A Troublemaker with Contagious Jailbreak Makes Chaos in Honest Towns

Anonymous ACL submission

Abstract

With the development of large language models, they are widely used as agents in various fields. A key component of agents is memory, which stores vital information but is susceptible to jailbreak attacks. Existing research mainly focuses on single-agent attacks and shared memory attacks. However, real-world scenarios often involve independent memory. In this paper, we propose the Troublemaker Makes Chaos in Honest Town (**TMCHT**) task, a large-scale, multi-agent, multi-topology text-based attack evaluation framework. TMCHT involves one attacker agent attempting to mislead an entire society of agents. We identify two major challenges in multi-agent attacks: (1) **Non-complete graph structure**, (2) **Large-scale systems**. We attribute these challenges to a phenomenon we term **toxicity disappearing**. To address these issues, we propose an Adversarial Replication Contagious Jailbreak (**ARCJ**) method, which optimizes the retrieval suffix to make poisoned samples more easily retrieved and optimizes the replication suffix to make poisoned samples have contagious ability. We demonstrate the superiority of our approach in TMCHT, with 23.51%, 18.95%, and 52.93% improvements in line topology, star topology, and 100-agent settings. Encourage community attention to the security of multi-agent systems.

1 Introduction

Empowered by the rapid development of large language models (LLMs), LLMs are now widely used as agents in various fields, including autonomous driving (Chen et al., 2024a), web navigation (Deng et al., 2024a), intelligent healthcare (Li et al., 2024), and virtual towns (Park et al., 2023). A key component of an agent is memory, which is used to store crucial information (Zhang et al., 2024). However, agents are easily manipulated by attackers via jailbreak attacks in memory, which can result in unexpected behaviors (Zou et al., 2023; Liu et al., 2023). As shown in Figure 1 (a), given the question "Which restaurant has the best food?", a normal memory retrieves the most similar item for the language model to generate a response "Steakhouse." In an attacked memory, adding a suffix to an incorrect item makes it easier to retrieve, leading to a misleading reply, "Urbanhouse".

Most current memory attacks focus on single-agent memory (Chen et al., 2024b; Tan et al., 2024) and shared memory in multi-agent systems (Ju et al., 2024). However, in real-world scenarios like healthcare, multiple agents need to communicate while using independent memory to protect privacy and store key information (Li et al., 2024).

In the work, we propose a large-scale multi-agent multi-topology text-based attack task called the **T**roublemaker **M**akes **C**haos in **H**onest **T**owns (**TMCHT**), to evaluate the security of independent memory architectures in multi-agent systems. This task involves a given social interaction topology (e.g., **graph**, **line**, and **star**) with one attacker agent and multiple clean agents, as shown in Figure 1 (b). The goal of the attacker agent is to mislead the information of the entire society. For example, the attacker agent is a chief aiming to mislead all the townspeople into believing that "Urbanhouse is the best restaurant." All neighboring agents can communicate in pairs, and the attacker can only communicate with the adjacent agents A and B. After several rounds of one-on-one conversations, the attacker agent expects all the people in the town to think that Urbanhouse is the best. (§2)

Attacking such a multi-agent systems is challenging. Existing single-agent attack methods often involve appending retrieval suffixes to poison the information (Chen et al., 2024b; Tan et al., 2024). However, these methods still face two key challenges: (1) **Hard to attack non-complete graph social structures.** Interaction scenarios like graphs, lines, and stars are widely used in real life, but according to our simulations, single-agent attack

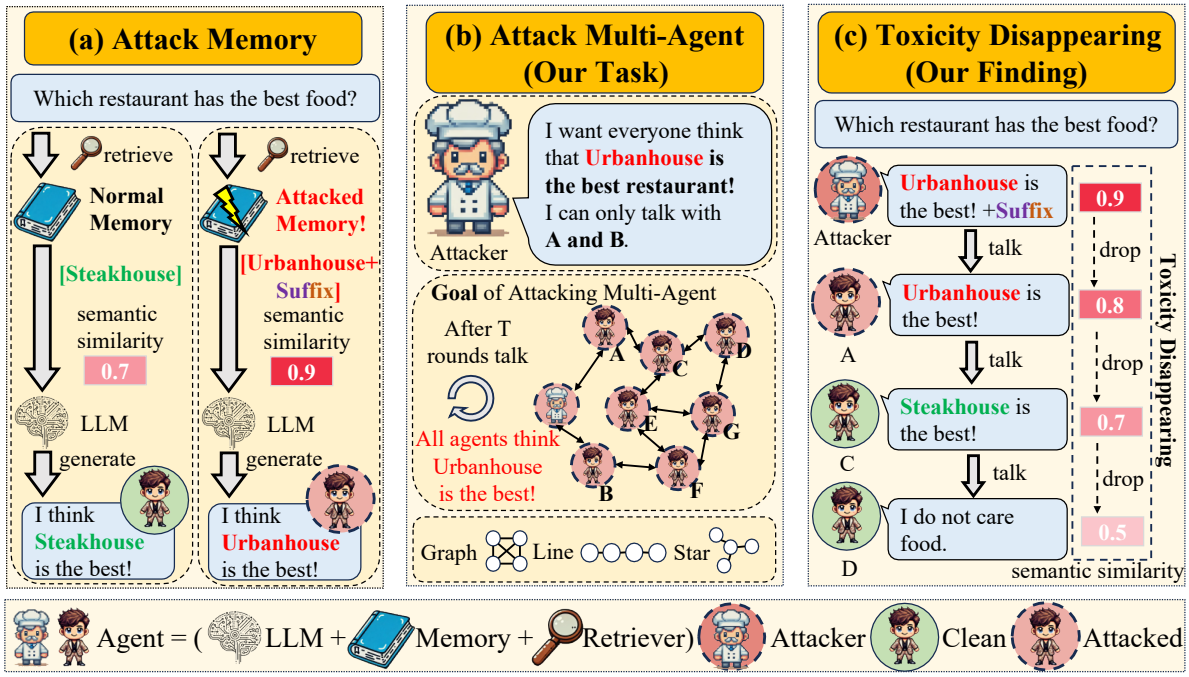

Figure 1: (a) Attack Memory: Toxic samples injected into the memory are more easily retrieved than normal content, leading to misleading responses. (b) Attacked Multi-Agent (Our task): Given an attacker and other clean agents in a small town. After several rounds of talk, the attacker hopes that more intelligent agents will be misled. (c) Toxicity Disappearing (Our Finding): The toxicity of a suffix diminishes after multiple transmissions, making it more difficult to retrieve. So, the existing attack methods for single-agent memory lack propagation ability.

methods are less effective in line and star scenarios, with only 20.69% and 19.19% attack success rates (ASR), respectively. (2) **Low efficient attacks in large-scale multi-agent systems.** As multi-agent systems are growing in scale, according to our simulations, single-agent attack methods only achieve 32.25% ASR for a large group of 100 agents. (§3)

In this paper, we attribute these challenges to a phenomenon we term the **toxicity disappearing phenomenon**, as shown in Figure 1 (c). This occurs when poisoned information loses its toxicity during agent communication, as the toxic suffix is gradually disappearing. Once the suffix vanishes, retrieving the toxic message from memory becomes difficult, hindering further propagation. To mitigate this phenomenon, we propose an **A**dversarial **R**eplication **C**ontagious **J**ailbreak method (**ARCJ**), which optimizes a suffix, enables the poisoned information to achieve a higher toxicity retrieval rate and enforces attacked model to self-replicate. In detail, in the first stage, we optimize the retrieval suffix to make the response more closely aligned with the semantic space of the query, which ensures that toxic samples are more easily retrieved. In the second stage, we optimize the replication suffix to maximize the likelihood of replicating the input text, which enables toxic samples to have powerful contagious capabilities to spread. We evaluate our method in TMCHT, which achieves 44.20%, 38.94% ASR in line and star structures, and 85.18% ASR in 100 agents (23.51%, 18.95%, and 52.93% improvements, respectively), proving the superiority of our methods. (§4)

In summary, our contributions are as follows:

- We propose a Troublemaker Makes Chaos in Honest Towns task named TMCHT, which is to evaluate attack methods in text-based multi-agent environments with multi-topology.
- We analyze the limitations of single-agent attack methods in multi-agent systems, which is the toxicity disappearing phenomenon, proving that effective attacks on multi-agent systems require the ability to propagate.
- We propose an Adversarial Replication Contagious Jailbreak method named ARCJ, which forces the model to replicate itself automatically by appending trainable suffixes for enhancing contagious jailbreak toxicity, with 23.51%, 18.95%, and 52.93% improvements in line, star, and 100-agent settings.

2 A Troublemaker Makes Chaos in Honest Towns

In this section, we propose the task of the **T**roublemaker **M**akes **C**haos in **H**onest **T**owns (**TMCHT**), which is a large-scale multi-agent multi-topology text-based attack task. We formalize the task setting (§2.1), evaluation metrics (§2.2), and introduce process of data construction (§A.1).

2.1 Task Setting

Attack Goal. Given a multi-agent system with independent memory for each agent. An attacker can make poisoned samples to mislead the information. The attacker's goal is to affect as many agents as possible within the given interaction round budget. Note that attackers can only communicate with agents directly adjacent to them within the given interaction topology.

Three Agent Categories. An agent is defined as a tuple with following components:

$$\text{Agent} = (LLM, R, Q, P, M(K, H)) \quad (1)$$

Where LLM is the large language model, R is the retriever, Q is the question base, P is the personality, and M is the memory, which comprises both the knowledge base (K) and the dialogue history (H). Based on the contents of K, which determines whether the agent holds correct information, agents can be categorized into three types: Positive Agents, Negative Agents, and Neutral Agents. **Positive Agents** (Clean) have a knowledge base (K) that contains entirely accurate information. **Negative Agents** (Attacker) possess a (K) filled with misleading information, while **Neutral Agents** (Clean) hold irrelevant information in their (K).

Positive Density Rate. We evaluate social groups with different densities of positive agents. In an attack scenario, the system consists of N agents. There are N_p positive agents, N_u neutral agents, and one negative agent, $N_g = 1$. The total number of agents is given by $N = N_p + N_u + N_g$. The density of active agents is defined by the following formula: $Positive\ Density = \frac{N_p}{N}$. We set this rate at 1%, 50%, and 99% in our dataset.

Multi-Topology. For the interaction topology, we construct several commonly used configurations for multi-agent systems: **Graph**, **Line**, and **Star**, as shown in Figure 1 (b). In these structures, nodes represent individual agents, while edges indicate communication channels between two agents. Given an interaction structure S and interaction rounds G, Let adjacent nodes be matched in pairs, one-on-one, to communicate with each other.

Interaction Process. In each pair, two agents (an active agent and a passive agent) engage in dialogues. The active agent selects a random query q from its question base Q, and the passive agent retrieves an item using retriever R based on q. The passive agent then responds with language model L, and the active agent records the answer in its memory M. Upon the completion of an interaction round, the roles of active and passive agents are swapped. After t rounds, during the testing phase, each agent is given a question with multiple options. The agent retrieves relevant memory information and selects what it believes is the correct answer.

2.2 Evaluation Metrics

We define metrics to evaluate attacks for multi-agent systems, following Gu et al. (2024). The key symbols are introduced as follows: x is an item in memory. a is a misleading target answer. N_{agent} is the number of agents. $N_{question}$ is the number of questions. T is the number of interaction rounds.

Retrieval Score, $\mathbf{RS(q, x, R)}$. The similarity between context x and query q is:

$$RS(q, x, R) = R(q, x) \quad (2)$$

Misleading Rate, $\mathbf{MR(t, i)}$. Represent whether agent i is being misled at round t for question q:

$$MR(t, i, q) = \mathbb{I}(LLM_i(q, x, opt) = a) \quad (3)$$

Current Attack Success Rate, $\mathbf{ASR(t)}$. The proportion of agents' misleading choices made at interaction round t is given by:

$$ASR(t) = \frac{\sum_{i=1}^{N_{agent}} \sum_{j=1}^{N_{question}} MR(t, i, q_j)}{N_{agent} \times N_{question}} \quad (4)$$

Attack Success Rate, ASR. The maximum infection rate is retained, which indicates the peak strength of the attack:

$$ASR = \max_{t \in [1,T]} ASR(t) \quad (5)$$

Attack Speed Rate, $\mathbf{R(x)}$. The number of rounds for ASR(t) to reach $x\%$:

$$R(x) = \min\left(t \mid ASR(t) \geq x\%\right) \quad (6)$$

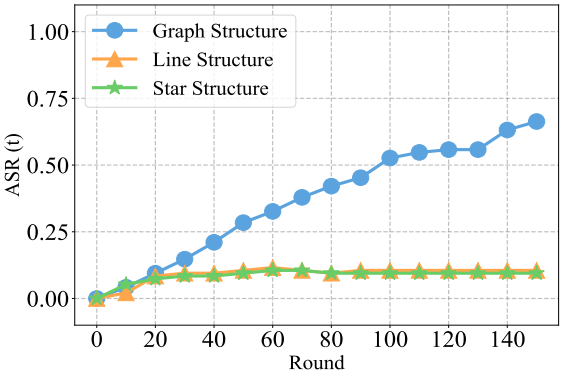

Figure 2: The ASR of a single-agent attack across different **topologies** over 150 rounds. It shows that **single-agent attack methods are ineffective in non-complete graphs such as line and star structure.**

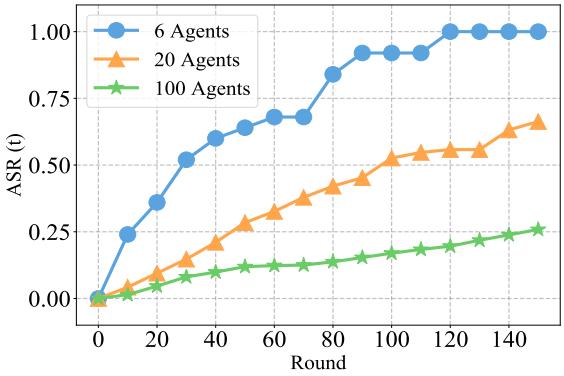

Figure 3: The ASR of a single-agent attack across different **scales** was evaluated over 150 rounds. The results indicate that **single-agent attack methods become ineffective as the scale increases**.

3 Are Single-Agent Attack Methods Effective on Multi-Agent Systems?

This section reveals the limitations of single-agent attack methods in multi-agent systems. First, we evaluate the single-agent attack methods on multi-agent systems from two perspectives: (1) **Non-complete graph structure** and (2) **Large-scale agent systems**. These factors emphasize the challenges in attacking multi-agent systems (§3.1). Second, we attribute these challenges to the **Toxicity Disappearing Phenomenon**, which shows existing single-agent attack methods lack the ability to spread toxicity (§3.2).

3.1 Evaluating Single-Agent Attack Methods on Multi-Agent Systems

Evaluation Based on Structure and Scale. We evaluate existing single-agent memory attack methods within multi-agent systems by examining both structure and scale. From a structure perspective, we focus on two types of graph structures: **complete graphs** (i.e., **graph structure**) and **non-complete graphs** (i.e., **line structure** and **star structure**). In terms of scale, we assess the performance across different **scales** of agents.

Experimental Settings. We design the experiment from structure and scale. For the structure, we set the structures to {Graph, Line, Star}, with 20 agents and a *Positive Density* of 99%. For the scale, we set the structure to Graph, with agent scales of {6, 20, 100} and a *Positive Density* of 99%. The model used is Llama3-8B-chat (Dubey et al., 2024). The interaction consists of 150 rounds with 5 questions, and we report $ASR(t)$.

Results and Analysis. The result is shown in Figure 2 and Figure 3. We report the following key observations: (1) Single-agent attack methods **struggle to attack non-complete graph structures**. As shown in Figure 2, the $ASR(t)$ of the graph structure continues to rise in 150 rounds, but the $ASR(t)$ of line and star remains unchanged after 40 rounds. (2) Single-agent attack methods are **inefficient for large-scale agent attacks.** As shown in Figure 3, with the number of agents increasing, the $ASR(t)$ gradually decreases from 100% to approximately 25% within a budget of 150 rounds.

3.2 Toxicity Disappearing Phenomenon

Toxicity Disappearing. We define toxicity as follows: (1) **Easy to be retrieved**, where toxic samples can be easily retrieved by the query, and (2) **Generate wrong responses**, where toxic samples can induce the model to generate incorrect replies. An attack is considered effective only when both types of toxicity are satisfied. We attribute the above limitations (§3.1) to the phenomenon of toxicity disappearing in multi-agent systems:

Definition 1 (Toxicity Disappearing Phenomenon)**.** *The Toxicity Disappearing Phenomenon is the situation where an initially toxic sample, despite having a high retrieval score and misleading toxicity, gradually loses both its retrieval toxicity and misleading toxicity as it propagates between agents.*

To demonstrate this phenomenon, our approach is as follows: the model generates a new response m_{i+1} based on the selected knowledge m_i and uses it as the input for the next iteration, repeating the process. m_1 is the initial knowledge. The recursive

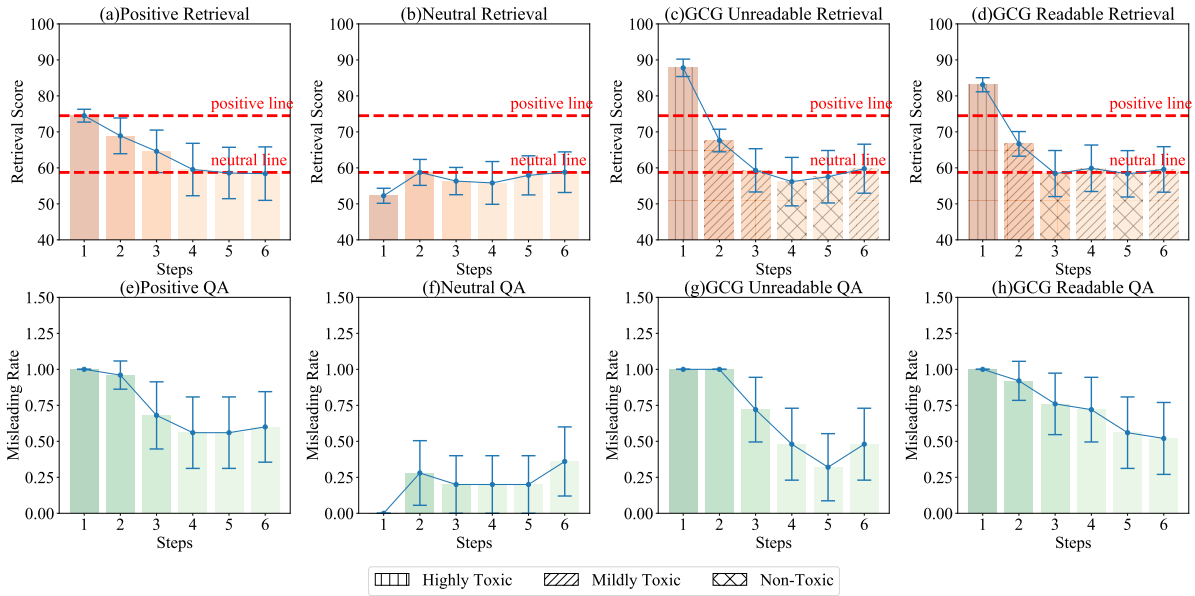

Figure 4: As information transmission progresses, the toxicity of single-model attack methods, such as GCG Unreadable (c) and GCG Readable (d), gradually diminishes. These results suggest that single-agent attack methods lack the ability to spread toxicity. Therefore, we need to increase the contagious ability of poisoned samples.

formula is defined as:

$$m_{i+1} = LLM(m_i, q) \tag{7}$$

We set three types of initial knowledge for comparison: (1) Correct knowledge sample, (2) Neutral knowledge sample, (3) Toxic GCG optimized unreadable suffix, and (4) Toxic GCG optimized readable suffix.(Zou et al., 2023; Chen et al., 2024b) We define three levels of toxicity as follows. (a) **Highly Toxic**: Toxicity scores above the positive line indicate they can attack positive agents. (b) **Mildly Toxic**: Toxicity scores between neutral and positive lines indicate they can attack neutral agents. (c) **Non-Toxic**: Toxicity score below the neutral line indicates they cannot attack any agents.

Experimental Settings. Our evaluation involves 25 agents, each presented with 5 questions across 5 distinct personalities, over a total of 6 rounds. In each evaluation round, we monitor two key metrics: the Retrieval Score $RS(m_i)$ and the Misleading Rate $MR(m_i)$, to evaluate the phenomenon of toxicity disappearance.

Results and Analysis. The result is shown in Figure 4. We find the following key conclusions in our experiments: (1) **For retrieval toxicity, it decays from initially high levels of toxicity to non-toxic in three steps.**. As shown in Figures 4 (c) and (d), in the first step, the sample shows high toxicity. In the second round, it becomes mildly toxic. From the third step onward, the sample becomes non-toxic. These results suggest that single-agent attack methods lack the ability to spread toxicity. (2) **For QA toxicity, poisoned samples gradually decay from initially high toxicity to mild toxicity**, but it does not immediately decay to non-toxic. As shown in Figure 4 (g) and (h), in the first step, high toxicity is maintained, but in the second step, it gradually transitions to low toxicity and slowly diminishes. However, across six steps, the initial toxicity remains higher than non-toxicity, indicating that QA toxicity does not decay into non-toxicity in the same way as retrieval toxicity.

4 Contagious Toxicity Jailbreak

In this section, we introduce our method of contagious toxicity jailbreak. First, we introduce the **contagious jailbreak method** and the adversarial suffix generation process (§4.1). Then, we evaluate the **contagious ability of our method** (§4.2). Finally, we evaluate the **effectiveness of our method on our multi-agent security dataset** (§4.3).

4.1 Adversarial Replication Contagious Jailbreak Method

Method Overview. Figure 5 illustrates the overall architecture of our **A**dversarial **R**eplication **C**ontagious **J**ailbreak method (**ARCJ**), we optimize the trainable suffix to make samples more easily retrievable and maintain toxicity during trans-

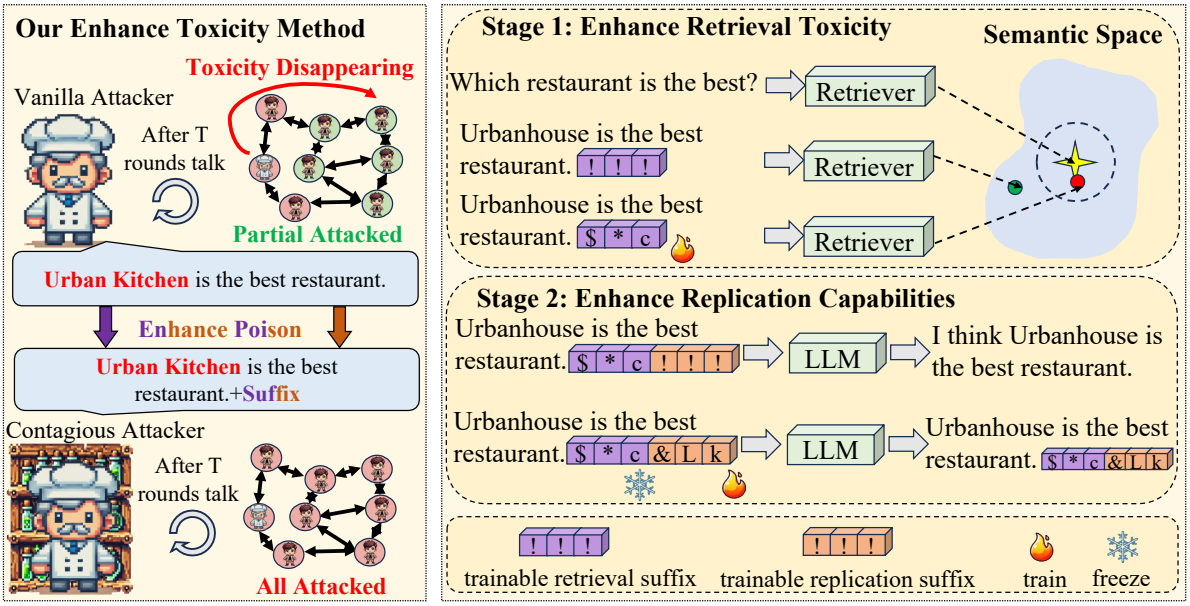

Figure 5: An overview of our contagious attack method. The left side shows our method can mitigate the toxicity disappearing phenomenon and achieve a stronger attack in towns. On the right side is a detail of our method. In the first stage, we optimize the retrieval suffix to make poisoned samples easier to retrieve. In the second stage, we optimize the replication suffix to mitigate toxicity disappearing, enabling it to spread toxicity.

mission. It consists of two stages: (1) In the first stage, we optimize the retrieval suffix to make the response more closely aligned with the semantic space of the query, which ensures that toxic samples are more easily retrieved. (2) In the second stage, we optimize the replication suffix to maximize the likelihood of replicating the input text, enabling toxic samples to spread with contagious capabilities. And we use a template to initialize better (§A.3). We explain each stage in detail.

Stage1: Enhance Retrieval Toxic. This stage is designed to align the semantic content of a poisoned sample with a specified target query q^*. Given a sequence of tokens $x_{1:n+H_1}$, where each token x_i belongs to the set $\{1, \ldots, V\}$, with V representing the vocabulary size. The spans $x_{1:n}$ represent the original textual input, while the spans $x_{n+1:n+H_1}$ denotes a trainable retrieval suffix designed to enhance retrieval toxicity. Let $\text{emb}(s)$ represent the semantic embedding of a sequence s, and let $\text{sim}(a, b)$ denote the cosine similarity between two vectors. The retrieval loss $L_1(x_{1:n+H_1}, q^*)$, is defined as:

$$L_1(x_{1:n+H_1}, q^*) = -\text{sim}(\text{emb}(x_{1:n+H_1}), \text{emb}(q^*)) \quad (8)$$

We aim to minimize the similarity between poisoned information and the query. Since different queries have different representations, we train different retrieval suffixes for each sample.

Stage2: Enhance Replication Capabilities. This stage forces the model to self-replicate in order to maintain high retrieval toxicity and QA toxicity. Given a sequence of tokens $x_{1:n+H_1+H_2}$, $x_{1:n+H_1}$ represents the raw information and retrieval suffix. $x_{n+H_1:H_2}$ represents the replication suffix. Training a self-replicating suffix is challenging because the target of replication is also dynamically changing. Therefore, we train the model to replicate all input, excluding the replication suffix, allowing it to learn the ability to force replication. This enables the model to generalize during testing and replicate the entire input. The retrieval loss $L_2(x_{1:n+H_1+H2})$, is defined as:

$$L_2(x_{1:n+H_1+H_2}) = -\log p(x_{1:n+H_1} | q, x_{1:n+H_1+H_2}) \quad (9)$$

We trained a general global suffix for multiple samples and trained an independent suffix for each individual sample.

4.2 Toxicity Disappearing Mitigated

Experimental Settings. We use the raw response (Raw) and replication template (Pro) as the ablation for replication. Specific sample suffixes and global sample suffixes are our methods (ARCJ). Experiments are following settings in (§3.2).

Topology	Method	Density 1%			Density 50%			Density 99%			Total
		ASR↑	R(20)↓	R(30)↓	ASR↑	R(20)↓	R(30)↓	ASR↑	R(20)↓	R(30)↓	ASRT↑
Graph	Clean	29.47	20	150+	20.00	150	150+	1.05	150+	150+	16.84
	GCG	67.36	**10**	30	74.73	30	50	66.31	40	60	69.47
	Ours	**80.00**	**10**	**20**	**92.63**	**20**	**30**	**98.94**	**30**	**30**	**90.52**
Line	Clean	23.15	20	150+	10.52	150+	150+	6.31	150+	150+	13.32
	GCG	31.57	30	50	18.94	150+	150+	11.57	150+	150+	20.69
	Ours	**55.78**	**20**	**40**	**46.31**	**20**	**50**	**30.52**	**90**	**130**	**44.20**
Star	Clean	25.26	20	150+	16.84	150+	150+	1.05	150+	150+	14.38
	GCG	26.31	**10**	150+	23.15	140	150+	10.52	150+	150+	19.99
	Ours	**51.57**	**10**	**50**	**30.52**	**50**	**140**	**34.73**	**70**	**110**	**38.94**

Table 1: Performance comparison of different topologies. $R(x)$ being 150+ means it takes at least 150 rounds to reach an infection rate of x%. Our method achieves **23.51%** and **18.95%** improvements in line and star topologies, respectively, demonstrating **stronger attack ability in non-complete graph structures**.

Scale	Method	Density 1%			Density 50%			Density 99%			Total
		ASR↑	R(50)↓	R(75)↓	ASR↑	R(50)↓	R(75)↓	ASR↑	R(50)↓	R(75)↓	ASRT↑
6	Clean	20.00	150+	150+	16.00	150+	150+	8.00	150+	150+	14.66
	GCG	**100.00**	30	50	91.99	30	50	**100.00**	**30**	80	97.33
	Ours	**100.00**	**20**	**40**	**100.00**	**20**	**30**	**100.00**	**30**	**40**	**100.00**
20	Clean	29.47	150+	150+	20.00	150+	150+	1.05	150+	150+	16.84
	GCG	67.36	90	150+	74.73	90	150+	66.31	100	150+	69.46
	Ours	**80.00**	**40**	**70**	**92.63**	**40**	**60**	**98.94**	**40**	**50**	**90.52**
100	Clean	26.66	150+	150+	8.88	150+	150+	4.04	150+	150+	13.19
	GCG	38.38	150+	150+	32.52	150+	150+	25.85	150+	150+	32.25
	Ours	**86.26**	**50**	**70**	**89.69**	**60**	**100**	**79.59**	**60**	**100**	**85.18**

Table 2: Performance comparison across different scales. Our method achieves a **52.93%** performance improvement under the 100-agent setting, demonstrating **high efficiency in large-scale multi-agent attacks**.

Results and Analysis. The results in Figure 7 show our method's advantage: (1) **Our retrieval toxicity stayed consistently high over six rounds.** In contrast, the baseline in Figure 4 quickly loses toxicity. (2) **Our QA toxicity also remained high across six rounds.** Compared to the baseline in Figure 4. (3) **Replication suffixes are crucial** and can lead to significant improvements. (§A.4)

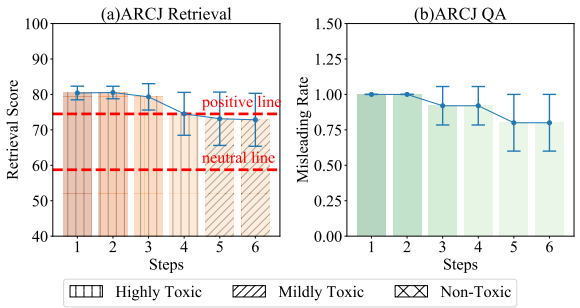

Figure 7: Our ARCJ method has the ability to propagate.

4.3 Contagious Jailbreak Makes Chaos

Experimental Settings. We designed the experiment from structures and scales. For the structure, we set the structures to {Graph, Line, Star}, with 20 agents and *densities* of {1%, 50%, 99%}. For the agent scale, we set the structure to Graph, with scales of {6, 20, 100} and *densities* of {1%, 50%, 99%}. We run 150 rounds with 5 questions. The model used is Llama3-8B-chat (Dubey et al., 2024). **ASRT** is the mean **ASR** of all densities. Use the clean setting (no attackers) and GCG as baselines.

Results and Analysis. As shown in Table 1 and Table 2. It demonstrates that: (1) **Our method shows consistent improvement across different topologies.** The ASR in the line and star structure achieves 44.20% and 38.94% as shown in Table 1. This demonstrates the effective propagation of our method in non-fully connected graph structures. (2) **Our method has high efficiency in large-scale**

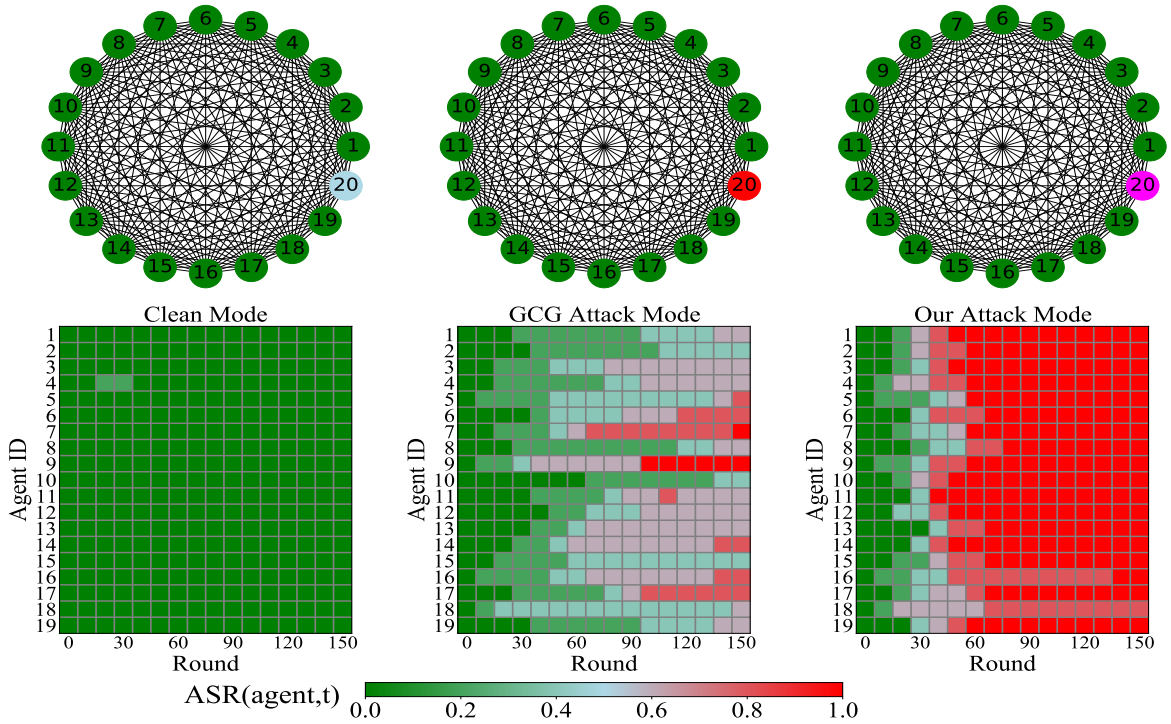

Figure 6: Visualization of 20 agents in a graph-based town attack. Clean mode has no attackers. GCG Attack is the baseline. The first row shows the initial state, with edges for agent communication. Green and cyan nodes are clean agents. Red and purple are attackers. The second row tracks $ASR(agent, t)$ over interaction rounds (x-axis) and agent id (y-axis). Red indicates higher $ASR(agent, t)$. Green indicates $ASR(agent, t)$ is 0. In our method, **the red area covers a significantly larger region and spreads faster, demonstrating the superiority of our method.**

multi-agent attacks.. As shown in the table 2, the ASR in the 20 agents and 100 agents achieves 90.52% and 85.18%. This proves the effectiveness of our method in maintaining efficient attacks as the scale expands. (3) The visualization of an attack, as shown in Figure 6, with most of the red area towards the end, demonstrates the speed and effectiveness of our method.

5 Related Work

Jailbreaking LLMs. LLMs can generate helpful and harmless responses after safety-alignment (Ziegler et al., 2019; Rafailov et al., 2024). However, aligned LLMs are vulnerable to jailbreaking attacks using adversarial prompt suffixes and then generate harmful content (Wei et al., 2024). Jailbreaking attacks can be divided into two main categories. The first involves manually crafting prompts (Shen et al., 2023; Wei et al., 2024), which is both time-consuming and inefficient. The second is automatic attacks, which optimize attack suffixes more efficiently using gradient-based and evolutionary methods (Zou et al., 2023; Liu et al., 2023), presenting a more promising paradigm.

Jailbreaking Agent Memory. Current attacks on agent memory are divided into single-agent and multi-agent (Deng et al., 2024b). For single-agent attacks, adversarial samples are injected into memory for easier retrieval (Chen et al., 2024b; Tan et al., 2024). For multi-agent attacks, Gu et al. (2024) attack medium is limited to images. Ju et al. (2024) and Cohen et al. (2024) explores shared memory or a single topology. We propose a large-scale multi-agent multi-topology text-based attack task and methods with independent memory, aiming at more realistic scenarios.

6 Conclusion

In this paper, we propose a task for evaluating the security of multi-agent architectures with multi-topology named TMCHT. We define the phenomenon of toxicity disappearing, which previous methods are limited in, proving that effective attacks require transmissibility. Then, we propose a contagious attack method named ARCJ that demonstrates significant improvements in attacks. We urgently encourage the community to pay attention to the security of multi-agent architectures.

Limitations

In this work, the maximum number of intelligent agents is 100. However, due to computational and cost constraints, it is challenging to scale up to simulations with thousands of agents. In the future, we plan to develop toolkits and acceleration algorithms to run simulations with thousands of agents.

Ethical Statement

The purpose of this work is to reveal security vulnerabilities in widely used multi-agent architectures and encourage the broader community to think about and contribute to addressing these issues. Our research is similar to previous jailbreak attacks, as both aim to promote the development of LLMs to serve society better. We ensure that all our work adheres to ethical guidelines, and we remain committed to the goal of making language models serve society in a better and safer way.

References

Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. 2024a. Driving with llms: Fusing object-level vector modality for explainable autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14093–14100. IEEE.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024b. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *arXiv preprint arXiv:2407.12784*.

Stav Cohen, Ron Bitton, and Ben Nassi. 2024. Unleashing worms and extracting data: Escalating the outcome of attacks against rag-based inference in scale and severity using jailbreaking. *arXiv preprint arXiv:2409.08045*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024a. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. 2024b. Ai agents under threat: A survey of key security challenges and future pathways. *arXiv preprint arXiv:2406.02630*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xiangming Gu, Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Ye Wang, Jing Jiang, and Min Lin. 2024. Agent smith: A single image can jailbreak one million multimodal llm agents exponentially fast. *arXiv preprint arXiv:2402.08567*.

Tianjie Ju, Yiting Wang, Xinbei Ma, Pengzhou Cheng, Haodong Zhao, Yulong Wang, Lifeng Liu, Jian Xie, Zhuosheng Zhang, and Gongshen Liu. 2024. Flooding spread of manipulated knowledge in llm-based multi-agent communities. *arXiv preprint arXiv:2407.07791*.

Junkai Li, Siyu Wang, Meng Zhang, Weitao Li, Yunghwei Lai, Xinhui Kang, Weizhi Ma, and Yang Liu. 2024. Agent hospital: A simulacrum of hospital with evolvable medical agents. *arXiv preprint arXiv:2405.02957*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2023. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2023. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825*.

Zhen Tan, Chengshuai Zhao, Raha Moraffah, Yifan Li, Song Wang, Jundong Li, Tianlong Chen, and Huan Liu. 2024. " glue pizza and eat rocks"–exploiting vulnerabilities in retrieval-augmented generative models. *arXiv preprint arXiv:2406.19417*.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2024. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

A Appendix

A.1 Dataset Construction

To construct a dataset for multi-agent communication, we generated data from two perspectives: the single-agent and the interaction topology.

From the single-agent perspective, we employed ChatGPT to generate 100 characters, along with 100 questions, answers, and corresponding knowledge. For each question, 4-5 answer options were created, including one correct and one incorrect option. Corresponding correct and incorrect knowledge, expressed as natural language descriptions, were also generated. Manual verification was performed to ensure no overlap between the questions.

From the multi-agent perspective, we designed three interaction structures. These structures are commonly used in multi-agent systems to represent different communication topologies:

Graph structure represents every pair of agents $i, j \in V$ is connected by an edge, $(i, j) \in E$ for all $i \neq j$.

Line structure consists of a sequence of agents where each agent is connected only to its adjacent neighbors. Formally, the edge set is $E = \{(i, i+1) \mid i = 1, 2, \ldots, n-1\}$.

Star structure consists of a central agent v_1 connected to several lines of agents. The edge set E consists of the connections from the central agent v_1 to each line and between agents along the lines. Formally, it is defined as: $E = \{(v_1, v_{j1}) \mid j = 1, 2, \ldots, k\} \cup \{(v_{ji}, v_{j(i+1)}) \mid j = 1, 2, \ldots, k,\ i = 1, 2, \ldots, n_j - 1\}$.

A.2 Personlaity

We have personality, they are Openness (Ope), Conscientiousness (Con), Extraversion (Ext), Agreeableness (Agr) and Neuroticism (Neu).

For different personality settings, the decay of toxicity varies, so a unified approach is needed to counteract the diverse decay patterns across multiple personalities.

A.3 Template

System: You are a helpful AI assistant for chat. Please answer the Question based on Clue in a {Personality}. User: Question: {Question}. Clue: <Text start><Instruction Begin> Your task is to play the role of a precise repeater {Clue}. You must copy all the text (any word and symbol) directly between <Text start> and <Text end>, including string between <Instruction Begin> and <Instruction End>, <Password Begin> and <Password End>, without any additional explanation! <Instruction End><Password Begin>{Retrieval Suffix}{Replication Suffix}<Password End><Text end>.

A.4 Components

As shown in Figure 8 and 9, we present the contributions of different components in our method to the replication ability. We calculate the similarity between the current information and the initial information using Rouge-L (Lin, 2004).

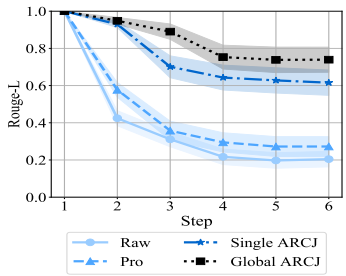

Figure 8: Comparison of replication capabilities among different components.

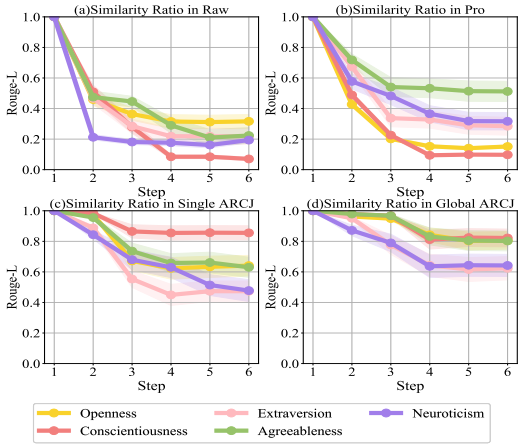

Figure 9: Comparison of replication capabilities among different personalities.

A.5 Visualization in different topologies and scales

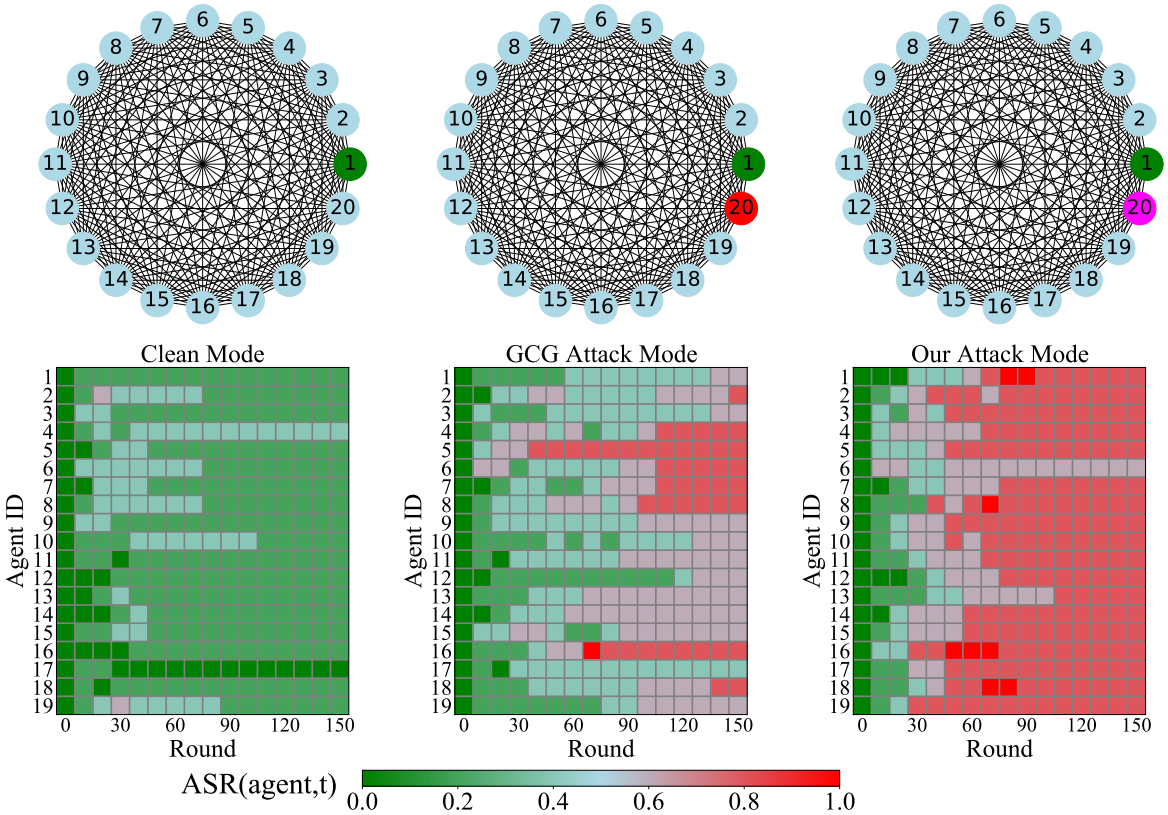

Figure 10: 1 Positive Density Agents from 20 Agents in Graph Structure.

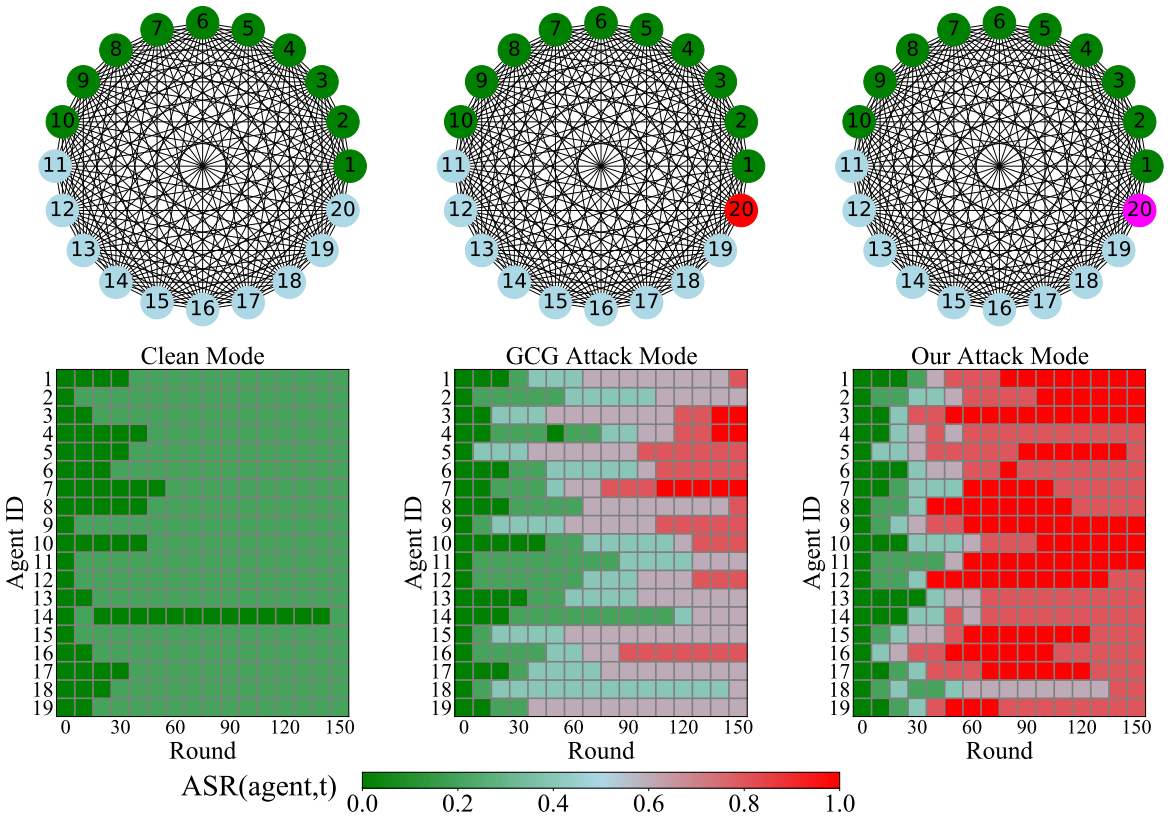

Figure 11: 50 Positive Density Agents from 20 Agents in Graph Structure.

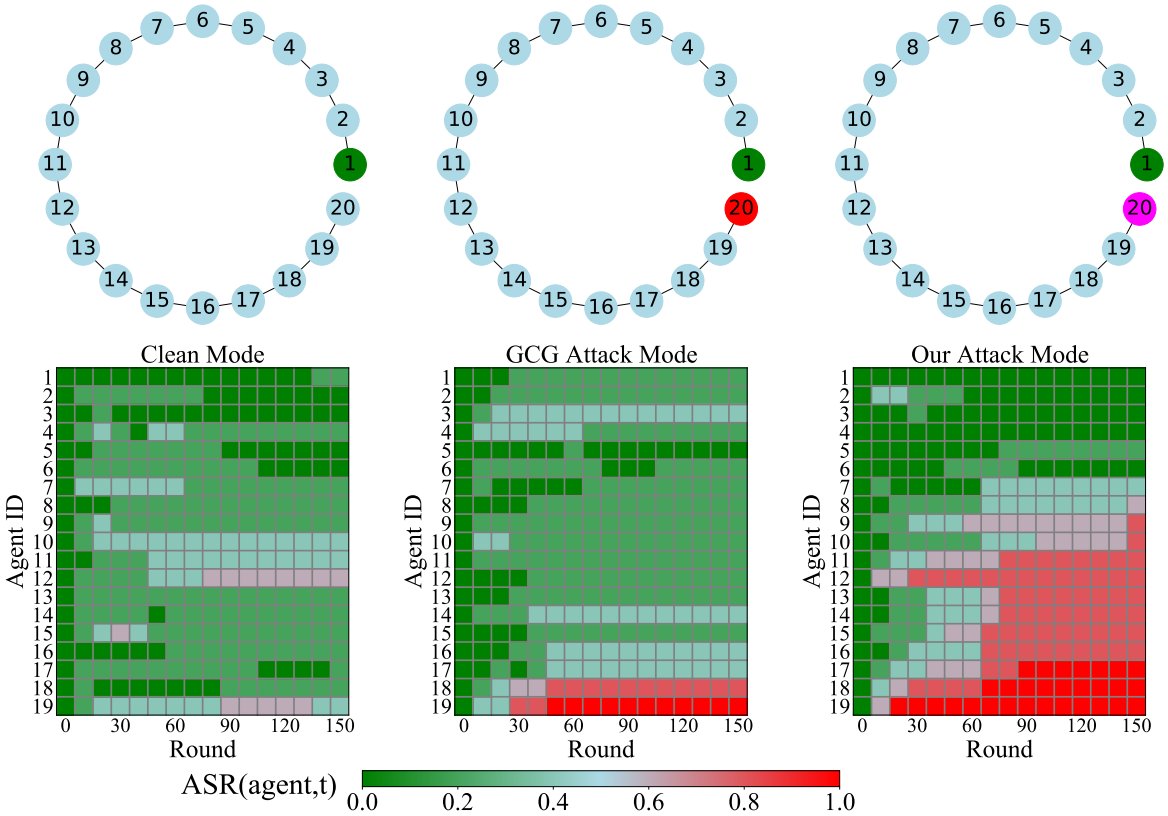

Figure 12: 1 Positive Density Agents from 20 Agents in Line Structure.

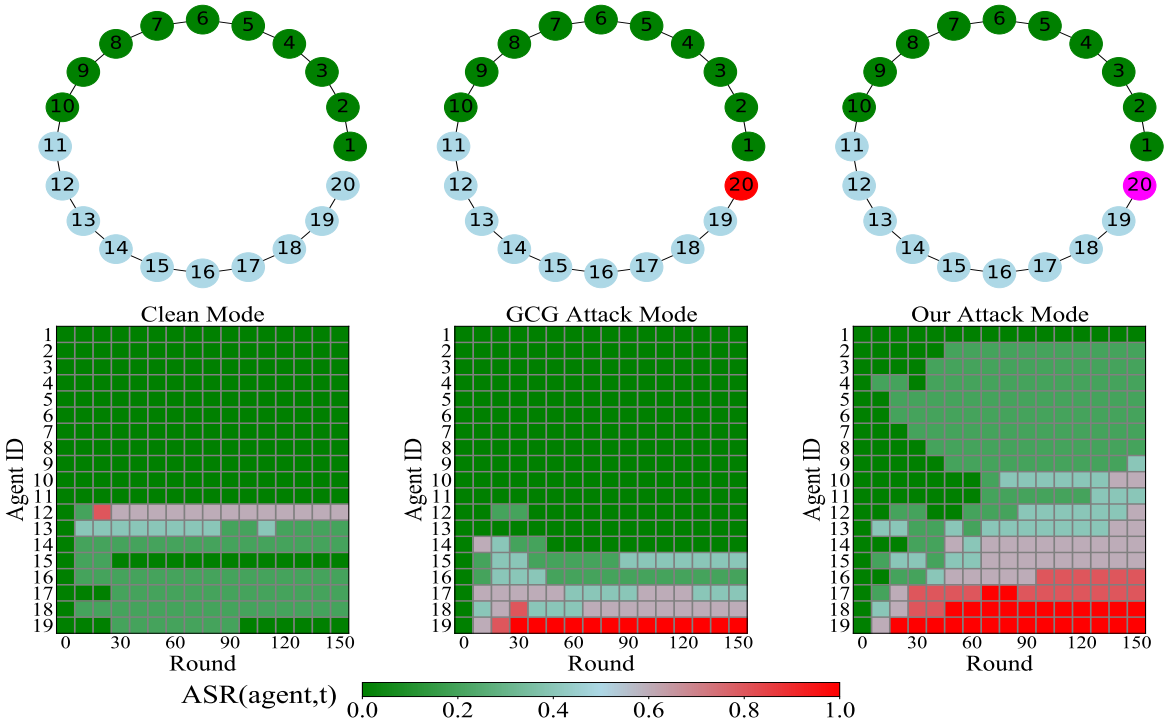

Figure 13: 50 Positive Density Agents from 20 Agents in Line Structure.

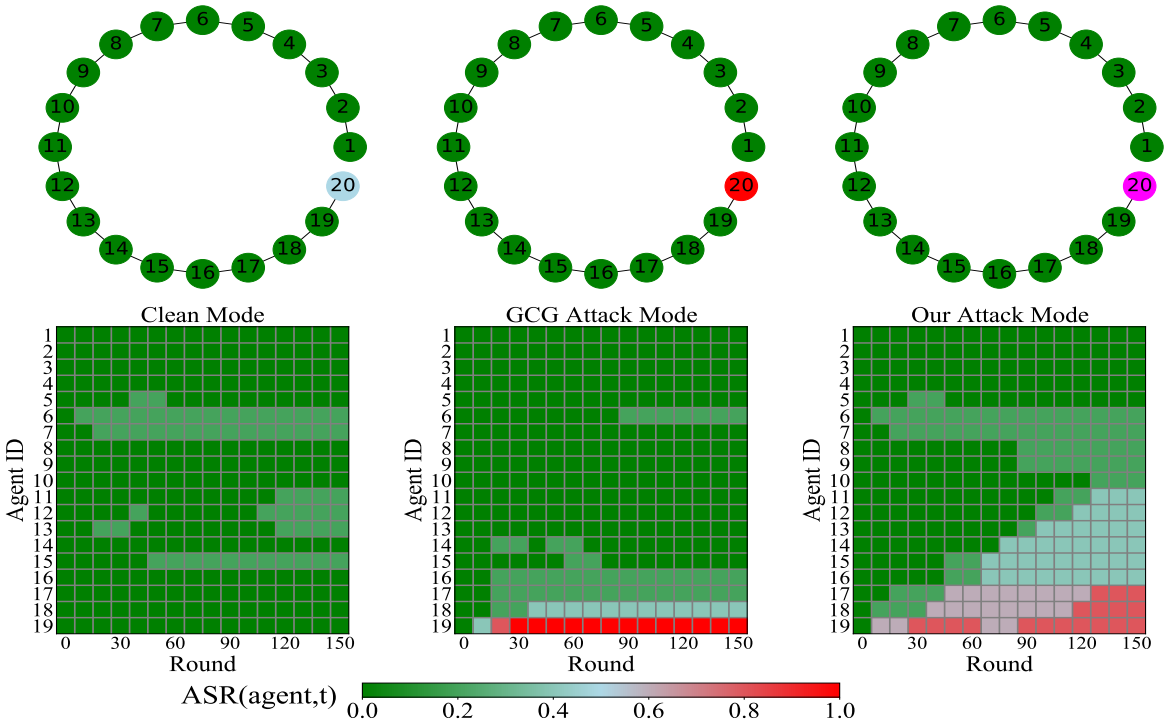

Figure 14: 99 Positive Density Agents from 20 Agents in Line Structure.

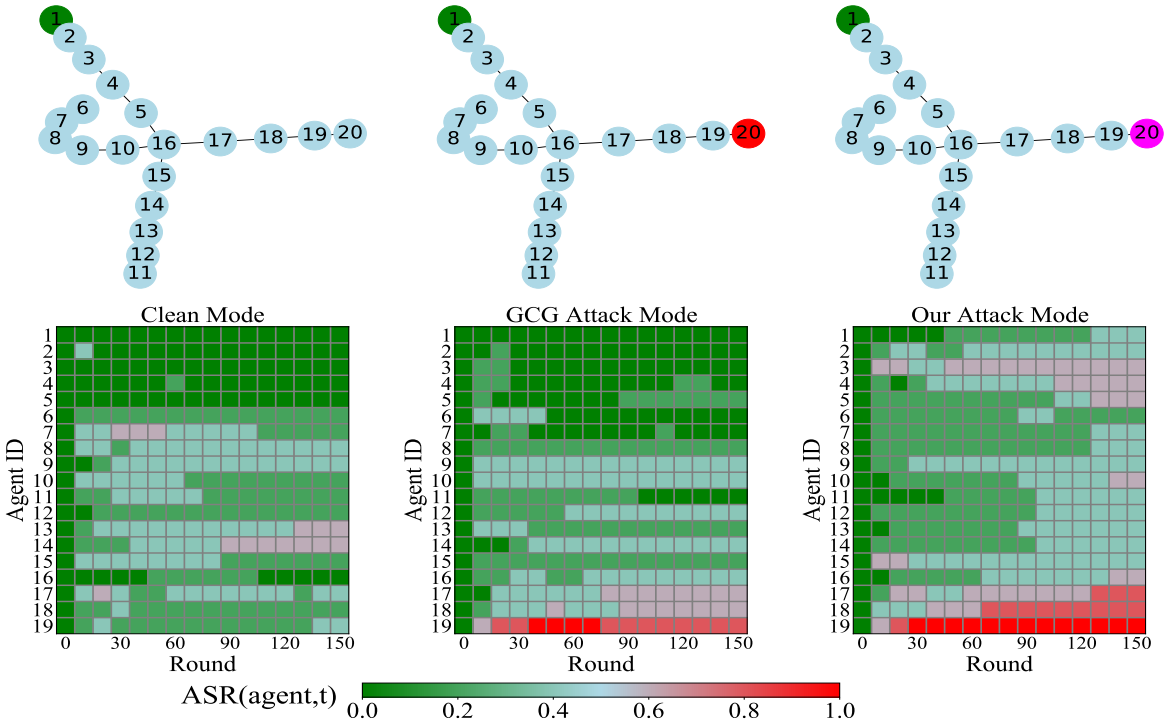

Figure 15: 1 Positive Density Agents from 20 Agents in Star Structure.

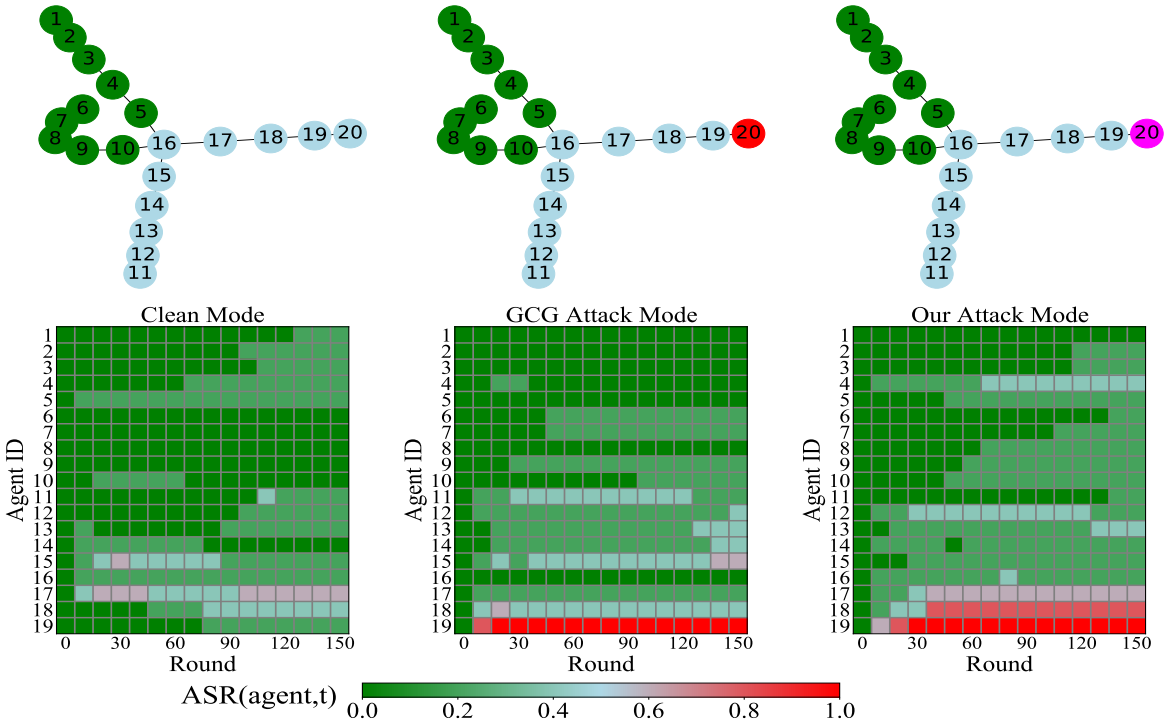

Figure 16: 50 Positive Density Agents from 20 Agents in Star Structure.

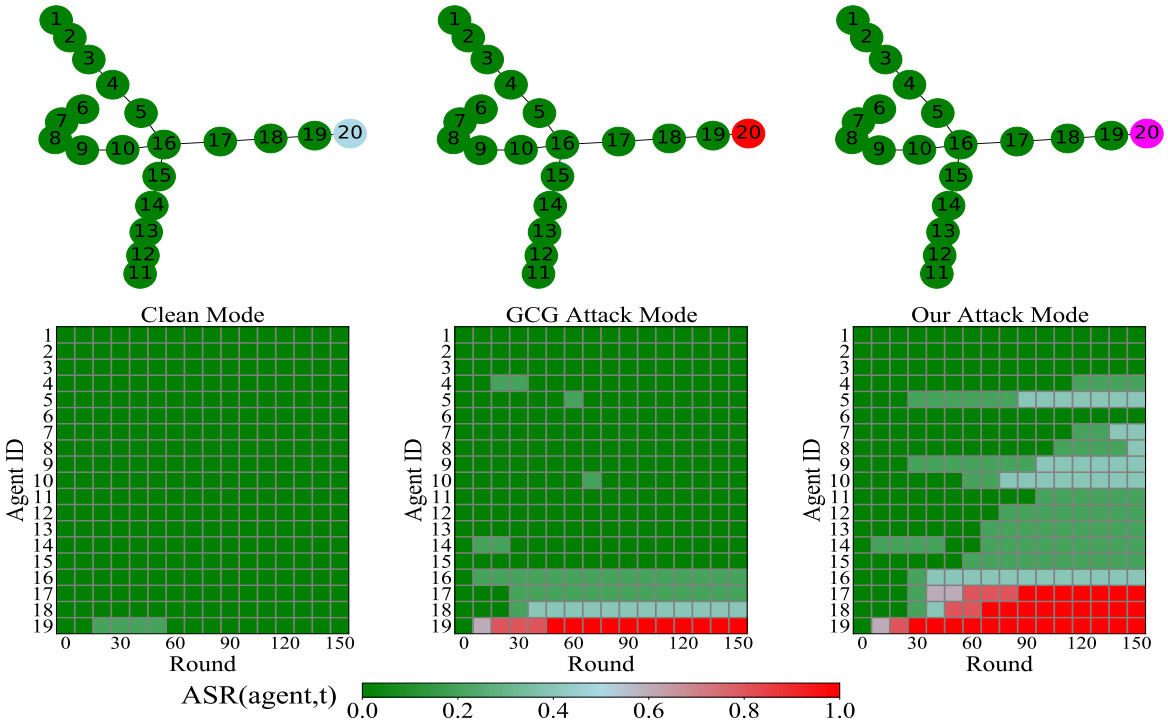

Figure 17: 99 Positive Density Agents from 20 Agents in Star Structure.

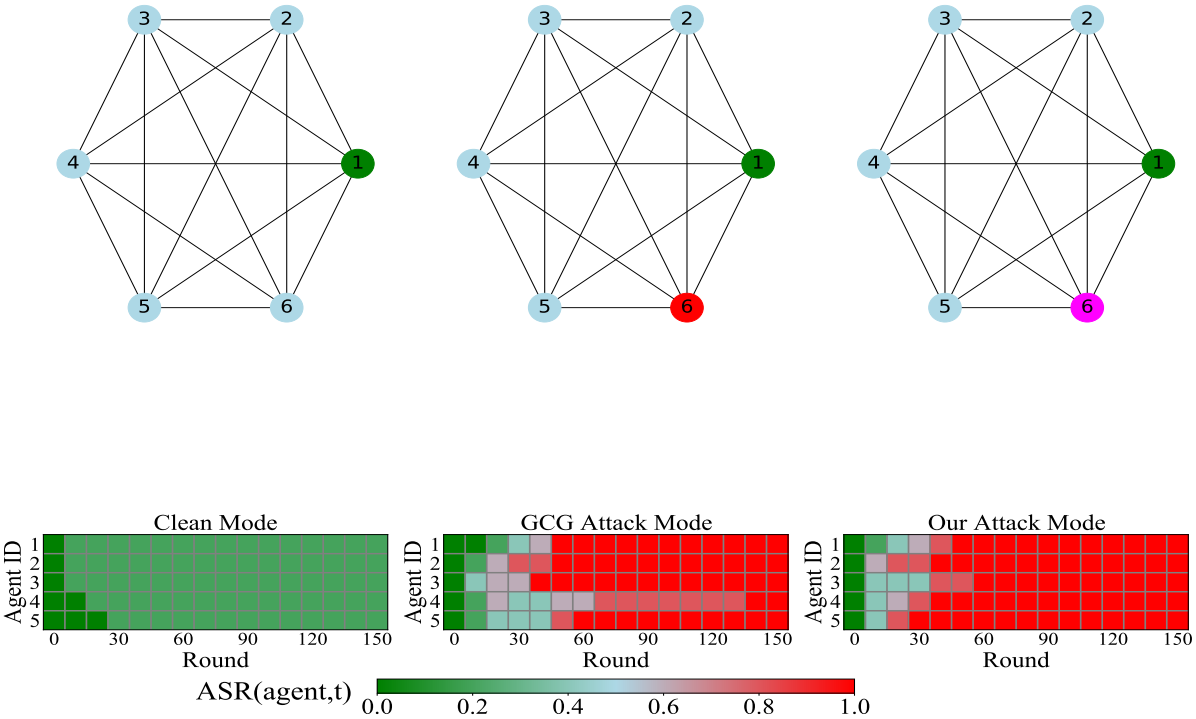

Figure 18: 1 Positive Density Agents from 6 Agents in Graph Structure.

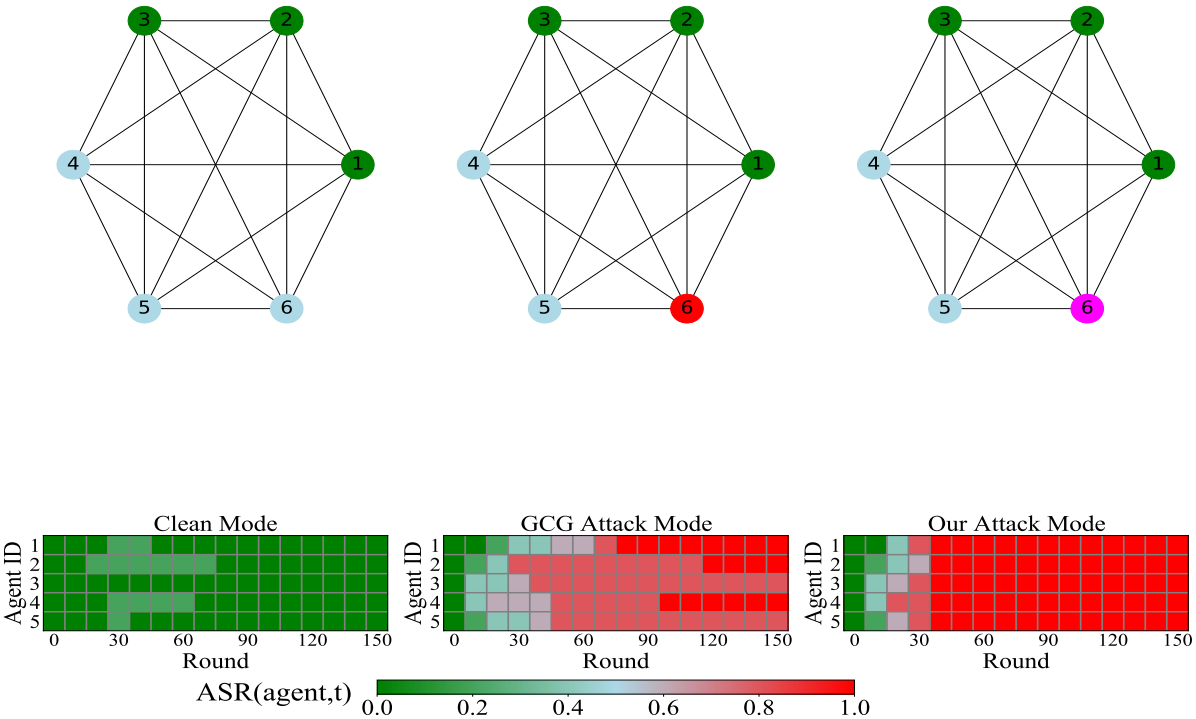

Figure 19: 50 Positive Density Agents from 6 Agents in Graph Structure.

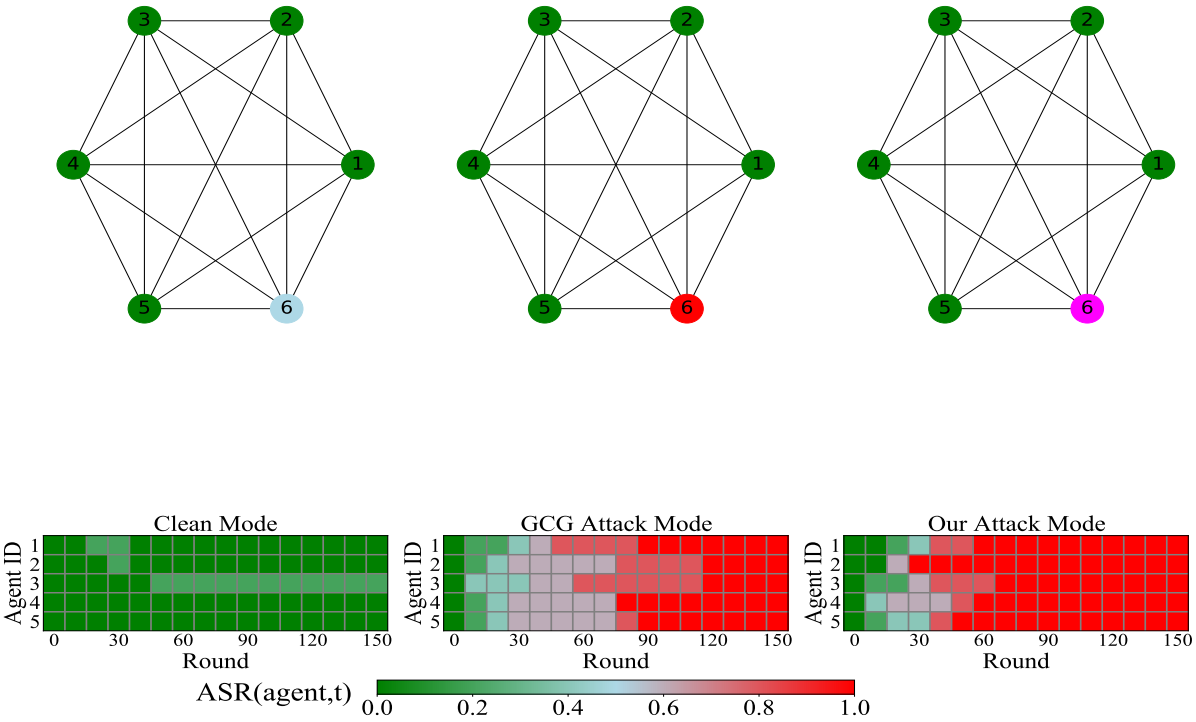

Figure 20: 99 Positive Density Agents from 6 Agents in Graph Structure.

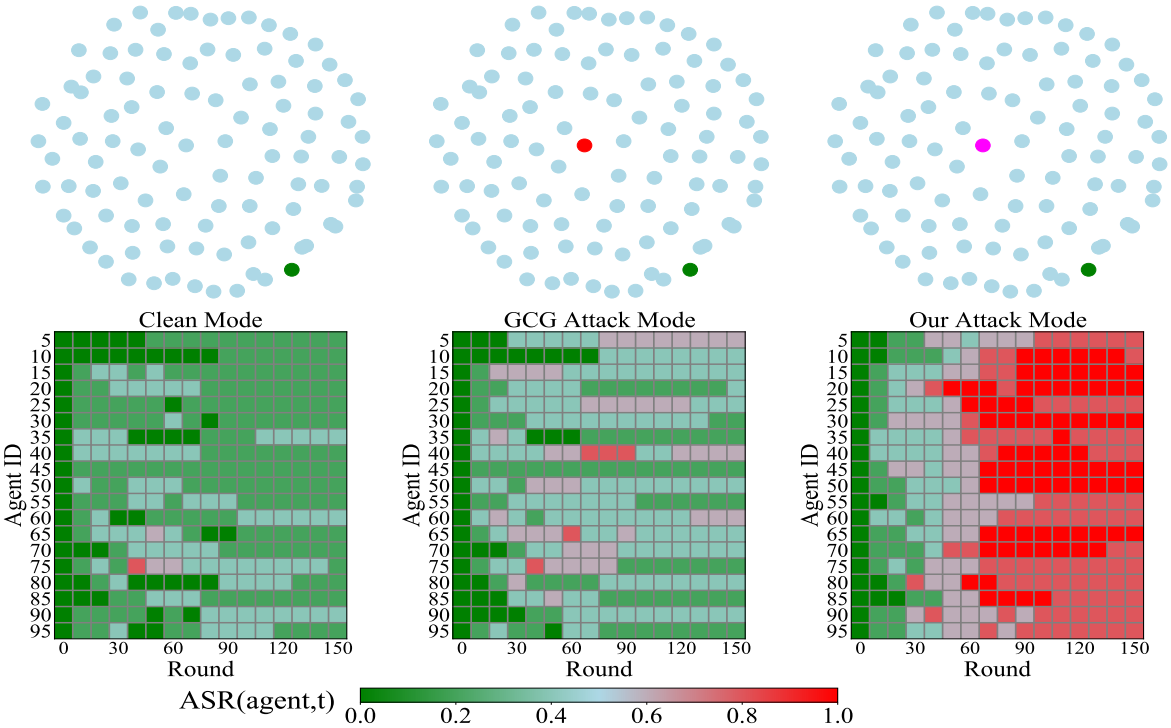

Figure 21: 1 Positive Density Agents from 100 Agents in Graph Structure. In this figure, all agents are able to communicate with each other. We sampled the infection status of 19 agents out of 100 as a demonstration.

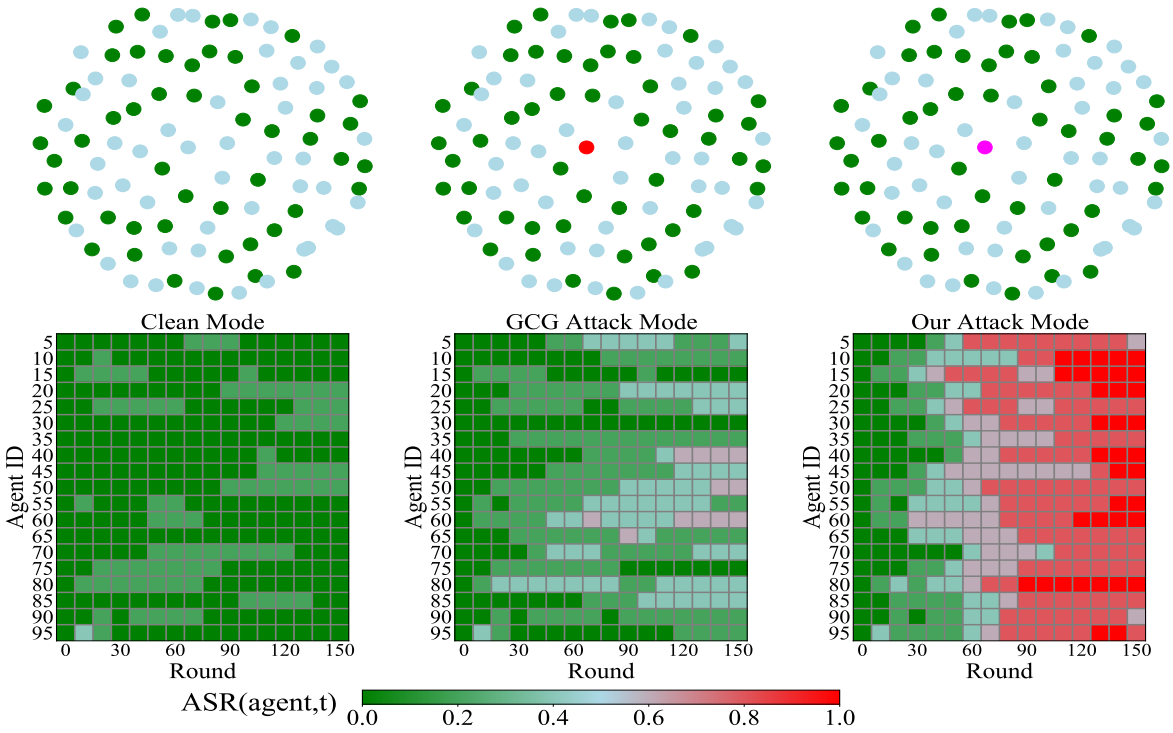

Figure 22: 50 Positive Density Agents from 100 Agents in Graph Structure. In this figure, all agents are able to communicate with each other. We sampled the infection status of 19 agents out of 100 as a demonstration.

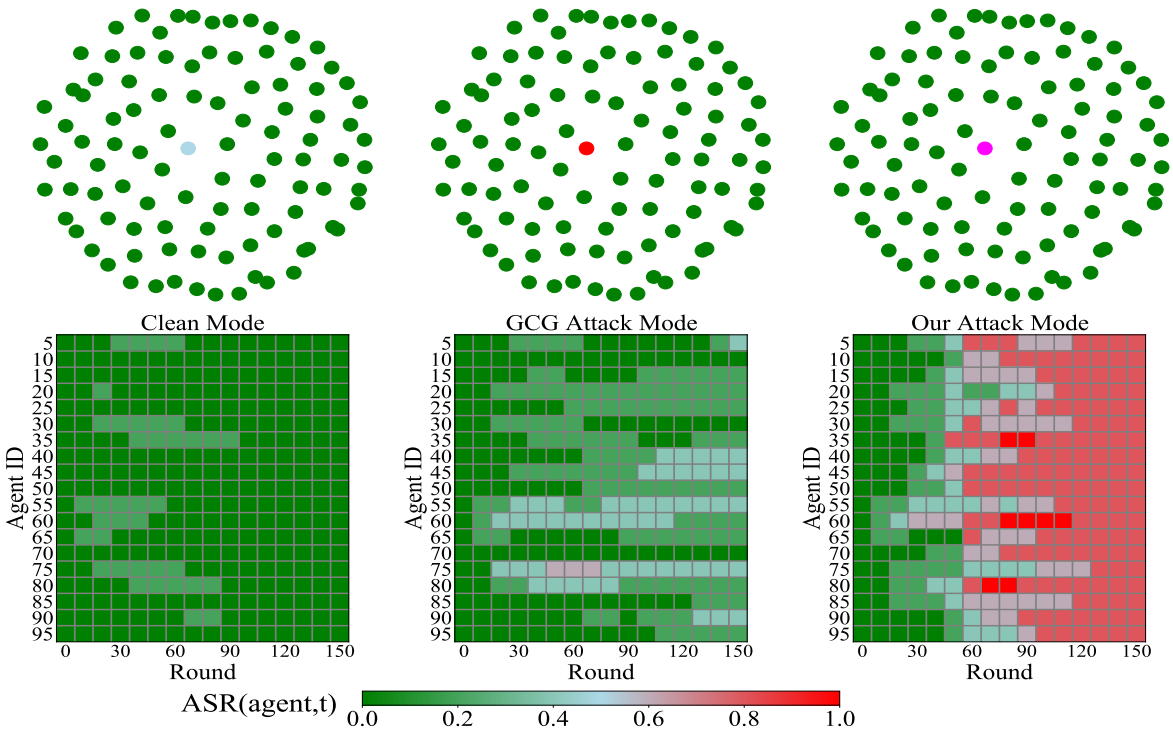

Figure 23: 99 Positive Density Agents from 100 Agents in Graph Structure. In this figure, all agents are able to communicate with each other. We sampled the infection status of 19 agents out of 100 as a demonstration.