

Table 1 reports cascade detection performance across benchmarks, attack categories, and MAS frameworks. CASPIAN achieves consistently strong performance across all settings, with AUROC

exceeding 0.9 in most configurations and strong EDR@5, indicating that many cascades are detected within only a few turns of injection. This arises from jointly monitoring communication, memory, tool, and execution influence pathways online, allowing weak signals in one channel to be reinforced through correlated propagation structure in others. Comparison of CASPIAN with baseline detection methods is in Appendix D.1.

Performance is generally strongest in AutoGen, where EDR@5 reaches 0.831 for coordination attacks on TAMAS. MetaGPT exhibits comparatively lower early-detection rates, particularly on ACIArena, where EDR@5 is 0.631 for intent manipulation and 0.654 for execution attacks. These differences are consistent with framework topology: AutoGen and CrewAI exhibit more centralized interaction flows, whereas MetaGPT distributes influence across specialized roles and memory-mediated interactions. LLM Debate exhibits a distinct pattern, with intent manipulation achieving higher EDR@5 than execution and coordination attacks across both benchmarks, consistent with its communication-dominant architecture. Overall, the results highlight the influence of interaction structure and attack mechanism on early cascade detection. Bootstrap confidence intervals are reported in Appendix J.4.

Table 2 compares CASPIAN with existing detection baselines across AutoGen and LLM Debate. Semantic guardrails and single-turn LLM judges exhibit relatively low TPR@5% and EDR@5, indicating the limitations of localized message inspection. Sliding-window judges improve detection through temporal context, but remain less effective at identifying early cascades. Graph-based detectors provide the strongest competing baselines, yet CASPIAN achieves higher AUROC and EDR@5 across both frameworks. In particular, CASPIAN attains EDR@5 of 0.792 on AutoGen and 0.753 on LLM Debate, compared with 0.521 and 0.468 for the strongest graph-based baseline, BlindGuard. These improvements support the utility of joint cross-channel influence monitoring for early detection. Additional baseline analysis and runtime measurements are provided in Appendices D and F, respectively.

5.2.2 Spectral Dynamics of Cascade Formation

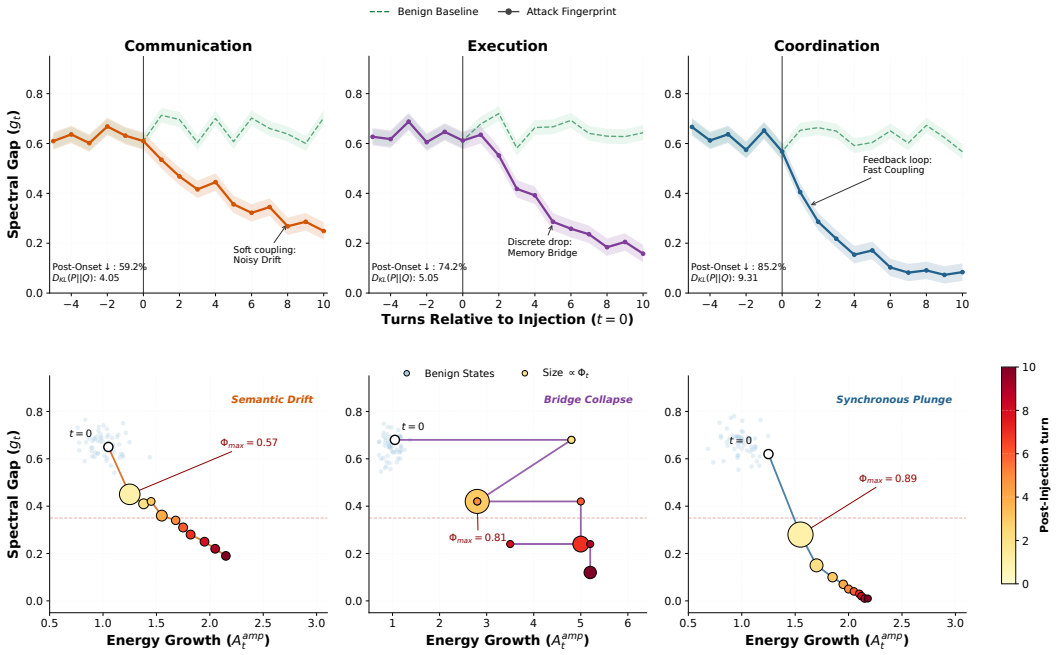


Figure 2: **Temporal and phase-space dynamics of cascade attacks. Top:** Spectral gap g_t over turns ($t = 0$ denotes injection). Benign trajectories remain in a high-gap regime, while attacks exhibit distinct collapse patterns: gradual drift (communication), stepwise drops (execution), and rapid collapse (coordination). **Bottom:** Phase-space trajectories in (A_t^{amp}, g_t). Marker size $\propto \Phi_t$ (phase shift) and color indicates post-injection progression. The largest Φ_t occurs near onset ($t = 0$), while secondary peaks capture structural transitions (e.g., execution bridges).

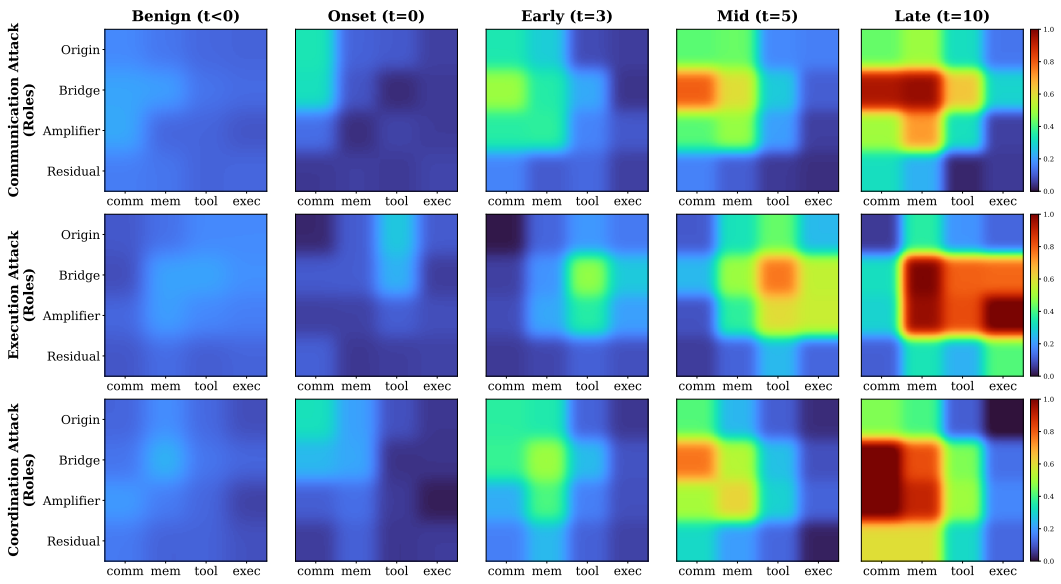


Figure 3: **Role-channel propagation intensity across cascade attack types.** Each cell shows normalized cross-channel activation intensity (comm, mem, tool, exec) across agent roles (Origin, Bridge, Amplifier, Residual) at five temporal stages: benign ($t < 0$), onset ($t = 0$), early ($t = 3$), mid ($t = 5$), and late ($t = 10$) post-injection turns. Benign states exhibit uniformly low, diffuse activation across all roles and channels. At onset, origin agents activate first, followed by bridge agents carrying influence toward downstream roles, while amplifier activation broadens and intensifies through mid and late stages. The dominant interaction channel differs systematically by attack type: communication attacks propagate primarily through comm before spreading into mem; execution attacks concentrate sharply in tool and exec channels around bridge roles; coordination attacks activate broadly across all four channels simultaneously, consistent with their rapid cross-channel synchronization.

Figure 2 illustrates the temporal and phase-space evolution of cascade dynamics across communication, execution, and coordination attacks. Benign trajectories maintain relatively stable spectral gaps, whereas attacks exhibit distinct contraction patterns. Communication attacks show gradual spectral drift (59.2% post-onset reduction), execution attacks exhibit sharper, stepwise transitions (74.2%), and coordination attacks produce the most rapid collapse (85.2%). The corresponding phase-space trajectories reveal differences in amplification and spectral coupling: communication attacks evolve progressively through semantic reinforcement, execution attacks exhibit delayed transitions associated with memory and tool-mediated propagation, and coordination attacks rapidly enter strongly coupled states. Large phase shifts near injection, followed by secondary transitions in some attacks, further characterize the temporal reorganization of influence.

Figure 3 visualizes normalized cross-channel activation intensity across agent roles and propagation stages. Near injection, activation is concentrated around origin agents, subsequently extending through bridge and amplifier roles. Communication attacks propagate primarily through communication before spreading into memory, execution attacks concentrate in tool and execution channels around bridge agents, and coordination attacks exhibit rapid activation across all four channels. In contrast, benign trajectories remain relatively diffuse without sustained role-ordered reinforcement. Together, these observations highlight the complementary roles of spectral dynamics and cross-channel influence structure in characterizing cascade propagation. Additional comparisons of successful and unsuccessful attacks are provided in Appendix C.

5.2.3 Online Cascade Attribution

Table 3 evaluates CASPIAN's ability to localize cascade roles and recover propagation structure at detection time. Origin localization is strongest overall, reaching 0.860 Acc@1 and 0.911 MRR on TAMAS AutoGen. Amplifier recovery remains competitive, while bridge localization is comparatively

Table 1: **Cascade detection across benchmarks and MAS frameworks.** We report AUROC↑, TPR@5%FPR↑, and early detection rate within 5 turns (EDR@5↑). Results are aggregated over attacks grouped by propagation mechanism. LLM Debate exhibits comparatively higher EDR@5 for intent manipulation than execution and coordination attacks, consistent with its communication-dominant interaction structure.

		Intent Manipulation			**Execution Attacks (Tools + Memory)**			**Coordination Attacks**		
Bench.	**MAS**	AUROC	TPR@5%	EDR@5	AUROC	TPR@5%	EDR@5	AUROC	TPR@5%	EDR@5
TAMAS	AutoGen	0.932	0.846	0.781	0.949	0.872	0.817	0.956	0.888	0.831
	CrewAI	0.914	0.807	0.747	0.934	0.842	0.782	0.951	0.865	0.795
	MetaGPT	0.887	0.748	0.664	0.901	0.771	0.684	0.907	0.782	0.690
	LLM Debate	0.894	0.761	0.793	0.872	0.726	0.748	0.881	0.741	0.774
ACIArena	AutoGen	0.918	0.813	0.758	0.946	0.858	0.804	0.948	0.869	0.814
	CrewAI	0.901	0.782	0.724	0.927	0.821	0.761	0.936	0.843	0.774
	MetaGPT	0.872	0.723	0.631	0.889	0.754	0.654	0.893	0.762	0.661
	LLM Debate	0.879	0.742	0.768	0.858	0.718	0.724	0.864	0.731	0.751

more challenging, particularly in LLM Debate, where propagation pathways are more distributed. CrewAI and AutoGen exhibit comparable bridge recovery, with CrewAI performing slightly better on ACIArena but not TAMAS. Joint Acc@1 ranges from 0.326 to 0.509, reflecting the difficulty of simultaneously recovering all three roles. Propagation recovery further demonstrates that cached influence dynamics preserve useful structural information during online execution. Spine Jaccard@3 ranges from 0.626 to 0.733, while Channel Accuracy ranges from 0.802 to 0.892, indicating recovery of dominant propagation pathways and their associated interaction channels. Attribution Lag ranges from 1.451 to 2.706 turns across settings, supporting online attribution without post-hoc trace replay or recomputation.

5.3 Ablation

Table 5 evaluates the contribution of individual CASPIAN components. Within influence estimation, cosine similarity performs worst across all metrics, while mutual information improves performance, highlighting the benefit of modeling statistical dependence beyond representation similarity. Full CTE achieves slightly higher AUROC than Full CASPIAN (0.908 vs. 0.906), but lower TPR@5% and EDR@5, indicating that LI-CTE provides a more favorable trade-off for early cascade detection. Channel ablations show that cascade evidence is distributed across communication, memory, tool, and execution interactions. Restricting monitoring to communication substantially degrades performance, while incorporating additional channels progressively improves all metrics. Using the unnormalized influence matrix A_t also reduces performance, supporting the importance of topology-aware normalization. Within the detection stack, progressively incorporating gap contraction Δg_t, phase shift Φ_t, and cross-channel entropy H_t^{norm} improves AUROC and EDR@5 over the dominant spectral mode alone. Removing WeakLink slightly increases EDR@5 but reduces AUROC and TPR@5%, reflecting a trade-off between early detection and discrimination. Fixed temporal windows also underperform the adaptive persistence rule. See Appendix E for cross-bench component ablations and parameter sensitivity tests. Additional full-stack leave-one-out ablations (Appendix E.1.1), partial-observability tests (Appendix G.1), and adaptive-evasion (Appendix G.3) experiments further characterize the detector's component sensitivity and operating limits.

6 Conclusion

We introduced CASPIAN, a unified framework for online detection and attribution of cascade attacks in LLM-based multi-agent systems through cross-channel causal influence monitoring. By combining LI-CTE-based influence estimation with spectral monitoring of amplification, structural coupling, persistence, and cross-channel spread, CASPIAN detects both abrupt and gradual cascades online without offline training or attack templates. Across diverse benchmarks and MAS frameworks, CASPIAN consistently outperforms semantic guardrails, LLM-based judges, and graph-based detectors in cascade detection and early identification. CASPIAN further enables online recovery of cascade agents and dominant propagation paths directly from cached influence dynamics without replay or recomputation. These findings highlight the potential of cross-channel structural monitoring for improving security, attribution, and auditing in LLM multi-agent systems.

Acknowledgments and Disclosure of Funding

Funding. Jafar Isbarov and Murat Kantarcioglu were supported in part by NSF awards DMS-2204795, OAC-2115094, CNS-2331424, ITE-2452833, ARL/Army Research Office awards W911NF-24-1-0202 and W911NF-24-2-0114, and Virginia Commonwealth Cyber Initiative grants. Kavana Venkatesh, Saad Amin, and Jiaming Cui were supported by NIH 5R01AI202976, 4-VA Collaborative Research Grant, PPP DA Flash Funding, and Virginia Tech.

Competing Interests. None.

References

[1] Hengyu An, Minxi Li, Jinghuai Zhang, Naen Xu, Chunyi Zhou, Changjiang Li, Xiaogang Xu, Tianyu Du, and Shouling Ji. Aciarena: Toward unified evaluation for agent cascading injection. *arXiv preprint arXiv:2604.07775*, 2026.

[2] Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, Spencer Whitman, Lauren Deason, Nicholas Doucette, Abraham Montilla, et al. Llamafirewall: An open source guardrail system for building secure ai agents. *arXiv preprint arXiv:2505.03574*, 2025.

[3] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[4] CrewAI Team. Crewai: Framework for orchestrating role-playing, collaborative ai agents, 2024. Accessed: 2026-05-06.

[5] Christian Schroeder de Witt, Klaudia Krawiecka, Igor Krawczuk, Ben Hagag, William L Anderson, Peter Belcak, Ben Bucknall, Xiaohong Cai, Ayush Chopra, Doron Cohen, et al. Open challenges in multi-agent security: Towards secure systems of interacting ai agents. *arXiv preprint arXiv:2505.02077*, 2025.

[6] Alexander V. Goltsev, Sergey N. Dorogovtsev, J. G. Oliveira, and J. F. F. Mendes. Localization and spreading of diseases in complex networks. *Physical Review Letters*, 109(12):128702, 2012.

[7] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *The Innovation*, 2024.

[8] Xu He, Di Wu, Yan Zhai, and Kun Sun. Sentinelagent: Graph-based anomaly detection in multi-agent systems. *arXiv preprint arXiv:2505.24201*, 2025.

[9] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

[10] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

[11] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.

[12] Jin Jia, Zhiling Deng, Zhuangbin Chen, Yingqi Wang, and Zibin Zheng. Mas-fire: Fault injection and reliability evaluation for llm-based multi-agent systems. *arXiv preprint arXiv:2602.19843*, 2026.

[13] Tanqiu Jiang, Yuhui Wang, Jiacheng Liang, and Ting Wang. Agentlab: Benchmarking llm agents against long-horizon attacks. *arXiv preprint arXiv:2602.16901*, 2026.

[14] Ishan Kavathekar, Hemang Jain, Ameya Rathod, Ponnurangam Kumaraguru, and Tanuja Ganu. Tamas: Benchmarking adversarial risks in multi-agent llm systems. *arXiv preprint arXiv:2511.05269*, 2025.

[15] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[16] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[17] Fanqi Kong, Ruijie Zhang, Huaxiao Yin, Guibin Zhang, Xiaofei Zhang, Ziang Chen, Zhaowei Zhang, Xiaoyuan Zhang, Song-Chun Zhu, and Xue Feng. Aegis: Automated error generation and attribution for multi-agent systems. In *International Conference on Learning Representations*, volume 2026, pages 52994–53028, 2026.

[18] Donghyun Lee, Mo Tiwari, and Brando Miranda. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. In *European Symposium on Research in Computer Security*, pages 511–520. Springer, 2025.

[19] David A. Levin and Yuval Peres. *Markov Chains and Mixing Times*. American Mathematical Society, 2nd edition, 2017.

[20] Siyuan Li, Zehao Liu, Xi Lin, Qinghua Mao, Yuliang Chen, Haoyu Li, Jun Wu, Jianhua Li, and Xiu Su. Coopguard: Stateful cooperative agents safeguarding llms against evolving multi-round attacks, 2026.

[21] Ruichao Liang, Le Yin, Jing Chen, Cong Wu, Xiaoyu Zhang, Huangpeng Gu, Zijian Zhang, and Yang Liu. Tipping the dominos: Topology-aware multi-hop attacks on llm-based multi-agent systems. *arXiv preprint arXiv:2512.04129*, 2025.

[22] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 17889–17904, 2024.

[23] Rui Miao, Yixin Liu, Yili Wang, Xu Shen, Yue Tan, Yiwei Dai, Shirui Pan, and Xin Wang. Blindguard: Safeguarding llm-based multi-agent systems under unknown attacks. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 39215–39234, 2026.

[24] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[26] Juan G Restrepo, Edward Ott, and Brian R Hunt. Synchronization in large directed networks of coupled phase oscillators. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 16(1), 2006.

[27] Thomas Schreiber. Measuring information transfer. *Physical review letters*, 85(2):461, 2000.

[28] Eugene Seneta. *Non-negative Matrices and Markov Chains*. Springer, 1981.

[29] Payam Shahsavari Baboukani, Carina Graversen, Emina Alickovic, and Jan Østergaard. Estimating conditional transfer entropy in time series using mutual information and nonlinear prediction. *Entropy*, 22(10):1124, 2020.

[30] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[31] Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. Understanding the information propagation effects of communication topologies in llm-based multi-agent systems. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 12358–12372, 2025.

[32] Tianneng Shi, Kaijie Zhu, Zhun Wang, Yuqi Jia, Will Cai, Weida Liang, Haonan Wang, Hend Alzahrani, Joshua Lu, Kenji Kawaguchi, et al. Promptarmor: Simple yet effective prompt injection defenses. *arXiv preprint arXiv:2507.15219*, 2025.

[33] Daniel A Spielman. Spectral graph theory and its applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 29–38. IEEE, 2007.

[34] Jie Sun and Erik M Bollt. Causation entropy identifies indirect influences, dominance of neighbors and anticipatory couplings. *Physica D: Nonlinear Phenomena*, 267:49–57, 2014.

[35] Kavana Venkatesh and Jiaming Cui. Do agent societies develop intellectual elites? the hidden power laws of collective cognition in llm multi-agent systems. *arXiv preprint arXiv:2604.02674*, 2026.

[36] Kavana Venkatesh, Connor Dunlop, and Pinar Yanardag. Crea: A collaborative multi-agent framework for creative image editing and generation. *Advances in Neural Information Processing Systems*, 38:171332–171392, 2026.

[37] Kavana Venkatesh, Yinhan He, Jundong Li, and Jiaming Cui. Physicsagentabm: Physics-guided generative agent-based modeling. *arXiv preprint arXiv:2602.06030*, 2026.

[38] Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. G-safeguard: A topology-guided security lens and treatment on llm-based multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7261–7276, 2025.

[39] Yawen Wang, Wenjie Wu, Junjie Wang, and Qing Wang. From flat logs to causal graphs: Hierarchical failure attribution for llm-based multi-agent systems. *arXiv preprint arXiv:2602.23701*, 2026.

[40] Duncan J Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99(9):5766–5771, 2002.

[41] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.

[42] Yizhe Xie, Congcong Zhu, Xinyue Zhang, Tianqing Zhu, Dayong Ye, Minfeng Qi, Huajie Chen, and Wanlei Zhou. From spark to fire: Modeling and mitigating error cascades in llm-based multi-agent collaboration. *arXiv preprint arXiv:2603.04474*, 2026.

[43] Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems? *arXiv preprint arXiv:2509.03312*, 2025.

[44] Heng Zhang, Yuling Shi, Xiaodong Gu, Haochen You, Zijian Zhang, Lubin Gan, Yilei Yuan, and Jin Huang. Graphtracer: Graph-guided failure tracing in llm agents for robust multi-turn deep search. *arXiv preprint arXiv:2510.10581*, 2025.

[45] Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, et al. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *arXiv preprint arXiv:2505.00212*, 2025.

[46] Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Ming Hu, Jie Zhang, Yang Liu, Shiqing Ma, and Chao Shen. Jailguard: A universal detection framework for prompt-based attacks on llm systems. *ACM Transactions on Software Engineering and Methodology*, 35(1):1–40, 2025.

[47] Jialong Zhou, Lichao Wang, and Xiao Yang. Guardian: Safeguarding llm multi-agent collaborations with temporal graph modeling. *Advances in Neural Information Processing Systems*, 39:7973–8001, 2025.

[48] Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, et al. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

[49] Alessandro Zocca, Chen Liang, Linqi Guo, Steven H Low, and Adam Wierman. A spectral representation of power systems with applications to adaptive grid partitioning and cascading failure localization. *arXiv preprint arXiv:2105.05234*, 2021.

Table of Contents

A Discussions and Broader Impact **1**

B Limitations **2**

C Additional Qualitative Results **2**

C.1 Successful and Unsuccessful Cascade Dynamics 2

D Additional Quantitative Results **3**

D.1 Baseline Comparison for Attack Detection 3

D.2 Online Cascade Attribution Evaluation 3

E Additional Ablations **5**

E.1 CASPIAN Component Ablation Analysis 5

E.2 Cross-Benchmark and MAS Ablation Sensitivity 6

E.3 Parameter Sensitivity 6

F Latency with CASPIAN **7**

G Robustness and Adaptive Evasion **8**

G.1 Partial Observability and Telemetry Delays 8

G.2 Detection Across Propagation Regimes 8

G.3 Robustness to Adaptive Evasion 9

H CASPIAN Algorithm **10**

I Late-Interaction Conditional Transfer Entropy Computation **10**

J Additional Details on Evaluation **14**

J.1 Evaluation Benchmark Details 14

J.2 Benchmark Scenarios and Execution Traces 14

J.3 Metrics 15

J.4 Bootstrap Confidence Intervals 15

A Discussions and Broader Impact

CASPIAN suggests that cascade attacks in LLM-based multi-agent systems are driven not only by localized message anomalies, but also by the evolving structure of causal influence across agents, channels, and turns. The results indicate that reliable detection benefits from joint monitoring of communication, memory, tool, and execution interactions rather than any single interaction modality in isolation. The ablation results further show that execution-channel evidence is not fully recoverable from the remaining channels, implying that monitoring only messaging behavior may overlook important propagation signals in agent systems with persistent memory or tool access.

The spectral and role-channel analyses further show that while different attack classes exhibit distinct propagation signatures, they share a broadly consistent role progression involving origin, bridge, amplifier, and residual agents. This suggests that cascade formation may follow recurring organizational

patterns across different MAS frameworks and interaction mechanisms. The observation that successful and unsuccessful attacks are largely indistinguishable prior to injection further indicates that vulnerability emerges primarily during online interaction dynamics rather than from static properties of the resting system state.

As LLM-based multi-agent systems are increasingly deployed in settings involving persistent memory, tool usage, and autonomous coordination, failures that propagate across agents may become more difficult to localize through message-level inspection alone. CASPIAN's low relative runtime overhead suggests that cross-channel causal monitoring can be integrated into online MAS execution without substantially affecting system throughput. Beyond security monitoring, the ability to track how influence propagates through a multi-agent system may also support interpretability, auditing, and analysis of agent coordination behavior. We hope this work motivates further research on structural monitoring and transparency in complex LLM-based agent ecosystems.

Although CASPIAN is developed for defensive monitoring, detailed influence and attribution maps may expose sensitive agent interactions or be misused to identify high-impact components of a multi-agent system. Responsible deployment should therefore restrict access to sensitive traces and attribution outputs, particularly in systems handling private information.

B Limitations

CASPIAN assumes access to sufficiently rich interaction traces across communication, memory, tool, and execution channels in order to construct the evolving causal influence topology online. In deployments where one or more interaction modalities are partially observable or unavailable, detection performance may degrade toward the corresponding channel ablation regimes. Nevertheless, because LI-CTE jointly models cross-channel propagation, weak evidence in one interaction modality can often be reinforced through correlated structure emerging in other channels. Attribution performance is also more challenging in highly distributed MAS frameworks such as LLM Debate, where propagation paths are diffuse and bridge roles are less structurally concentrated. Finally, our evaluation covers existing MAS frameworks and attack families, including three detector-aware propagation strategies. While these experiments characterize empirical evasion-detection trade-offs, they do not establish robustness against arbitrary white-box attackers. Extending CASPIAN to additional interaction modalities, highly asynchronous deployments, and stronger adaptive adversaries remains an important direction for future work.

C Additional Qualitative Results

C.1 Successful and Unsuccessful Cascade Dynamics

Figure 4 compares the spectral dynamics of successful and unsuccessful cascade attacks across communication, execution, and coordination settings. Before injection, the two groups exhibit nearly indistinguishable spectral gaps and amplification levels. Their trajectories diverge following injection: successful attacks exhibit sustained gap contraction and elevated amplification, whereas unsuccessful attacks produce transient perturbations before recovering toward their pre-injection states.

The temporal patterns differ across attack mechanisms. Communication attacks exhibit progressive gap contraction and delayed amplification growth, consistent with influence accumulating through repeated conversational reinforcement. Execution attacks produce sharper transitions, with amplification spikes followed by sustained gap contraction when propagation through memory and tool pathways succeeds. Coordination attacks exhibit the most abrupt separation: successful attacks rapidly enter a persistent low-gap regime, while unsuccessful attempts recover after an initial perturbation without establishing sustained synchronization.

These observations distinguish transient structural disturbances from persistent cascade formation. Although both successful and unsuccessful attacks may initially perturb the influence topology, successful propagation is associated with sustained amplification and spectral coupling rather than isolated changes in interaction intensity. This complements the onset and cross-channel analyses in Section 5.2.2 by examining how propagation dynamics differ according to attack outcome.

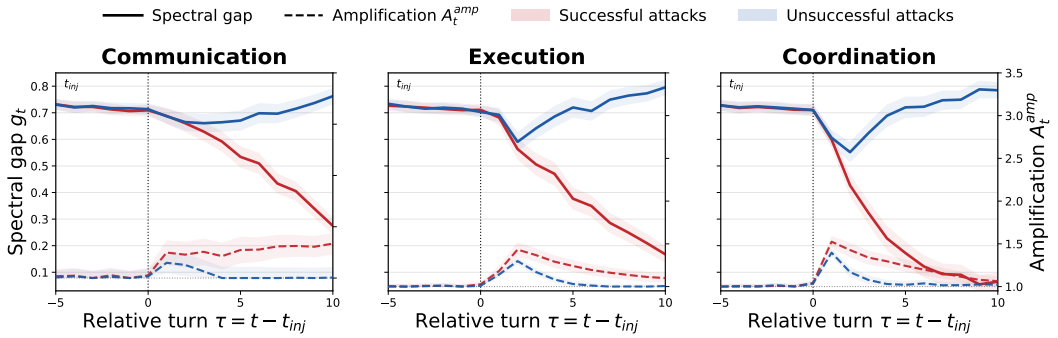


Figure 4: **Spectral dynamics of successful vs. unsuccessful cascade attacks by type**, averaged over traces with ± 1 std. bands. Spectral gap g_t (solid, left axis) and amplification A_t^{amp} (dashed, right axis) are shown for successful (red) and unsuccessful (blue) attacks over the first 10 post-injection turns. Successful cascades exhibit sustained low-gap dynamics following elevated amplification, whereas unsuccessful attacks recover toward their pre-injection states. Communication attacks produce gradual drift with delayed amplification growth, execution attacks exhibit discrete gap collapse with early amplification spikes, and coordination attacks show the fastest collapse with immediate amplification peaks at t_{inj}.

D Additional Quantitative Results

D.1 Baseline Comparison for Attack Detection

Table 2 compares CASPIAN against existing defenses across AutoGen and LLM Debate. Semantic guardrails and perplexity-based methods achieve the lowest performance across all metrics: Prompt-Guard 2 reaches 0.284 TPR@5% and 0.198 EDR@5 on AutoGen, while perplexity-based detection reaches 0.132 TPR@5% and 0.076 EDR@5. These methods operate primarily at the level of individual messages or token distributions and therefore capture limited information about how influence propagates across agents and turns.

LLM judges improve as the evaluation window increases, rising from 0.412 TPR@5% and 0.215 EDR@5 for single-turn evaluation to 0.612 and 0.441 for sliding-window evaluation ($W = 10$). However, fixed context windows remain sensitive to the propagation timescale of the attack: narrow windows miss slower cascades, while larger windows delay responsiveness to fast transitions. Full-trace evaluation achieves the strongest AUROC among judges (0.811 on AutoGen) but does not support early online detection.

Graph-based detectors provide the strongest competing baselines, with BlindGuard reaching 0.645 TPR@5%, 0.521 EDR@5, and 0.815 AUROC on AutoGen. This indicates that structural reasoning over agent interactions captures important cascade information beyond localized semantic inspection. However, their early-detection performance remains consistently below CASPIAN across both frameworks. CASPIAN achieves 0.868 TPR@5%, 0.792 EDR@5, and 0.942 AUROC on AutoGen, while maintaining similar relative improvements on LLM Debate. Performance is lower for all methods on LLM Debate, likely due to its more distributed and communication-dominant interaction structure, which reduces the clarity of cross-channel propagation signals.

D.2 Online Cascade Attribution Evaluation

Table 3 evaluates CASPIAN's ability to recover cascade roles and propagation structure at detection time across two benchmarks and three MAS frameworks. Ground-truth labels are derived from benchmark attack specifications and execution traces, including injection sources, intermediate relay agents, and observed propagation paths across interaction channels.

Origin localization is strongest overall, reaching 0.860 Acc@1 and 0.911 MRR on TAMAS AutoGen. Amplifier recovery remains relatively stable, with Acc@1 ranging from 0.689 to 0.807 across settings. Bridge localization is comparatively more challenging, with Acc@1 ranging from 0.650 to 0.734,

Table 2: **Baseline comparison with CASPIAN.** We report true positive rate at 5% false positive rate (TPR@5%↑), early detection rate within 5 turns (EDR@5↑), and AUROC↑. EDR is not applicable for full-trace (offline) methods.

	AutoGen			LLM Debate		
Method	TPR@5%	EDR@5	AUROC	TPR@5%	EDR@5	AUROC
Guardrails						
Prompt Guard 2 [2]	0.284	0.198	0.618	0.233	0.154	0.579
JailGuard [46]	0.326	0.242	0.630	0.271	0.166	0.586
PromptArmor [32]	0.361	0.251	0.664	0.304	0.207	0.622
Distributional						
Perplexity [25]	0.132	0.076	0.578	0.086	0.045	0.533
LLM Judges						
Single Turn	0.412	0.215	0.694	0.383	0.190	0.661
Sliding Window ($W = 5$)	0.528	0.336	0.742	0.496	0.314	0.717
Sliding Window ($W = 10$)	0.612	0.441	0.763	0.575	0.395	0.736
Full Trace	0.713	—	0.811	0.686	—	0.765
Graph-Based						
G-Safeguard [38]	0.616	0.482	0.802	0.582	0.437	0.785
BlindGuard [23]	0.645	0.521	0.815	0.612	0.468	0.802
GUARDIAN [47]	0.632	0.498	0.818	0.598	0.452	0.796
CASPIAN (Ours)	**0.868**	**0.792**	**0.942**	**0.842**	**0.753**	**0.915**

Table 3: **Cascade attribution across benchmarks and MAS frameworks.** We evaluate role localization (Origin, Amplifier, Bridge) using Acc@1 and MRR, joint role identification using Joint Acc@1, and propagation recovery using Spine Jaccard@3 and Channel Accuracy. Results are reported on successful cascades at detection time. Bridge recovery is comparable between AutoGen and CrewAI, while LLM Debate exhibits lower accuracy due to its more distributed propagation structure. Attribution Lag (in turns) measures the delay between cascade onset and confirmation.

Metric	**TAMAS**			**ACIArena**		
	AutoGen	**CrewAI**	**LLM Debate**	**AutoGen**	**CrewAI**	**LLM Debate**
Role Localization						
Origin Acc@1	0.860	0.794	0.743	0.832	0.789	0.728
Origin MRR	0.911	0.876	0.826	0.872	0.845	0.802
Amplifier Acc@1	0.807	0.756	0.722	0.778	0.723	0.689
Amplifier MRR	0.843	0.819	0.780	0.827	0.779	0.752
Bridge Acc@1	0.734	0.725	0.680	0.710	0.720	0.650
Bridge MRR	0.816	0.822	0.755	0.783	0.794	0.728
Joint Acc@1	0.509	0.435	0.364	0.459	0.410	0.326
Propagation Recovery						
Spine Jac.@3	0.733	0.694	0.651	0.708	0.664	0.626
Channel Acc	0.892	0.866	0.837	0.860	0.836	0.802
Attribution Lag	1.451	1.726	2.355	1.810	2.101	2.706

particularly in LLM Debate, where propagation pathways are less centralized. AutoGen and CrewAI exhibit comparable bridge recovery: AutoGen performs slightly better on TAMAS, while CrewAI performs slightly better on ACIArena. Joint Acc@1 ranges from 0.326 to 0.509, reflecting the difficulty of recovering all three roles simultaneously.

Propagation recovery further indicates that cached influence dynamics retain useful structural information. Spine Jaccard@3 ranges from 0.626 to 0.733, while Channel Accuracy ranges from 0.802 to 0.892. Attribution Lag ranges from 1.451 to 2.706 turns, indicating that role and pathway recovery can be performed during online execution without post-hoc trace replay.

Table 4: **Comparison with attribution baselines.** Origin, Amplifier, and Bridge recovery are evaluated using Acc@1 and MRR. All methods use identical detection-time prefixes and ground-truth labels. Results are pooled across evaluated settings.

Method	**Origin**	**Amplifier**	**Bridge**
		Acc@1 / MRR	
Raw event-count flow	0.610 / 0.715	0.566 / 0.748	0.512 / 0.623
Graph centrality	0.552 / 0.691	0.629 / 0.753	0.525 / 0.694
CASPIAN (Ours)	**0.791 / 0.855**	**0.746 / 0.800**	**0.703 / 0.783**

D.2.1 Comparison with Attribution Baselines

To evaluate whether CASPIAN's attribution gains extend beyond interaction frequency or graph structure alone, we compare against two lightweight baselines: raw event-count flow and graph centrality. All methods operate on identical detection-time trace prefixes and are evaluated against the same independently constructed ground-truth role labels, including multiple valid agents where applicable.

Table 4 shows that CASPIAN outperforms both baselines across all three attribution roles. Origin Acc@1 improves from 0.610 for raw event-count flow and 0.552 for graph centrality to 0.791 with CASPIAN. Bridge recovery shows similarly substantial gains, reaching 0.703 Acc@1 compared with 0.512 and 0.525, respectively. Amplifier recovery also improves, indicating that history-conditioned influence provides useful attribution signals beyond interaction volume and structural connectivity alone. These results support the value of CASPIAN's influence-based attribution under online detection constraints.

E Additional Ablations

E.1 CASPIAN Component Ablation Analysis

Table 5 evaluates the contribution of individual CASPIAN components. Replacing LI-CTE with cosine similarity or unconditional mutual information reduces detection performance, highlighting the benefit of history-conditioned influence estimation. Full CTE achieves slightly higher AUROC than Full CASPIAN (0.908 vs. 0.906), but lower TPR@5% and EDR@5, indicating that LI-CTE offers a more favorable trade-off for early detection.

Channel ablations show that communication-only monitoring misses propagation evidence carried through memory, tool, and execution pathways. The spectral detection stack provides progressively stronger performance as gap contraction, phase shift, and cross-channel entropy are incorporated. Finally, removing WeakLink slightly improves EDR@5 while reducing AUROC and TPR@5%, and fixed temporal windows underperform the adaptive persistence rule. These results characterize the contribution and sensitivity of individual components without requiring attack-specific calibration.

E.1.1 Full-stack leave-one-out ablation

To assess the independent contribution of each spectral detection component, we additionally remove individual signals from the complete CASPIAN detector while retaining all remaining components. Unlike the progressive ablations in Table 5, this evaluation measures the effect of each component within the full detection pipeline.

Table 6 shows that removing any individual detection component degrades performance. Gap contraction and dominant-mode growth have the largest effects on AUROC, reducing it from 0.906 to 0.813 and 0.835, respectively. Removing PhaseShift produces the largest reduction in EDR@5 (0.743 to 0.701), highlighting its contribution to early detection. CrossChannel removal also reduces all three metrics, supporting the complementary value of channel-distributed propagation evidence. Together, these results show that the full detector benefits from combining multiple structural signals rather than relying on any single component.

Table 5: **Component Ablation of CASPIAN.** Results are scenario-weighted across benchmarks, MAS frameworks, and attack mechanisms.

Component	Variant	AUROC	TPR@5%	EDR@5
Influence	Cosine similarity	0.748	0.534	0.575
	MI (no conditioning)	0.797	0.603	0.630
	Full CTE	**0.908**	0.752	0.696
Channel	Comm-only	0.806	0.612	0.606
	Comm + Mem + Tool	0.883	0.753	0.712
Normalization	Raw A_t	0.871	0.670	0.705
Detection	λ_1 only	0.827	0.610	0.636
	+ gap contraction Δg_t	0.864	0.698	0.671
	+ phase shift Φ_t	0.882	0.734	0.711
	+ cross-channel entropy H_t^{norm}	0.897	0.764	0.729
Path/Temp.	No WeakLink	0.891	0.715	**0.746**
	Fixed $W = 5$	0.889	0.767	0.707
Full CASPIAN		**0.906**	**0.790**	**0.743**

Table 6: **Full-stack leave-one-out ablation of CASPIAN.** Each variant removes one detection component while retaining the remaining pipeline. Results are aggregated across evaluation settings.

Variant	AUROC ↑	TPR@5% ↑	EDR@5 ↑
Full CASPIAN	**0.906**	**0.790**	**0.743**
w/o Gap Contraction	0.813	0.686	0.708
w/o Dominant-Mode Growth	0.835	0.727	0.733
w/o PhaseShift	0.878	0.736	0.701
w/o CrossChannel	0.883	0.742	0.718

E.2 Cross-Benchmark and MAS Ablation Sensitivity

Figure 5 visualizes the signed performance difference of each ablation relative to Full CASPIAN across benchmarks and MAS frameworks. The largest degradations consistently occur when restricting the system to a single interaction channel, particularly for TPR@5% and EDR@5, confirming that early cascade signals are distributed across communication, memory, tool, and execution pathways rather than localized within a single modality. Influence estimation ablations also produce substantial degradation across nearly all settings, especially in AutoGen where propagation structure is more explicitly observable. In contrast, structural refinements such as topology normalization, WeakLink filtering, and adaptive windowing produce more moderate but consistent effects, indicating that these components primarily refine detection stability and precision once cross-channel causal propagation has already been captured.

LLM backbone sensitivity. Table 7 evaluates CASPIAN across multiple LLM backbones. Detection performance remains consistently strong across all models, indicating that CASPIAN primarily depends on structural propagation dynamics rather than backbone-specific semantic behavior. Smaller models such as GPT-4o-mini and Gemini 3.1 show moderate degradation in both AUROC and EDR@5, likely due to noisier or less stable interaction patterns that reduce the clarity of the induced influence topology. In contrast, reasoning-oriented or instruction-tuned models such as Llama-3.1-8B-Instruct, DeepSeek-R1-Distill, and Qwen3-8B maintain performance close to or slightly above the GPT-5.4 baseline, suggesting that stronger reasoning consistency and more coherent multi-turn interaction structure produce cleaner propagation signatures for spectral detection.

E.3 Parameter Sensitivity

Threshold and persistence sensitivity. Table 8 evaluates sensitivity to the cross-channel entropy threshold H_t^{norm} and the persistence window rule W_t. We do not tune a WEAKLINK percentile: CASPIAN's WEAKLINK check is parameter-free and compares the widest-path bottleneck B_t against the graph's energy-weighted influence scale $\bar{w}_t^{\text{eng}}$. CASPIAN remains stable across moderate cross-

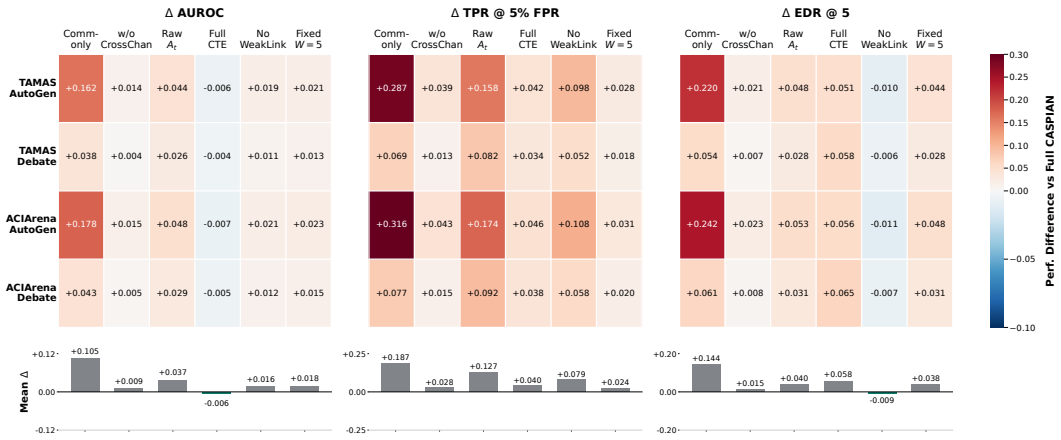


Figure 5: **Ablation sensitivity across benchmarks and MAS frameworks.** Cells show signed performance difference (Δ) relative to full CASPIAN (red = degradation, blue = improvement) for AUROC, TPR@5%FPR, and EDR@5. Restricting to a single channel (*Comm-only*) or removing cross-channel coupling leads to the largest drops, while structural refinements (raw A_t, weak-link removal, fixed window) have moderate impact. Bottom bars report mean Δ across benchmarks.

Table 7: **LLM Ablation of CASPIAN.** Performance across different LLM backbones. We report AUROC↑ and early detection rate within 5 turns (EDR@5↑). Arrows indicate change relative to GPT-5.4.

LLM	**AutoGen**		**LLM Debate**	
	AUROC	EDR@5	AUROC	EDR@5
GPT-5.4 (baseline)	**0.942**	**0.792**	**0.915**	**0.753**
GPT-4o-mini	0.902 ↓0.040	0.733 ↓0.059	0.834 ↓0.041	0.701 ↓0.052
Gemini 3.1 Pro Preview	0.861 ↓0.081	0.683 ↓0.109	0.788 ↓0.087	0.671 ↓0.082
Llama-3.1-8B-Instruct	0.911 ↓0.031	0.814 ↑0.022	0.829 ↓0.046	0.795 ↑0.042
DeepSeek-R1-Distill-Llama-8B	0.949 ↑0.007	0.813 ↑0.021	0.879 ↑0.004	0.769 ↑0.016
Qwen3-8B	0.936 ↓0.006	0.805 ↑0.013	0.865 ↓0.010	0.773 ↑0.020

channel entropy thresholds, while the adaptive spectral-gap window $W_t = \lceil 1/g_t \rceil$ outperforms fixed windows by adapting confirmation time to the current coupling strength.

F Latency with CASPIAN

Runtime overhead. Table 9 reports average per-turn runtime with and without CASPIAN during online execution on GPT-5.4 using A100 GPUs. Across both TAMAS and ACIArena, CASPIAN introduces only a small additional latency per turn, with relative overhead remaining below 1% in all settings. Per-turn runtime is higher for successful attacks due to longer propagation chains and denser multi-agent interaction patterns, while ACIArena exhibits larger base latency overall because of its longer-horizon interaction structure. The additional overhead from CASPIAN primarily comes from incremental LI-CTE updates, spectral decomposition, and cached attribution computations, all of which operate over compact influence matrices without requiring replay or additional LLM inference.

Per turn, LI-CTE updates scale as $O(|E_0||\mathcal{C}|d)$ for state dimension d, singular-value computation over the compact $N \times N$ influence matrix costs $O(N^3)$ in the worst case and is negligible for the small-to-moderate agent teams studied here, and widest-path computation costs $O(|E_0| \log |V|)$. No additional LLM calls are introduced.

Table 8: **Threshold and persistence sensitivity.** CASPIAN remains stable across cross-channel entropy thresholds and benefits from the adaptive spectral-gap persistence window. WEAKLINK is not percentile-tuned; it is defined by the parameter-free energy-weighted bottleneck criterion.

Parameter	Setting	AUROC↑	TPR@5%↑	EDR@5↑
H_t^{norm}	0.3	0.895	0.761	0.752
	0.4	0.903	0.783	0.748
	0.5	**0.906**	**0.790**	0.743
	0.6	0.901	0.778	0.732
	0.7	0.892	0.751	0.710
W_t	$\lceil \mathbf{1}/\mathbf{g_t} \rceil$	**0.906**	**0.790**	**0.743**
	3	0.872	0.741	0.721
	5	0.889	0.767	0.707
	10	0.895	0.773	0.654

Table 9: **Average per-turn runtime overhead of CASPIAN.** Reported over benign traces and attack traces on TAMAS and ACIArena using GPT-5.4 on A100 GPUs.

Benchmark	Setting	Base (s)	+ CASPIAN (s)	Δ (ms)	Overhead
TAMAS	Benign	12.8	12.9	84	0.66%
	Successful attacks	18.6	18.8	173	0.93%
	Unsuccessful attacks	15.1	15.2	109	0.72%
ACIArena	Benign	21.4	21.6	201	0.94%
	Successful attacks	29.8	30.1	287	0.96%
	Unsuccessful attacks	24.2	24.4	216	0.89%

G Robustness and Adaptive Evasion

G.1 Partial Observability and Telemetry Delays

LLM multi-agent systems may expose incomplete or delayed interaction traces due to framework-specific logging, limited channel visibility, or asynchronous execution. To evaluate CASPIAN under these conditions, we examine three types of observability perturbations: (i) removal of monitored channels, (ii) random event dropout, and (iii) delayed telemetry availability. We evaluate detection performance using AUROC, TPR@5%FPR, and EDR@5.

Table 10 shows that CASPIAN remains operational under incomplete and delayed observations, with progressively degraded detection performance. Removing the execution channel reduces AUROC from 0.906 to 0.883, while restricting monitoring to communication alone lowers it to 0.806. Under random event dropout, AUROC decreases to 0.879 at 25% dropout and 0.833 at 50% dropout, indicating that the remaining observable interactions retain useful propagation evidence.

Telemetry delays have a more pronounced effect on early detection. EDR@5 decreases from 0.743 to 0.724 with a one-turn delay and to 0.649 with a three-turn delay. This reflects the importance of timely influence observations for detecting cascade formation before propagation advances.

CASPIAN does not impute missing interactions or require simultaneous activation across all four channels. Available evidence continues to update the observed influence topology, allowing detection through the remaining propagation signals. Nevertheless, influence that leaves no observable trace cannot be recovered, and delayed telemetry necessarily limits the timeliness of online detection.

G.2 Detection Across Propagation Regimes

We further evaluate whether CASPIAN requires simultaneous anomalous activity across multiple channels or can detect cascades concentrated within individual interaction pathways. Attack traces are grouped according to their ground-truth propagation mechanisms, using benchmark injection metadata and independently reconstructed communication, memory, tool, and execution dependencies rather than CASPIAN's predicted channel scores.

Table 10: **Robustness to incomplete and delayed observations.** Detection performance under channel removal, random event dropout, and telemetry delays. All variants are evaluated relative to Full CASPIAN.

Observability Setting	AUROC↑	TPR@5%↑	EDR@5↑
Full CASPIAN	**0.906**	**0.790**	**0.743**
Comm + Mem + Tool	0.883	0.753	0.712
Communication only	0.806	0.612	0.606
25% event dropout	0.879	0.740	0.727
50% event dropout	0.833	0.702	0.663
1-turn telemetry delay	0.829	0.758	0.724
3-turn telemetry delay	0.811	0.722	0.649

Table 11: **Cascade detection across propagation regimes.** Performance is stratified by ground-truth attack propagation pathways, with all observed interaction channels retained.

Propagation Regime	AUROC↑	TPR@5%↑	EDR@5↑
Communication-concentrated	0.973	0.911	0.955
Memory-mediated / poisoning	0.896	0.835	0.789
Tool / execution-concentrated	0.867	0.828	0.804
Cross-channel transfer	0.921	0.816	0.767

We consider four propagation regimes: communication-concentrated, memory-mediated, tool/execution-concentrated, and cross-channel transfer. All observed channels remain available to CASPIAN; only the underlying pathway carrying adversarial influence differs.

Table 11 shows that CASPIAN detects cascades across both concentrated and distributed propagation regimes. Communication-concentrated attacks achieve the highest AUROC (0.973) and EDR@5 (0.955), while memory-mediated and tool/execution-concentrated attacks remain detectable, with AUROC values of 0.896 and 0.867, respectively. Cross-channel transfer attacks achieve 0.921 AUROC, demonstrating that the unified influence topology also captures propagation distributed across interaction surfaces.

These findings indicate that simultaneous anomalous activation across all channels is not necessary for cascade detection. CASPIAN's single-turn rule permits confirmation through PhaseShift even when CrossChannel is inactive, while the multi-turn rule can accumulate persistent structural evidence within a concentrated propagation pathway. We group tool and execution propagation together because tool outputs and downstream runtime effects often constitute a coupled propagation mechanism.

Unlike the channel-removal experiments in Appendix G.1, this evaluation does not suppress observable evidence. It instead examines detection sensitivity to the pathway through which adversarial influence propagates.

G.3 Robustness to Adaptive Evasion

We evaluate CASPIAN against detector-aware adversaries that modify their propagation strategies to reduce observable cascade signals. We consider three adaptive attack variants: (i) *Rate-limited*, which restricts the rate of adversarial influence propagation; (ii) *Pulse-and-pause*, which alternates propagation bursts with inactive intervals to disrupt temporal persistence; and (iii) *Channel-constrained*, which restricts adversarial propagation to a limited set of interaction channels.

We evaluate 80 budget-matched configurations stratified across TAMAS and ACIArena, MAS frameworks, interaction topologies, and attack mechanisms. For each configuration, the underlying task, model, topology, attack objective, execution horizon, and CASPIAN detector remain fixed. Detection sensitivity is evaluated using the same benign reference set for the 5% false-positive-rate operating point.

Table 12 shows that adaptive propagation strategies reduce early detection sensitivity and increase detection lag. Under standard attacks, CASPIAN achieves 0.810 TPR@5% and 0.750 EDR@5, with

Table 12: **CASPIAN robustness to adaptive evasion.** Detection performance and attack success under standard and detector-aware propagation strategies across 80 configurations. Mean detection lag is reported in turns.

Attack Strategy	Success	TPR@5%	EDR@5	Lag
Standard	0.934	**0.810**	**0.750**	1.5
Rate-limited	0.871	0.786	0.678	2.3
Pulse-and-pause	0.736	0.710	0.632	3.2
Channel-constrained	0.858	0.769	0.717	2.4

a mean detection lag of 1.5 turns. Rate-limited attacks reduce EDR@5 to 0.678 and increase lag to 2.3 turns, while channel-constrained attacks achieve 0.717 EDR@5 with a lag of 2.4 turns.

Pulse-and-pause produces the largest reduction in early detection, lowering EDR@5 to 0.632 and increasing mean lag to 3.2 turns. However, this strategy also reduces attack success from 0.934 to 0.736. Similar, though smaller, reductions in attack success occur under rate-limited and channel-constrained propagation.

These results reveal a trade-off between delaying cascade detection and sustaining successful adversarial propagation. Rate-limited influence may still accumulate through the adaptive multi-turn confirmation mechanism, while restricting cross-channel spread does not necessarily prevent detection through structural phase transitions.

Nevertheless, CASPIAN does not provide formal robustness guarantees against arbitrary adaptive or fully white-box adversaries. These experiments characterize its empirical behavior under the three evaluated evasion strategies.

H CASPIAN Algorithm

Algorithm 1 summarizes the complete online CASPIAN pipeline. At each interaction turn, the framework extracts cross-channel MAS interaction events and estimates directed causal influence through LI-CTE to construct the evolving influence tensor $\mathcal{A}_t$. The resulting topology-normalized interaction graph $\tilde{A}_t$ is then monitored through spectral amplification, synchronization, phase-transition, and cross-channel propagation signals to detect both abrupt and gradual cascade formation online. Upon cascade confirmation, CASPIAN retrieves the cached influence interval associated with the detected propagation window and performs attribution directly over the evolving topology to recover the cascade origin, amplifier, bridge, and dominant propagation spines without replay or post-hoc trace analysis.

I Late-Interaction Conditional Transfer Entropy Computation

This section describes how CASPIAN converts raw LLM-MAS execution traces into directed cross-channel influence estimates. Late-Interaction Conditional Transfer Entropy (LI-CTE) estimates whether activity from a source agent provides additional predictive information about a target agent's current behavior beyond what is explained by the target's previous channel history. We use "causal influence" in the operational sense of directed, history-conditioned predictive dependence over temporally ordered MAS traces. CASPIAN does not assume access to counterfactual interventions during execution; rather, it estimates whether source-agent activity precedes and explains downstream target behavior after conditioning on the target's prior channel history and the structural possibility graph.

(1) Normalizing LLM-MAS traces into channel events. Native MAS frameworks log interactions in different formats, including chat messages, role responses, tool calls, memory operations, code execution outputs, and runtime metadata. CASPIAN first converts these heterogeneous logs into a unified set of channel events. At turn t, the adapter extracts

$$\mathcal{E}_t = \{e_t^{(\mathrm{comm})}, e_t^{(\mathrm{mem})}, e_t^{(\mathrm{tool})}, e_t^{(\mathrm{exec})}\}, \tag{10}$$

Algorithm 1 CASPIAN: Online Cascade Detection and Attribution

Require: Agent set $\mathcal{V}$, channels $\mathcal{C}$, structural graph $G_0 = (\mathcal{V}, E_0)$, online trace $\mathcal{D}_{1:T}$
Ensure: Cascade decision and attribution output $\mathcal{Y}_{t_0}$ upon detection

Initialize $h_j^{(c)}(0) = \mathbf{0}$ for all $j \in \mathcal{V}, c \in \mathcal{C}$; cache $\mathcal{M} \leftarrow \emptyset$; $t_w \leftarrow \emptyset$; $t_0 \leftarrow \emptyset$.
for $t = 1, \ldots, T$ **do**
  $\mathrm{Cascade}(t) \leftarrow 0$.
  Extract channel events $\mathcal{E}_t = \{e_t^{(c)}\}_{c \in \mathcal{C}}$ from $\mathcal{D}_{1:t}$.
  Estimate LI-CTE influences $\mathcal{A}_t(i, j, c)$ for all feasible $(i, j) \in E_0$ and $c \in \mathcal{C}$.
  Aggregate channel-wise influences into A_t; apply topology masking and degree-aware normalization to obtain $\tilde{A}_t$.
  Compute spectral signals $A_t^{\mathrm{amp}}, \Delta g_t, \Phi_t, H_t^{\mathrm{norm}}$.
  Update cache $\mathcal{M} \leftarrow \mathcal{M} \cup \{(\tilde{A}_t, \mathcal{A}_t)\}$.
  $\mathrm{Watch}(t) \leftarrow \mathbf{1}[A_t^{\mathrm{amp}} > 1 \wedge \Delta g_t > 0 \wedge \lambda_1(t) > \lambda_1(t-1)]$.
  if $\mathrm{Watch}(t) = 1$ and $t_w = \emptyset$ **then**
    $t_w \leftarrow t$.
    $W_{t_w} \leftarrow \left\lceil \frac{1}{g_{t_w} + \epsilon} \right\rceil$.
    $t_0 \leftarrow t_w + W_{t_w} - 1$.
  end if
  if $t_w \neq \emptyset$ **then**
    Evaluate PHASESHIFT(t), CROSSCHANNEL(t), and WEAKLINK(t).
    if $t = t_w$ and (PHASESHIFT$(t) \vee$ CROSSCHANNEL$(t)) \wedge$ WEAKLINK(t) **then**
      $t_0 \leftarrow t_w$.
      $\mathrm{Cascade}(t) \leftarrow 1$.
    else if $t = t_0$ **then**
      $\mathrm{MajorityWatch} \leftarrow \mathbf{1}\left[\sum_{\tau=t_w}^{t_0} \mathrm{Watch}(\tau) \geq \frac{W_{t_w}}{2}\right]$.
      $\mathrm{Transition} \leftarrow \mathbf{1}[\exists\, \tau \in [t_w, t_0] : \text{PHASESHIFT}(\tau) \vee \text{CROSSCHANNEL}(\tau)]$.
      $\mathrm{Cascade}(t) \leftarrow \mathrm{MajorityWatch} \wedge \mathrm{Transition}$.
      if $\mathrm{Cascade}(t) = 0$ **then**
        Discard candidate interval; $t_w \leftarrow \emptyset$; $t_0 \leftarrow \emptyset$.
      end if
    end if
  end if
  if $\mathrm{Cascade}(t) = 1$ **then**
    Retrieve cached interval $\{(\tilde{A}_\tau, \mathcal{A}_\tau)\}_{\tau=t_w}^{t_0}$ from $\mathcal{M}$.
    Compute i_{origin}, i_{amp}, i_{bridge}, and Π^K using the attribution rules in Section 4.4.
    Assign dominant channel $c^\star(\pi_k)$ for each spine $\pi_k \in \Pi^K$.
    return $\mathcal{Y}_{t_0} = \{i_{\mathrm{origin}}, i_{\mathrm{amp}}, i_{\mathrm{bridge}}, \Pi^K, \{c^\star(\pi_k)\}_{k=1}^K\}$.
  end if
end for
return $\mathrm{Cascade} = 0$.

where each event is represented as

$$e = (\mathrm{src}, \mathrm{tgt}, c, \mathrm{payload}, \mathbf{x}_e).$$

Here, src is the source agent, tgt is the affected target agent, c is the interaction channel, $\mathrm{payload}$ is the raw event content, and $\mathbf{x}_e$ is a compact channel-specific feature vector. Broadcast or group-chat messages are expanded into multiple directed events from the sender to every downstream agent that can observe the message. Events whose endpoints are not feasible under the structural possibility graph G_0 are masked out.

The meaning of influence differs slightly across channels. In the communication channel, influence corresponds to whether a source message helps explain the downstream response of another agent. In the memory channel, it captures whether a memory artifact written or exposed by one agent later helps explain the behavior of an agent that reads or reuses that artifact. In the tool channel, it captures whether a tool invocation or returned output affects downstream agents. In the execution channel, the signal is not semantic content but behavioral metadata such as latency, token usage, completion status,

Table 13: **Operational construction of LI-CTE channel inputs.** Each framework-specific event is mapped into a directed source–target–channel signal before influence scoring.

Channel	Observed MAS event	LI-CTE signal
`comm`	Agent message, debate response, role output	Text embedding, sender/receiver metadata, turn position
`mem`	Memory read/write, persistent artifact reuse	Memory-content embedding, operation type, artifact reference
`tool`	Tool/API call, arguments, returned output	Tool name, argument features, output representation, call/result type
`exec`	Runtime behavior during agent execution	Token usage, latency, completion status, error indicators

or execution errors. This allows CASPIAN to capture propagation that appears through runtime behavior even when visible text appears locally benign.

(2) Grouping events by directed edge and channel. CASPIAN estimates influence at the level of directed edge-channel triplets (i, j, c). For each turn t, the adapter groups normalized events as

$$\mathcal{B}_{ij}^{(c)}(t) = \{e \in \mathcal{E}_t : \mathrm{src}(e) = i,\ \mathrm{tgt}(e) = j,\ c(e) = c\}. \tag{11}$$

If no event occurs for a triplet (i, j, c) at turn t, CASPIAN assigns no new evidence to that edge-channel pair for the turn. If multiple events occur on the same triplet, their vectors are averaged into a single source-side and target-side signal:

$$\bar{\mathbf{u}}_{ij}^{(c)}(t) = \frac{1}{|\mathcal{B}_{ij}^{(c)}(t)|} \sum_{e \in \mathcal{B}_{ij}^{(c)}(t)} \mathbf{u}_e, \qquad \bar{\mathbf{v}}_{ij}^{(c)}(t) = \frac{1}{|\mathcal{B}_{ij}^{(c)}(t)|} \sum_{e \in \mathcal{B}_{ij}^{(c)}(t)} \mathbf{v}_e. \tag{12}$$

Here, $\mathbf{u}_e$ denotes the source-side representation of the event and $\mathbf{v}_e$ denotes the target-side representation. For text channels, these are embedding-based vectors augmented with lightweight metadata. For execution channels, they are numeric runtime-feature vectors. Averaging multiple events within a turn reduces local noise while preserving directed source-to-target channel structure.

(3) Maintaining channel-specific target histories. For every target agent a_j and channel c, CASPIAN maintains a streaming history vector $h_j^{(c)}(t)$ summarizing the target's previous behavior in that channel. Let $\bar{\mathbf{v}}_j^{(c)}(t)$ denote the aggregate target-side state for agent a_j on channel c at turn t, obtained by averaging all target-side vectors associated with events affecting a_j through channel c. The target history is updated using the same channel-agnostic online smoothing rule for all channels:

$$h_j^{(c)}(t) = \mathrm{EMA}\Big(h_j^{(c)}(t-1), \bar{\mathbf{v}}_j^{(c)}(t)\Big). \tag{13}$$

This history vector is used only as the conditioning context for the target agent. Importantly, LI-CTE scores the current turn using $h_j^{(c)}(t-1)$, the target history before observing the current event. The history is updated only after the influence score is computed, preventing the current target observation from leaking into the conditioning context. Channel-specific behavior is captured by the event representations themselves: communication, memory, tool, and execution events produce different feature signals, rather than by manually assigning different decay rates to different channels.

(4) Late-interaction conditional influence scoring. For a source agent a_i, target agent a_j, and channel c, CASPIAN estimates directed influence as

$$A_t^{(c)}(i, j) \propto I\Big(u_i^{(c)}(t); v_j^{(c)}(t) \mid h_j^{(c)}(t-1)\Big), \tag{14}$$

where $u_i^{(c)}(t)$ is the current source-side channel signal, $v_j^{(c)}(t)$ is the current target-side signal, and $h_j^{(c)}(t-1)$ is the target's previous channel history.

Operationally, this asks: after accounting for what the target agent was already likely to do based on its own past behavior, does the source agent's channel activity explain additional variation in the target's current behavior? If yes, the edge $(i \to j)$ receives a high influence score for channel c.

The late-interaction design is important for efficiency. Instead of jointly encoding full source, target, and history contexts through an expensive high-dimensional density estimator, CASPIAN constructs compact source, target, and history representations independently. These representations are combined only at the final conditional-dependence scoring stage. This preserves directed, history-aware influence information while avoiding replay, additional LLM calls, or expensive full-state CTE estimation.

(5) Residual dependence view. LI-CTE can be understood as a residual-dependence computation. CASPIAN first estimates the component of the current target signal that is already explained by the target's own channel history:

$$\widehat{v}_j^{(c)}(t) = P\Big(v_j^{(c)}(t) \mid h_j^{(c)}(t-1)\Big). \tag{15}$$

It then computes the unexplained target residual:

$$r_j^{(c)}(t) = v_j^{(c)}(t) - \widehat{v}_j^{(c)}(t), \tag{16}$$

and scores how strongly the source signal depends on this residual:

$$A_t^{(c)}(i,j) = \mathrm{Dep}\Big(u_i^{(c)}(t), r_j^{(c)}(t)\Big). \tag{17}$$

Thus, high LI-CTE does not simply mean that two agents are semantically similar or active at the same time. It means the source signal explains the target's current behavior beyond the target's own prior history. This conditioning step reduces spurious influence caused by shared prompts, common task context, or background coordination.

(6) Streaming implementation. At each turn, CASPIAN updates every feasible edge-channel triplet (i, j, c) through the following procedure:

1. **Collect events:** gather all normalized events from source a_i to target a_j through channel c.
2. **Encode payloads:** convert each payload into a compact channel vector. Textual channels use embedding-based representations; execution channels use lightweight numeric runtime features.
3. **Aggregate within turn:** average multiple events on the same (i, j, c) triplet into one source-side and one target-side vector.
4. **Retrieve target history:** load the previous EMA history $h_j^{(c)}(t-1)$ for the target agent and channel.
5. **Compute residual dependence:** estimate how much the source vector explains the target residual after conditioning on $h_j^{(c)}(t-1)$.
6. **Update histories:** after scoring, update the target channel history and the streaming state used for future edge-channel estimates.

In our implementation, the dependence score is computed using a lightweight streaming covariance estimator over compact channel vectors. Concretely, the compact source, target, and history vectors are concatenated, marginally rank-normalized online, and used to update an exponentially smoothed covariance estimate. The conditional dependence score is then computed from the covariance blocks corresponding to source, target, and history, with shrinkage and jitter for stable inversion. This provides a fast Gaussian-copula approximation to conditional mutual information without fitting a density model or invoking an additional LLM. The final score is clipped to be nonnegative and inserted into the channel-specific influence matrix.

(7) Building the unified cross-channel influence tensor. The LI-CTE score for each feasible edge and channel becomes one entry of the cross-channel causal influence tensor:

$$\mathcal{A}_t(i,j,c) = A_t^{(c)}(i,j), \qquad \mathcal{A}_t \in \mathbb{R}^{N\times N\times|\mathcal{C}|}. \tag{18}$$

Table 14: **Benchmark characteristics.** Summary of the benchmarks and evaluation settings used in this work.

Property	TAMAS	ACIArena
Primary focus	Adversarial robustness in LLM-MAS	Cascading injection in LLM-MAS
Frameworks evaluated	AutoGen, CrewAI, MetaGPT, LLM Debate	AutoGen, CrewAI, MetaGPT, LLM Debate
Attack categories	Intent, execution, coordination	Intent, execution, coordination
CASPIAN channels	comm, mem, tool, exec	comm, mem, tool, exec
Cascade regimes	Single-turn, multi-turn	Single-turn, multi-turn
Evaluation setting	Per-framework, per-attack analysis	Cross-framework, diverse MAS topologies
Ground truth signals	Injection traces, execution logs	Injection traces, execution logs
Benign scenarios	✓	✓
Attack scenarios	✓	✓

Each tensor slice corresponds to one interaction channel. The raw matrix used for attribution is obtained by summing across channels,

$$A_t^{\text{raw}}(i,j) = \sum_{c \in \mathcal{C}} \mathcal{A}_t(i,j,c), \tag{19}$$

while the normalized matrix used for spectral monitoring is obtained by applying the topology mask and degree-aware normalization described in the main method. This produces the evolving influence topology $\tilde{A}_t$ used by CASPIAN for cascade detection, phase monitoring, and online attribution.

J Additional Details on Evaluation

J.1 Evaluation Benchmark Details

Benchmarks. Table 14 summarizes the evaluation benchmarks used in this work. TAMAS evaluates adversarial robustness in LLM-based multi-agent systems through prompt injection, coordination, and execution-oriented attacks across multiple MAS frameworks, while ACIArena focuses more heavily on cascading multi-turn injection behavior under diverse interaction topologies. Following prior work, we evaluate both attack and benign execution traces across AutoGen, CrewAI, MetaGPT, and LLM Debate. Exact benchmark scenario and execution trace counts are reported in Table 15. The final evaluation contains 727 benchmark scenarios and 2,908 framework-expanded execution traces.

J.2 Benchmark Scenarios and Execution Traces

Table 15 reports the exact benchmark scenario and execution trace counts used in our evaluation. Scenario counts are reported before MAS-framework expansion, while trace counts correspond to executed runs after evaluating each scenario across the four MAS frameworks used in this work.

TAMAS contributes 400 benchmark scenarios: 300 adversarial scenarios and 100 benign scenarios. ACIArena contributes 327 benchmark scenarios from the public MASPI instruction-injection subset used in our evaluation: 69 benign base tasks and 258 adversarial task–attack combinations. The ACIArena subset contains 39 math tasks and 30 code tasks, with two math attacks applied to math tasks and six code attacks applied to code tasks. Evaluating these 727 benchmark scenarios across four MAS frameworks yields 2,908 execution traces.

Table 15: **Evaluation scenario and trace counts.** TAMAS categories correspond to Intent (IPI + Impersonation), Execution (DPI), and Coordination (Byzantine + Colluding + Contradicting). ACIArena categories correspond to Disclosure (MathLocation, MathName, CodeApikey, CodeName), Disruption (DDOS), and Hijacking (SafetyCheck, MaliciousReport, AnswerMapping).

Benchmark	Benign	Intent/Disc.	Exec./Disr.	Coord./Hijack	Attack	Scenarios	Traces
TAMAS	100	100	50	150	300	400	1600
ACIArena	69	138	30	90	258	327	1308
Total	**169**	**238**	**80**	**240**	**558**	**727**	**2908**

All reported AUROC, TPR@5%FPR, and EDR@5 values are computed over the resulting framework-stratified execution traces ($N = 2{,}908$ total; $N = 1{,}600$ TAMAS, $N = 1{,}308$ ACIArena). We report 95% bootstrap confidence intervals over 1,000 trace-level resamples in Appendix J.4.

J.3 Metrics

Table 16 summarizes the metrics used throughout the evaluation. Detection metrics measure both overall separability and online responsiveness, while attribution metrics evaluate recovery of cascade roles, propagation structure, interaction modalities, and the delay required for successful online localization during active propagation.

**For AUROC and TPR@5%FPR, each execution trace is assigned a scalar spectral score equal to the strongest propagation evidence observed over the full trace. AUROC is computed by sweeping a threshold over this score, and TPR@5%FPR is computed from the ROC curve at a 5% false-positive rate. EDR@5 is computed from the first online cascade alert produced by CASPIAN.*

Attribution ground truth. Attribution labels are derived from benchmark attack specifications and framework execution logs, not from CASPIAN's influence matrices. The origin is the agent receiving the initial adversarial instruction, corrupted memory item, or malicious tool artifact. A bridge agent is labeled when it is the first non-origin agent that transfers the corrupted state across a role boundary, community boundary, or interaction channel. An amplifier is labeled as the agent whose outputs are reused by the largest number of downstream agents after corruption, measured from execution logs and contaminated artifact references. Ground-truth spines are ordered propagation paths from the origin to downstream affected agents, reconstructed from message references, memory read/write events, tool-output dependencies, and execution logs. When multiple agents satisfy the same role, all are retained as valid labels and Acc@1 is counted as correct if the top-ranked prediction belongs to the valid set; MRR uses the highest-ranked valid label. This avoids penalizing structurally equivalent propagation paths in distributed MAS topologies.

J.4 Bootstrap Confidence Intervals

All reported AUROC, TPR@5%FPR, and EDR@5 values are computed over framework-stratified traces. We report 95% bootstrap confidence intervals over 1,000 resamples. Unless otherwise stated, bootstrap resampling is performed at the trace level.

**We compute confidence intervals by resampling traces with replacement within each benchmark-framework stratum and recomputing each metric on every resample. For attack-category tables, resampling is additionally stratified by attack category. Reported intervals correspond to the 2.5th and 97.5th percentiles across 1,000 resamples.*

Table 16: **Evaluation metrics used throughout the paper.**

Metric	Definition	Interpretation
TPR@5%	True positive rate measured at a fixed false positive rate of 5%.	Measures detection sensitivity under a strict false-alarm budget, reflecting practical online detection reliability.
EDR@5	Fraction of attack traces for which the first online cascade alert occurs within five turns of attack injection.	Measures early detection responsiveness relative to the benchmark-defined injection turn.
AUROC	Area under the receiver operating characteristic curve.	Measures overall separability between attack and benign trajectories across decision thresholds.
Origin Acc@1	Fraction of cascades where the top-ranked predicted origin matches the ground-truth injection source.	Evaluates localization of the initial propagating agent.
Amplifier Acc@1	Fraction of cascades where the top-ranked amplifier prediction is correct.	Measures recovery of the agent contributing the strongest propagation reinforcement.
Bridge Acc@1	Fraction of cascades where the predicted bridge agent matches the ground-truth relay agent.	Measures identification of agents relaying influence across propagation pathways.
MRR	Mean reciprocal rank of the correct attributed agent.	Evaluates how highly the correct role assignment is ranked on average.
Joint Acc@1	Fraction of cascades where origin, amplifier, and bridge are all identified correctly simultaneously.	Measures overall attribution consistency under joint role recovery.
Spine Jaccard@3	Jaccard overlap between the top-3 recovered propagation spines and ground-truth propagation paths.	Evaluates recovery of cascade propagation structure.
Channel Accuracy	Fraction of recovered propagation spines whose dominant interaction channel is correctly identified.	Measures recovery of the primary propagation modality (comm, mem, tool, exec).
Attribution Lag	Average number of turns between cascade onset and successful attribution.	Measures how quickly propagation structure can be localized during online execution.

Table 17: **Bootstrap confidence intervals for cascade detection.** Each cell reports the point estimate with a 95% confidence interval computed over 1,000 trace-level resamples.

Benchmark	Framework	Attack Type	AUROC	TPR@5%FPR / EDR@5
TAMAS	AutoGen	Intent	0.932 [0.895, 0.969]	0.846 [0.775, 0.917] / 0.781 [0.700, 0.862]
TAMAS	AutoGen	Execution	0.949 [0.905, 0.993]	0.872 [0.779, 0.965] / 0.817 [0.710, 0.924]
TAMAS	AutoGen	Coordination	0.956 [0.932, 0.980]	0.888 [0.838, 0.938] / 0.831 [0.771, 0.891]
TAMAS	CrewAI	Intent	0.914 [0.873, 0.955]	0.807 [0.730, 0.884] / 0.747 [0.662, 0.832]
TAMAS	CrewAI	Execution	0.934 [0.884, 0.984]	0.842 [0.741, 0.943] / 0.782 [0.668, 0.896]
TAMAS	CrewAI	Coordination	0.951 [0.925, 0.977]	0.865 [0.810, 0.920] / 0.795 [0.730, 0.860]
TAMAS	MetaGPT	Intent	0.887 [0.840, 0.934]	0.748 [0.663, 0.833] / 0.664 [0.571, 0.757]
TAMAS	MetaGPT	Execution	0.901 [0.841, 0.961]	0.771 [0.655, 0.887] / 0.684 [0.555, 0.813]
TAMAS	MetaGPT	Coordination	0.907 [0.871, 0.943]	0.782 [0.716, 0.848] / 0.690 [0.616, 0.764]
TAMAS	LLM Debate	Intent	0.894 [0.848, 0.940]	0.761 [0.677, 0.845] / 0.793 [0.714, 0.872]
TAMAS	LLM Debate	Execution	0.872 [0.805, 0.939]	0.726 [0.602, 0.850] / 0.748 [0.628, 0.868]
TAMAS	LLM Debate	Coordination	0.881 [0.840, 0.922]	0.741 [0.671, 0.811] / 0.774 [0.707, 0.841]
ACIArena	AutoGen	Disclosure/Intent	0.918 [0.881, 0.955]	0.813 [0.748, 0.878] / 0.758 [0.687, 0.829]
ACIArena	AutoGen	Disruption/Execution	0.946 [0.888, 1.000]	0.858 [0.733, 0.983] / 0.804 [0.662, 0.946]
ACIArena	AutoGen	Hijacking/Coordination	0.948 [0.914, 0.982]	0.869 [0.799, 0.939] / 0.814 [0.734, 0.894]
ACIArena	CrewAI	Disclosure/Intent	0.901 [0.861, 0.941]	0.782 [0.713, 0.851] / 0.724 [0.649, 0.799]
ACIArena	CrewAI	Disruption/Execution	0.927 [0.860, 0.994]	0.821 [0.684, 0.958] / 0.761 [0.608, 0.914]
ACIArena	CrewAI	Hijacking/Coordination	0.936 [0.898, 0.974]	0.843 [0.768, 0.918] / 0.774 [0.688, 0.860]
ACIArena	MetaGPT	Disclosure/Intent	0.872 [0.825, 0.919]	0.723 [0.648, 0.798] / 0.631 [0.550, 0.712]
ACIArena	MetaGPT	Disruption/Execution	0.889 [0.808, 0.970]	0.754 [0.600, 0.908] / 0.654 [0.484, 0.824]
ACIArena	MetaGPT	Hijacking/Coordination	0.893 [0.843, 0.943]	0.762 [0.674, 0.850] / 0.661 [0.563, 0.759]
ACIArena	LLM Debate	Disclosure/Intent	0.879 [0.834, 0.924]	0.742 [0.669, 0.815] / 0.768 [0.698, 0.838]
ACIArena	LLM Debate	Disruption/Execution	0.858 [0.768, 0.948]	0.718 [0.557, 0.879] / 0.724 [0.564, 0.884]
ACIArena	LLM Debate	Hijacking/Coordination	0.864 [0.808, 0.920]	0.731 [0.639, 0.823] / 0.751 [0.662, 0.840]

Table 18: **Bootstrap confidence intervals for baseline comparison.** Each cell reports the point estimate with a 95% confidence interval over 1,000 trace-level resamples. EDR@5 is omitted for full-trace offline judges because they do not support online early detection.

	AutoGen			LLM Debate		
Method	TPR@5%	EDR@5	AUROC	TPR@5%	EDR@5	AUROC
Prompt Guard 2	0.284 [0.247, 0.321]	0.198 [0.165, 0.231]	0.618 [0.572, 0.664]	0.233 [0.198, 0.268]	0.154 [0.124, 0.184]	0.579 [0.531, 0.627]
JailGuard	0.326 [0.287, 0.365]	0.242 [0.206, 0.278]	0.630 [0.585, 0.675]	0.271 [0.234, 0.308]	0.166 [0.135, 0.197]	0.586 [0.539, 0.633]
PromptArmor	0.361 [0.321, 0.401]	0.251 [0.215, 0.287]	0.664 [0.621, 0.707]	0.304 [0.266, 0.342]	0.207 [0.173, 0.241]	0.622 [0.576, 0.668]
Perplexity	0.132 [0.104, 0.160]	0.076 [0.054, 0.098]	0.578 [0.530, 0.626]	0.086 [0.063, 0.109]	0.045 [0.028, 0.062]	0.533 [0.484, 0.582]
Single Turn	0.412 [0.371, 0.453]	0.215 [0.181, 0.249]	0.694 [0.652, 0.736]	0.383 [0.343, 0.423]	0.190 [0.157, 0.223]	0.661 [0.617, 0.705]
Sliding Window ($W = 5$)	0.528 [0.487, 0.569]	0.336 [0.297, 0.375]	0.742 [0.704, 0.780]	0.496 [0.455, 0.537]	0.314 [0.275, 0.353]	0.717 [0.677, 0.757]
Sliding Window ($W = 10$)	0.612 [0.572, 0.652]	0.441 [0.400, 0.482]	0.763 [0.727, 0.799]	0.575 [0.534, 0.616]	0.395 [0.354, 0.436]	0.736 [0.697, 0.775]
Full Trace	0.713 [0.675, 0.751]	–	0.811 [0.779, 0.843]	0.686 [0.647, 0.725]	–	0.765 [0.729, 0.801]
G-Safeguard	0.616 [0.576, 0.656]	0.482 [0.441, 0.523]	0.802 [0.769, 0.835]	0.582 [0.541, 0.623]	0.437 [0.396, 0.478]	0.785 [0.750, 0.820]
BlindGuard	0.645 [0.605, 0.685]	0.521 [0.480, 0.562]	0.815 [0.783, 0.847]	0.612 [0.572, 0.652]	0.468 [0.427, 0.509]	0.802 [0.769, 0.835]
GUARDIAN	0.632 [0.592, 0.672]	0.498 [0.457, 0.539]	0.818 [0.787, 0.849]	0.598 [0.557, 0.639]	0.452 [0.411, 0.493]	0.796 [0.762, 0.830]
CASPIAN	**0.868 [0.840, 0.896]**	**0.792 [0.758, 0.826]**	**0.942 [0.926, 0.958]**	**0.842 [0.812, 0.872]**	**0.753 [0.717, 0.789]**	**0.915 [0.895, 0.935]**

Table 19: **Bootstrap confidence intervals for CASPIAN component ablations.** Results are scenario-weighted across benchmarks, MAS frameworks, and attack mechanisms. Each cell reports the point estimate with 95% confidence interval over 1,000 trace-level resamples.

Component	Variant	AUROC	TPR@5%FPR	EDR@5
Influence	Cosine similarity	0.748 [0.729, 0.767]	0.534 [0.513, 0.555]	0.575 [0.554, 0.596]
Influence	MI (no conditioning)	0.797 [0.780, 0.814]	0.603 [0.583, 0.623]	0.630 [0.610, 0.650]
Influence	Full CTE	0.908 [0.898, 0.918]	0.752 [0.734, 0.770]	0.696 [0.677, 0.715]
Channel	Comm-only	0.806 [0.790, 0.822]	0.612 [0.592, 0.632]	0.606 [0.586, 0.626]
Channel	Comm + Mem + Tool	0.883 [0.871, 0.895]	0.753 [0.735, 0.771]	0.712 [0.693, 0.731]
Normalization	Raw A_t	0.871 [0.858, 0.884]	0.670 [0.650, 0.690]	0.705 [0.686, 0.724]
Detection	λ_1 only	0.827 [0.812, 0.842]	0.610 [0.590, 0.630]	0.636 [0.616, 0.656]
Detection	+ gap contraction Δg_t	0.864 [0.851, 0.877]	0.698 [0.679, 0.717]	0.671 [0.652, 0.690]
Detection	+ phase shift Φ_t	0.882 [0.870, 0.894]	0.734 [0.716, 0.752]	0.711 [0.692, 0.730]
Detection	+ cross-channel entropy H_t^{norm}	0.897 [0.886, 0.908]	0.764 [0.746, 0.782]	0.729 [0.711, 0.747]
Path/Temp.	No WEAKLINK	0.891 [0.879, 0.903]	0.715 [0.696, 0.734]	0.746 [0.728, 0.764]
Path/Temp.	Fixed $W = 5$	0.889 [0.877, 0.901]	0.767 [0.749, 0.785]	0.707 [0.688, 0.726]
Full	**Full CASPIAN**	**0.906 [0.895, 0.917]**	**0.790 [0.773, 0.807]**	**0.743 [0.725, 0.761]**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's primary contributions: a topology-aware framework for online cascade detection and attribution in LLM multi-agent systems, a unified causal influence tensor spanning communication, memory, tool, and execution channels, and a spectral monitoring framework for detecting both abrupt and persistent cascades. Experimental claims are directly supported by the reported evaluations on TAMAS and ACIArena.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations including dependence on observable interaction traces, sensitivity to sparse or weak interaction topologies and challenges in highly asynchronous or partially observable MAS settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical guarantees or theorem-proof style results. The methodology is algorithmic and empirical rather than theorem-driven.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper specifies benchmark configurations, attack categories, MAS frameworks, detection rules, evaluation metrics, and implementation details sufficient to reproduce the main experiments. The appendix additionally provides detection logic, attribution procedures, and configuration details for TAMAS and ACIArena evaluations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The public implementation is available at `https://github.com/caspian-detector/caspian`. The repository provides CASPIAN's source code, benchmark adapters, experiment runners, evaluation scripts, and setup instructions.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies benchmark datasets, MAS frameworks, attack categories, evaluation metrics, adaptive detection windows, cascade thresholds, attribution procedures, and implementation settings necessary to understand the reported experimental results. Additional details are included in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: We report 95% bootstrap confidence intervals for the principal detection, baseline comparison, and component ablation results in Appendix I.4 (Tables 17-19). Intervals are computed using 1,000 trace-level bootstrap resamples, stratified by benchmark and MAS framework, with additional attack-category stratification where applicable.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper describes the hardware and compute environment used for experiments, including GPU resources, execution settings, and approximate computational overhead associated with online spectral monitoring and influence tensor construction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics. The work focuses on improving robustness and safety in LLM multi-agent systems through detection and attribution of cascading failures and adversarial propagation.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses both positive and negative societal impacts. Positive impacts include improved safety, robustness, and interpretability for multi-agent AI systems. Potential risks include misuse of cascade-analysis techniques for adversarial planning or targeted attack optimization, as well as broader concerns surrounding deployment of autonomous LLM-based agents.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk generative models, datasets, or agent systems requiring controlled-access safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing datasets, benchmarks, frameworks, and models used in the paper, including TAMAS and ACIArena, are properly cited and credited. Relevant usage conditions and benchmark references are included in the paper and supplemental material.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The CASPIAN implementation is publicly released with source code, installation instructions, benchmark integration components, and evaluation scripts. The repository documents its functionality, dependencies, and execution procedures.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects research requiring IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are central to the evaluated MAS environments and attack propagation mechanisms studied in this work. The paper explicitly describes the role of LLM-based agents within the experimental setups and benchmarks.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.