Grey-box Adversarial Attack on Communication in Communicative Multi-agent Reinforcement Learning

Xiao Ma, *Student Member, IEEE*, Wu-Jun Li, *Member, IEEE*

***Abstract*—Effective communication is a necessary condition for intelligent agents to collaborate in multi-agent environments. Although increasing attention has been paid to communicative multi-agent reinforcement learning (CMARL), the vulnerability of the communication mechanism in CMARL has not been well investigated, especially when there exist malicious agents that send adversarial communication messages to other regular agents. Existing works about adversarial communication in CMARL focus on black-box attack where the attacker cannot access any model within the multi-agent system (MAS). However, a more practical attack is grey-box attack, where the attacker has access to the models of their controlled agents. To the best of our knowledge, no research has been conducted to investigate grey-box attacks on communication in CMARL. In this paper, we propose the first grey-box attack method on communication in CMARL, which is called victim-simulation based adversarial attack (VSAA). At each timestep, the attacker simulates a victim attacked by other regular agents' communication messages and generates adversarial perturbations on its received communication messages. The attacker then sends the aggregation of these perturbations to the regular agents through communication messages, which will induce non-optimal actions of the regular agents and subsequently degrade the performance of the MAS. Experimental results on multiple tasks show that VSAA can effectively degrade the performance of the MAS. The findings in this paper will make researchers be aware of the grey-box attack in CMARL.**

***Index Terms*—Multi-agent, reinforcement learning, communication, grey-box attack.**

I. Introduction

MULTI-AGENT reinforcement learning (MARL) has achieved promising success in a variety of challenging domains, including game playing [1]–[3], robotics [4], [5] and secure cooperative networks [6]. Effective communication is a necessary condition for intelligent agents to collaborate in multi-agent settings, especially in many real-world scenarios where there are a large number of agents [7]–[9]. Since agents can share or exchange information by communication, e.g., historical knowledge or current observation, intelligent agents in a multi-agent system (MAS) can obtain more information from communication and perform better than those without communication. Researchers have recently proposed many methods for communicative multi-agent reinforcement learning (CMARL) [2], [10]–[21], which are called CMARL methods. These CMARL methods have shown good performance.

Although these CMARL methods have achieved promising successes and have been widely used in many applications, it has been found that when the trained CMARL models are deployed in real-world applications, the communication mechanism in CMARL methods is vulnerable [21], [22] and easy to receive the attacker's attention [23]. Attackers are capable of gaining control over agents, which typically become malicious after being controlled. By manipulating the controlled agents, attackers can disrupt the collaboration among agents by sending adversarial communication messages. To illustrate the impact of such attacks, we use a warehouse robotic coordination system as an example, where each robot often has a limited observation range. In a large warehouse, multiple robots are responsible for picking up items from shelves and delivering items to designated locations. These robots need to communicate with others to make up for the lack of observation range and avoid collisions. However, when a robot is controlled by an attacker, it can generate and send adversarial communication messages to other robots. Even though these messages are adversarial, other robots might perceive them as regular communication messages. This can result in non-optimal actions that bring potential risks to the robots and the overall warehouse robotic coordination system. Hence, studying the adversarial attack on communication in CMARL methods has become necessary to ensure the robustness and safety of CMARL in practical applications.

In machine learning and cyber-security, adversarial attacks can be divided into three categories: white-box attacks, black-box attacks, and grey-box attacks. White-box attacks occur when the attacker has full knowledge of the system, such as all models. In contrast, black-box attacks do not have any system knowledge. Grey-box attacks are somewhere in between white-box attacks and black-box attacks, with the attacker having only partial knowledge of the system, such as a subset of all models. Grey-box attacks are more practical than white-box attacks and have stronger attack capabilities than black-box attacks. Grey-box attacks have been extensively studied in multiple domains, including computer vision [24], natural language processing [25], and financial trading systems [26]. A typical class of grey-box attack methods is transfer-based attacks [27], [28], which exploit the transferability of adversarial examples between models with similar architectures.

As for the design of an adversarial attack on communication in the domain of CMARL, to the best of our knowledge, there

All authors are with the National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China. Wu-Jun Li is the corresponding author. E-mail: maxiao@smail.nju.edu.cn; liwujun@nju.edu.cn.

Manuscript received March xx, 2024; revised xxxx xx, 2024.

exists only one work [23], called model-based message attack (MA). MA [23] focuses on the black-box attack in which the attacker cannot access any model within the MAS and learns an adversarial policy model to generate malicious messages that minimize negative team rewards. However, in real-world applications, it is easy for attackers who have gained control over an agent to access the model of their controlled agent. This is because, in some applications, an attacker might have access to the model of its controlling agent through granted permissions (e.g., for debugging or behavior tuning). In addition, sharing their models in the MAS, especially collaborative MAS, might be helpful for the whole MAS's coordination, which also increases the probability of access to the model of its controlled agent for the attacker. Hence, although grey-box attacks on communication in CMARL methods are practical and grey-box attacks have attracted a lot of attention in multiple domains, to the best of our knowledge, no research has been conducted to investigate the grey-box attack on communication in CMARL methods.

In this paper, we propose a novel and effective grey-box attack method on communication in CMARL, which is inspired by the following Chinese proverb,

"Employing an adversary's own techniques to achieve attack." -Zhu Xi.

This novel grey-box attack method is called victim-simulation based adversarial attack (VSAA). The main contributions of this paper are listed as follows:

- To the best of our knowledge, VSAA is the first grey-box attack method on communication in CMARL.
- In VSAA, the attacker adopts a victim-simulation approach by simulating the controlled agent as a victim subjected to its received communication messages. By generating adversarial communication messages, the attacker disrupts the collaboration among the agents in the MAS.
- We propose multiple variants of VSAA under different norms, including ℓ_1-norm, ℓ_2-norm, and ℓ_∞-norm.
- Experimental results show that VSAA can effectively degrade the performance of the MAS on multiple tasks. Even if the MAS system has a defense mechanism, VSAA is still effective.
- The findings in this paper will make researchers be aware of the grey-box attack in CMARL.

The remainder of this paper is organized as follows. Section II gives an introduction to the related work. Section III introduces the notations and definitions of different attack modes on communication in CMARL, including white-box attack, black-box attack, and grey-box attack. Section IV outlines the essential criteria that a practical attack method on communication in CMARL should fulfill, delves into the challenges of designing a grey-box attack on communication in CMARL, and introduces the proposed VSAA method. To evaluate the effectiveness of the VSAA method, experimental studies are conducted in Section V. The conclusion is given in Section VI.

We note that a shorter conference version (2 pages) of this paper appeared in [29]. Our initial conference paper provided only a preliminary discussion of the effectiveness of VSAA. In this paper, we conduct a detailed analysis of the related work, thoroughly explore the essential design criteria and challenges inherent in crafting a grey-box attack method on communication in CMARL, and dissect the inspiration behind our method, providing an intuitive illustration of its rationality. We also propose distinct variants of VSAA under different norms. In the experimental section, we deliver a more comprehensive analysis, including a didactic example, experimental results on more tasks, a sensitivity analysis of hyper-parameters, and an extensive discussion about defense mechanisms.

II. Related Work

A. CMARL Methods

Communication can help agents to cooperate toward common goals, especially for a cooperative and partially observable setting. Through communication, agents can share information among agents to obtain global information. Hence, learning to communicate is an active area in MARL, and researchers have recently proposed many methods for CMARL. According to message addressing, the existing CMARL methods can be divided into three categories, including broadcasting, targeted communication, and networked communication [30].

Broadcasting means that communication messages are sent to all agents. In other words, each agent sends messages and receives other messages from all other agents. R(D)IAL [16] sends communication messages by using a discrete communication channel to achieve communication among agents. In contrast, CommNet [10] chooses a continuous vector channel and allows agents to calculate the sum of communication messages they receive. Due to the success of CommNet, CommNet has been adopted as the backbone of many works [14], [15], [20], [23]. Broadcasting is simple and effective, but its communication efficiency has a lot of space for improvement.

Targeted communication and networked communication are proposed to improve communication efficiency. Targeted communication means that agents can decide communication time, communication content and agents that need to communicate. ATOC [17], VAIN [14], TarMAC [15] and IC3Net [20] all utilize an attention mechanism to achieve a targeted communication topology. Among these works, VAIN, TarMAC, and IC3Net can be seen as extensions to the CommNet. Furthermore, some works [2], [21] learn to send succinct messages to overcome the limitation of communication bandwidth. Networked communication means that agents communicate with their neighborhoods in the network [11]–[13]. Although networked communication can improve the communication efficiency of MASs, agents in the MAS need to predefine their neighborhoods. Therefore, broadcasting and targeted communication are still simple and widely used communication mechanisms in real-world scenarios.

B. Adversarial Attacks on Communication in CMARL

As promising progress has been achieved in CMARL, increasing attention has been paid to its potential vulnerabilities.

It has been found that the communication mechanism in the CMARL method is fragile [21] and susceptible to some external impacts [22], [23], [31], notably drawing the attention of potential attackers. Hence, the attack on communication mechanisms in CMARL has attracted some attention from researchers. In [32], researchers focus on the natural emergence of attack. In [33], researchers are concerned about the vulnerability of communication mechanisms when there is a central control to communicate with other agents. However, these works do not specifically focus on the design of adversarial (malicious) communication messages in CMARL. Another study [23] proposes a black-box attack method, MA, on communication messages where attackers cannot access any model within the MAS. In MA [23], attackers generate adversarial communication messages by collecting all agents' interactive data with the environment rather than using the model. From the perspective of attackers, collecting uncontrolled agents' interactive data with the environment is not practical. In contrast, attackers can easily access their controlled agent's data, including its model and interactive data with the environment. This type of attack, which can utilize the model of their controlled agent, is called the grey-box attack. In CMARL, although grey-box attacks are more practical than black-box attacks, no works have studied grey-box attacks on communication in CMARL. This motivates us to study grey-box attacks on communication in CMARL like VSAA.

C. Defending against Adversarial Attacks on Communication in CMARL

In recent years, researchers have begun to pay attention to defending against adversarial attacks on communication in CMARL. The work in [34] introduces the Theory of Mind as a rationalization of other agents' behaviors through observed actions to determine the trust beliefs of other agents' communicated messages. However, in MAS, the characteristic of partial observation might make agents hard to observe other agents' actions, which will limit the application of this work. In [35], researchers focus on adversarial communication in the training phase to update the learning parameter effectively. Message re-constructor (MR) [23] utilizes the model of message re-constructor to construct the communication messages. However, attackers can access the model for grey-box attacks. Hence, attackers can easily prevent their generated communication messages from being reconstructed by utilizing the model of message re-constructor. That is to say, the MR defense method is infeasible under grey-box attacks. As for ablated message ensemble (AME) [36], it uses random k-sample messages to learn a message ablation policy during the training phase, and the message ablation policy can defend against communication attacks during the evaluation phase. Under grey-box attacks, AME is a potentially feasible defense method and we would discuss its feasibility in the following section.

III. Preliminaries

A. Notation

We model a communicative multi-agent task as a decentralized partially observable Markov decision process (Dec-POMDP) [37] with communication, which can be defined by a tuple $G = \langle \mathcal{N}, T, S, O, A, P, R, Z, M, n, \gamma \rangle$. $\mathcal{N} \equiv \{0, 1, 2, ..., n-1\}$ is the finite set of agents. T is the episode time horizon. Here, we take the timestep $t \in \{1, \ldots, T\}$ as an illustration. At timestep t, $s_t \in S$ is the true state of the environment. Each agent $i \in \mathcal{N}$ has its own partial observation $o_t^{(i)} \in O$, which is drawn from the true state s_t according to the observation function $Z(s_t, i)$. Agent i sends its communication message $m_t^{(i)} \in M$ to other agents $-i$, where $M \in \mathbb{R}^D$ denotes the communication message space and $-i = \mathcal{N} \backslash \{i\}$ denotes the set of other agents except agent i. Agent i receives communication messages $\boldsymbol{m}_t^{-i} = \{m_t^{(j)} | j \in -i\}$ from other agents. Based on its partially observable state $o_t^{(i)}$ and received communication messages $\boldsymbol{m}_t^{-i}$, agent i selects an action $a_t^{(i)} \in A$ guided by the policy $\pi_{\theta_i}(a|o_t^{(i)}, \boldsymbol{m}_t^{-i})$ and then actions of all agents form a joint action $\boldsymbol{a}_t = \{a_t^{(i)} | i \in \mathcal{N}\}$. Here, A represents the action space, $\theta_i \in \Theta$ corresponds to the model parameter of agent i, and Θ represents the model space. Please note that the communication mechanism is synchronized. At timestep t, all agents have received all communications from timestep 1 to timestep $t-1$. The joint action $\boldsymbol{a}_t$ causes a transition on the environment according to the state transition function $P(s_{t+1}|s_t, \boldsymbol{a}_t)$ and results in a shared reward $r_t = R(s_t, \boldsymbol{a}_t)$. At the end of the episode, we have $\boldsymbol{\tau} = (s_1, \boldsymbol{o}_1, \boldsymbol{a}_1, r_1, \ldots, s_T, \boldsymbol{o}_T, \boldsymbol{a}_T, r_T)$.

In all CMARL methods, the goal of all agents is to find their optimal policy $\{\pi_{\theta_i^*}(a|o^{(i)}, \boldsymbol{m}^{-i}) | i \in \mathcal{N}\}$ by maximizing the total expected shared return $\sum_{t=1}^{T} \gamma^{t-1} r_t$. Here, $\gamma \in [0, 1]$ is a discount factor. After the training process of CMARL methods, agent i will be deployed with the corresponding trained policy $\pi_{\theta_i^*}(a|o^{(i)}, \boldsymbol{m}^{-i})$ with the model parameter θ_i^*. The attack on communication in CMARL typically occurs when the model of trained policies has already been deployed in the real environment.

The attackers aim to degrade the performance of the MAS by sending adversarial messages to other agents. In this paper, the agent controlled by attackers is called *malicious agent*. $\mathcal{B} \subset \mathcal{N}$ is the finite set of malicious agents with a size of b, where $0 \leq b \leq n$. Please note that $b = 0$ means that there is no malicious agent in MAS. The agent not controlled by attackers is called the *regular agent*, and $\mathcal{R} = \mathcal{N} - \mathcal{B}$ is the finite set of regular agents with a size of $n - b$. At each timestep t, a malicious agent $i \in \mathcal{B}$ sends the adversarial communication message $\tilde{m}_t^{(i)}$ to other agents $-i$. On the other hand, a regular agent $i \in \mathcal{R}$ sends the true communication message $m_t^{(i)}$. We formally denote the communication message sent by agent i at timestep t as $\hat{m}_t^{(i)}$. Hence, we have:

$$\hat{m}_t^{(i)} = \begin{cases} \tilde{m}_t^{(i)}, & \text{if agent } i \in \mathcal{B} \text{ is malicious,} \\ m_t^{(i)}, & \text{if agent } i \in \mathcal{R} \text{ is regular.} \end{cases} \tag{1}$$

B. Attack Mode

In the MAS, an agent's data encompasses both its interactive data with the environment, e.g., observations and actions, and its model. Depending on whether the attacker can access the models within the MAS, we divide attack modes into three categories:

- White-box attack: The attacker has complete access to all agents' models within the MAS.
- Black-box attack: The attacker is unable to access any agent's model within the MAS.
- Grey-box attack: The attacker can only access the models of the agents they control.

White-box attacks are the most powerful attacks since the attacker has complete knowledge about all models within the MAS. However, white-box attacks are also the least realistic attack in practice [23], [38]. In fact, white-box attacks can be viewed as a special case of grey-box attacks, where the attacker controls all agents in the MAS. For black-box attacks, the attacker does not have direct access to any agent's model. As a result, the attacker might need to rely on alternative sources of data to generate effective adversarial messages that can degrade the performance of the MAS. For instance, in MA [23], the attacker needs to collect all agents' interactive data with the environment. Hence, the practicality and attack capability of black-box attacks are limited by the way they are employed. Grey-box attacks are more practical than white-box attacks and have stronger attack capabilities than black-box attacks. When the attacker successfully controls an agent, the attacker can easily access the model of this controlled agent. In addition, we assume that the attacker can easily access data from the agents under their control. While this assumption might seem strong, it is actually weaker than that in black-box attack methods (e.g., MA) and aligns with many real-world MASs in which data encryption might be absent, weak, or vulnerable. Furthermore, even though encryption provides some security, it is not foolproof. Attackers can exploit weak encryption or other vulnerabilities to access data. Therefore, this assumption lays the foundation for analyzing attack mechanisms while acknowledging the potential for future work to explore more secure encryption methods or settings. Please note that other agents' models still remain invisible to the attacker. In Figure 1, we summarize the difference among these three categories of attacks. To the best of our knowledge, no works have studied grey-box attacks on communication in CMARL. Hence, we focus on the grey-box attack in this paper.

C. Fast Gradient Sign Method

Fast gradient sign method (FGSM) [39] is a widely applied adversarial attack technique. It leverages gradient information from the model to create adversarial perturbations by adding slight changes to the input. Given a model's loss function $V(x, y; \theta)$, where θ represents the parameters of the model, x is the input, and y is the true label of the input x. Here, we take ℓ_∞- norm as illustration. Under ℓ_∞- norm, FGSM generates adversarial examples using the following formula:

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}\left(\nabla_x V(\theta, x, y)\right),$$

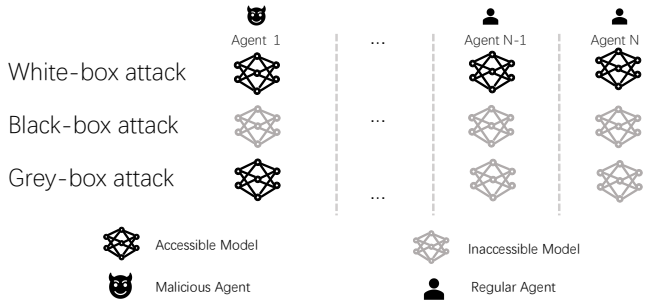


Fig. 1. The accessible models and inaccessible models in a MAS by the attacker for different attack modes. Without loss of generality, we assume agent 1 is malicious (controlled by an attacker) and other agents are regular.

where x_{adv} is the adversarial example generated. $\nabla_x V(\theta, x, y)$ is the gradient of the loss with respect to the input x and is called perturbation. ϵ is a constant that controls the magnitude of the perturbation. $\text{sign}(\cdot)$ represents the sign function. By adding perturbations aligned with the gradient's sign, FGSM maximizes the loss function, increasing the likelihood of misclassification.

IV. Method

In this section, we begin by outlining the design criteria that a practical attack method on communication in CMARL should fulfill. We then delve into the challenges in designing a grey-box attack method on communication in CMARL that satisfies these criteria. Lastly, we propose VSAA, a simple yet effective grey-box attack method on communication for CMARL.

A. Design Criteria for Attack Methods

In CMARL, attackers aim to degrade the performance of MAS by imperceptibly sending malicious communication messages. It is necessary to improve the practicality of the attack method adopted by attackers. Hence, the attack method on communication in CMARL needs to meet the following criteria:

(1) Byzantine fault tolerance: Attack methods can maintain their effectiveness even if other malicious agents are corrected and unable to send malicious communication messages.
(2) Imperceptible difference: Attack methods should ensure that the difference between true messages and malicious messages is imperceptible.
(3) Masquerade as regular agents: Malicious agents controlled by attackers should still take optimal actions and behave consistently with regular agents to evade suspicion, besides sending adversarial communication messages.

The first criterion is inspired by Byzantine fault tolerance in distributed computing [40], [41]. The criterion of byzantine fault tolerance ensures that the attack method can handle failures and maintain its effectiveness even when other malicious agents are corrected. When the difference between true

and malicious communication messages is large, people can identify the attacker. Hence, we need the criterion of imperceptible difference. This criterion enhances the stealthiness of the attack method and then enhances the practicality of the attack method. Furthermore, the third criterion ensures that the attack method cannot influence the malicious agent's actions, which makes it difficult for humans or other agents to detect their malicious intent solely based on observed actions. In a word, these design criteria are crucial for the practicality and effectiveness of attack methods on communication in CMARL.

B. Challenges

When the MAS has been deployed with pre-trained models, attackers aim to make malicious agents send malicious messages and then make other agents that have received malicious messages not take optimal actions. We find that the effectiveness of using random perturbations as an attack method to generate adversarial communication messages is typically limited, as shown in Section V-C. This highlights the necessity of designing an effective attack method to generate adversarial communication messages.

In an ideal case, malicious agent $i \in \mathcal{B}$ has knowledge of how its true message $m_t^{(i)}$ affects other agents' actions $\{\pi_{\theta_j^*}(a|o_t^{(j)}, m_t^{(i)}, \boldsymbol{m}_t^{-j\setminus\{i\}})|j \in -i\}$ at timestep t. Based on this knowledge, this malicious agent can generate a corresponding adversarial communication message $\tilde{m}_t^{(i)}$ to prevent other agents from taking optimal actions. However, it is hard for the malicious agent to have knowledge of how its true message affects other agents' actions. Here, we consider a MAS with three agents, where agent 0 is malicious due to being under the attacker's control, while agents 1 and 2 are regular. At each timestep t, agent 0 expects to know $\{\pi_{\theta_1^*}(a|o_t^{(1)}, m_t^{(0)}, m_t^{(2)}), \pi_{\theta_2^*}(a|o_t^{(2)}, m_t^{(0)}, m_t^{(1)})\}$, which can reflect the effect of the true message $m_t^{(0)}$ on other agents' actions. Leveraging this knowledge and the true message $m_t^{(0)}$, the attacker generates corresponding adversarial message $\tilde{m}_t^{(0)}$. This ideal case exemplifying attackers generating malicious agents is depicted in the left part of Figure 2. However, achieving this ideal case presents several challenges in real applications, shown as follows:

(1) Deadlock occurrence: Deadlock occurs when the $b > 1$ malicious agents need the communication message of the current timestep to construct the malicious communication message. Here, we also take the above MAS with three agents as an example and assume that there are two malicious agents, agent 0 and agent 1. While generating their adversarial communication messages, malicious agents 0 and 1 both require receiving messages from other agents. Specifically, malicious agent 0 requires receiving the message $\tilde{m}_t^{(1)}$ to generate $\tilde{m}_t^{(0)}$ and the generation of $\tilde{m}_t^{(0)}$ also requires $\tilde{m}_t^{(1)}$. Consequently, the generation of adversarial communication messages for these two malicious agents gets stuck in a deadlock situation, as shown in the right part of Figure 2. When a MAS is stuck in a deadlock, malicious agents stop taking any action that deviates from regular agents' behavior, which fails to meet Criterion 3. Please note that introducing additional communication between malicious agents to avoid deadlock is not feasible as it would make a detectable difference between malicious agents and regular agents.

(2) Inaccessible to other agents' data: In real applications, the attacker who controls malicious agent $i \in \mathcal{B}$ does not have direct access to other agents' data, including their models and interactive data with the environment. Hence, the malicious agent $i \in \mathcal{B}$ is hard to know how its true message $m_t^{(i)}$ affects other agents' actions $\{\pi_{\theta_j^*}(a|o_t^{(j)}, m_t^{(i)}, \boldsymbol{m}_t^{-j\setminus\{i\}})|j \in -i\}$ at any timestep t. As a result, the attacker cannot generate a corresponding adversarial message $\tilde{m}_t^{(i)}$ to degrade the performance of the MAS by making others not take optimal actions.

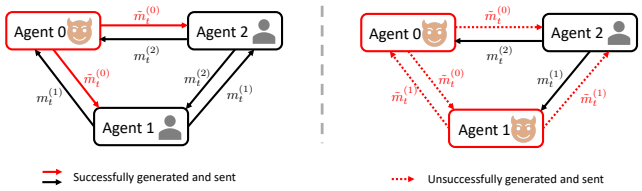


Fig. 2. The communication process in a MAS with three agents. (Left) The communication process when agent 0 is malicious ($b = 1$). (Right) The communication process when agent 0 and agent 1 are malicious ($b = 2$). The solid arrow represents the corresponding message that has been generated and sent. The dotted arrow represents the corresponding message that has not been generated and sent.

C. VSAA

In this section, we present the details of VSAA. VSAA can satisfy the design criteria and solve the challenges.

1) Using Outdated Messages: Intuitively, when two timesteps t and $t_1 \leq t$ are close, the communication messages $m_t^{(i)}$ and $m_{t_1}^{(i)}$ generated by any agent $i \in \mathcal{N}$ have certain similarities in most cases. Following [42], we assume that the parametric policy π_{θ_i} is differentiable and l_π-Lipschitz. That is to say, for any $i \in \mathcal{N}$, $o^{(i)} \in O$, $a \in A$, $\boldsymbol{m}^{-i} \in M^{n-1}$ and $\theta_i \in \Theta$, there exists a universal constant l_π, such that $\left\|\nabla \pi_{\theta_i}(a|o^{(i)}, \boldsymbol{m}^{-i})\right\| \leq l_\pi$. This assumption can be easily satisfied by many common types of policies, e.g., soft-max [43], [44] or neural network with Lipschitz and smooth activation functions [45]–[47]. Hence, there exists

$$\|\pi_{\theta_i}(a|o_t^{(i)}, \boldsymbol{m}_t^{-i}) - \pi_{\theta_i}(a|o_t^{(i)}, \boldsymbol{m}_{t_1}^{-i})\| \leq l_\pi d_{t,t_1}^{(i)}, \quad \forall\, i, t, t_1 \leq t, a, \tag{2}$$

where $d_{t,t_1}^{(i)}$ is defined as $d_{t,t_1}^{(i)} = \|\boldsymbol{m}_t^{-i} - \boldsymbol{m}_{t_1}^{-i}\|$. This implies that if the value of $d_{t,t_1}^{(i)}$ is small, the action taken by agent i will not be changed with a large possibility.

Motivated by this, we propose to use outdated yet received messages at the previous timestep $t-1$ to approximate the optimal action at each timestep t, which can avoid the occurrence of deadlock (Challenge 1). Please note that because the communication mechanism is synchronized, the

interaction at each timestep t with the environment is independent. Hence, multiple interactions over multiple timesteps with the environment do not result in interference among malicious agents. Formally, when the MAS has been deployed with the pre-trained policies $\{\pi_{\theta_i^*}(a|o^{(i)}, \boldsymbol{m}^{-i})|i \in \mathcal{N}\}$, we use $\arg\max_a \pi_{\theta_i^*}(a|o_t^{(i)}, \boldsymbol{m}_{t-1}^{-i})$ to approximate $\arg\max_a \pi_{\theta_i^*}(a|o_t^{(i)}, \boldsymbol{m}_t^{-i})$ at timestep t for each agent $i \in \mathcal{N}$.

2) Victim Simulation: Even though the deadlock occurrence can be avoided by using outdated messages, the inability to access other agents' data still hinders the design of an effective grey-box attack on communication for attackers. Prior works [39], [48], [49] have demonstrated that some adversarial attacks generated for one model might also induce classification errors in another model as well, and such property is referred to as transferability [50]–[52]. Here, the MAS has been deployed with the pre-trained policies $\{\pi_{\theta_i^*}(a|o^{(i)}, \boldsymbol{m}^{-i})|i \in \mathcal{N}\}$. Specifically, when agent $i \in \mathcal{N}$ receives an adversarial communication message, i.e., $\tilde{m}^{(j)}$, $j \in -i$, the action taken by agent i would deviate from the optimal action, shown as follows:

$$\arg\max_a \pi_{\theta_i^*}(a|o^{(i)}, \tilde{m}^{(j)}, \boldsymbol{m}^{-i\setminus\{j\}}) \neq \arg\max_a \pi_{\theta_i^*}(a|o^{(i)}, m^{(j)}, \boldsymbol{m}^{-i\setminus\{j\}}).$$

Based on the transferability of adversarial attack, $\tilde{m}^{(j)}$ not only maliciously interferes with agent i's action but also maliciously interferes with other agents' actions. This motivates us to propose a victim-simulating scenario, where malicious agent $i \in \mathcal{B}$ is a simulated victim attacked by its received communication messages $\hat{\boldsymbol{m}}^{-i}$ and then generates its adversarial communication message. The idea behind this victim-simulating scenario is also inspired by the proverbial concept of "employing an adversary's own techniques to achieve attack." That is to say, the malicious agent regards other agents as adversaries, constructs adversarial communication messages sent by adversaries, and then employs them to generate its adversarial communication message (attack). Please note that determining the similarity between the simulated agent and other agents is a difficult problem for researchers, which might impact the performance of the proposed VSAA. However, almost all CMARL methods adopt parameter sharing, which could help MAS training and alleviate this problem.

In this victim-simulating scenario, the first objective of the malicious agent $i \in \mathcal{B}$ is to construct the adversaries' adversarial communication messages. Hence, the malicious agent $i \in \mathcal{B}$ tries to generate perturbations on the adversaries' communication messages $\hat{\boldsymbol{m}}^{-i}$ with the intention of causing itself to deviate from its optimal action. The optimal action of the malicious agent i at timestep t is $\arg\max_a \pi_{\theta_i^*}(a|o_t^{(i)}, \hat{\boldsymbol{m}}_t^{-i})$. As discussed in the above Section IV-B and Section IV-C1, we use $\pi_{\theta_i^*}(a|o_t^{(i)}, \hat{\boldsymbol{m}}_{t-1}^{-i})$ to approximate $\pi_{\theta_i^*}(a|o_t^{(i)}, \hat{\boldsymbol{m}}_t^{-i})$ to avoid the deadlock occurrence and the approximate optimal action of malicious agent $i \in \mathcal{B}$ is denoted as $\hat{a}_t^{(i),*}$, shown as follows:

$$\hat{a}_t^{(i),*} = \arg\max_a \pi_{\theta_i^*}(o_t^{(i)}, \hat{\boldsymbol{m}}_{t-1}^{-i}). \tag{3}$$

As a victim, malicious agent $i \in \mathcal{B}$ simulates a scenario in which its optimal action is maliciously influenced by the received communication message $\hat{\boldsymbol{m}}_{t-1}^{-i}$. For each agent $j \in -i$, the attacker introduces a cost function to perturb received messages $\hat{m}_{t-1}^j$ and then generates a perturbation on $\hat{m}_{t-1}^j$. The cost function for the agent $j \in -i$ is formally shown as follows:

$$J(o_t^{(i)}, m^{(j)}, \hat{\boldsymbol{m}}_{t-1}^{-i\setminus\{j\}}, \hat{a}_t^{(i),*}; \theta_i^*) = -\sum_{a\in A} p(a) \log \pi_{\theta_i^*}(a|o_t^{(i)}, m^{(j)}, \hat{\boldsymbol{m}}_{t-1}^{-i\setminus\{j\}}), \tag{4}$$

where $p(a)$ is given by

$$p(a) = \begin{cases} 1, & \text{if } a = \hat{a}_t^{(i),*}, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Although our method can be combined with different techniques for crafting adversarial examples, we choose FGSM [39] for efficiently generating adversarial examples as an illustration. Here, we take the generation of perturbation on the communication message from agent $j \in -i$, $\hat{m}_{t-1}^{(j)}$, as an illustration. By minimizing and linearizing the cost function (4) around $\hat{m}_{t-1}^{(j)}$, the optimal perturbations under different norms (ℓ_∞- norm, ℓ_1- norm and ℓ_2- norm) around $\hat{m}_{t-1}^{(j)}$ are shown as follows:

- ℓ_∞- norm:

$$\eta_t^{(i,j)} = \text{sign}(\text{grad}_{i,j,t}), \tag{6}$$

where $\text{grad}_{i,j,t}$ is defined as follows:

$$\text{grad}_{i,j,t} = \left.\frac{\partial J(o_t^{(i)}, m^{(j)}, \hat{\boldsymbol{m}}_{t-1}^{-i\setminus\{j\}}, \hat{a}_t^{(i),*}; \theta_i^*)}{\partial m^{(j)}}\right|_{m^{(j)}=\hat{m}_{t-1}^j}. \tag{7}$$

- ℓ_1- norm:

$$\eta_t^{(i,j)} = D \cdot e_{f(\text{grad}_{i,j,t})}, \tag{8}$$

where $\text{grad}_{i,j,t}$ has been defined in (7). D is the dimension of the communication message space. e_f is a vector with a 1 in the $f(\text{grad}_{i,j,t})$-th coordinate and zeros elsewhere. $f(\text{grad}_{i,j,t}) = \arg\max_d |[\text{grad}_{i,j,t}]_d|, 1 \le d \le D$ and $[\text{grad}_{i,j,t}]_d$ is the d-th coorinate of $\text{grad}_{i,j,t}$.

- ℓ_2- norm:

$$\eta_t^{(i,j)} = \sqrt{D} \cdot \frac{\text{grad}_{i,j,t}}{\|\text{grad}_{i,j,t}\|_2}, \tag{9}$$

where $\text{grad}_{i,j,t}$ has been defined in (7) and D is the dimension of the message space.

Given the norm type for the perturbations mentioned above, malicious agent $i \in \mathcal{B}$ can generate adversarial communication messages $\{\hat{m}_{t-1}^{(j)} + \epsilon\eta_t^{(i,j)}|j \in -i\}$ for adversaries. Here, $\epsilon > 0$ is a constant. Based on adversarial attacks' transferability, malicious agent i can choose one of these adversarial messages as its message and send it to other agents. Though this way can also achieve the aim of degrading the performance of MAS, the difference between true messages and malicious messages might be perceptible, which would violate Criterion 2.

3) Generation of Adversarial Communication Messages: As analyzed above, malicious agent $i \in \mathcal{B}$ can obtain $n-1$ candidate perturbations $\{\eta_t^{(i,j)} | j \in -i\}$, given the type of norm. The communication message $\hat{m}_{t-1}^{(j)}$ contains the learned information by the agent j, e.g., its historical knowledge and its observations at timestep $t-1$. $\eta_t^{(i,j)}$ as the perturbation of message $\hat{m}_{t-1}^{j}$ might be used to destroy the learned information by the agent j. Hence, for the malicious agent $i \in \mathcal{B}$, adding these perturbations to its true messages can not only interfere with other agents' actions but also make adversarial communication messages imperceptible. Here, we propose to use the mean of the candidate perturbations to aggregate these perturbations, which is defined as follows:

$$\eta_t^{(i)} = \frac{1}{n-1} \sum_{j \in -i} \eta_t^{(i,j)}. \tag{10}$$

The adversarial communication message $\tilde{m}_t^{(i)}$ generated by malicious agent $i \in \mathcal{B}$ under different norms is shown as follows:

- ℓ_∞- norm:

$$\tilde{m}_t^{(i)} = m_t^{(i)} + \epsilon_\infty \operatorname{sign}(\eta_t^{(i)}); \tag{11}$$

- ℓ_1- norm:

$$\tilde{m}_t^{(i)} = m_t^{(i)} + \epsilon_1 D \cdot e_{f(\eta_t^{(i)})}; \tag{12}$$

- ℓ_2- norm:

$$\tilde{m}_t^{(i)} = m_t^{(i)} + \epsilon_2 \sqrt{D} \frac{\eta_t^{(i)}}{\|\eta_t^{(i)}\|_2}. \tag{13}$$

Here, $\epsilon_\infty > 0$, $\epsilon_1 > 0$ and $\epsilon_2 > 0$ are constants, which control the magnitude of perturbations. Subsequently, malicious agent i sends $\tilde{m}_t^{(i)}$ to others.

Please note that when multiple attackers are present simultaneously and adversarial communication messages from malicious agent i are generated by its received adversarial communication messages from other malicious agents, these adversarial communication messages are still different from true communication messages and can still mislead regular agents. Figure 3 shows the generation process of adversarial communication messages by VSAA. Algorithm 1 presents the procedure of VSAA under ℓ_∞-norm when taking malicious agent $i \in \mathcal{B}$ as an example.

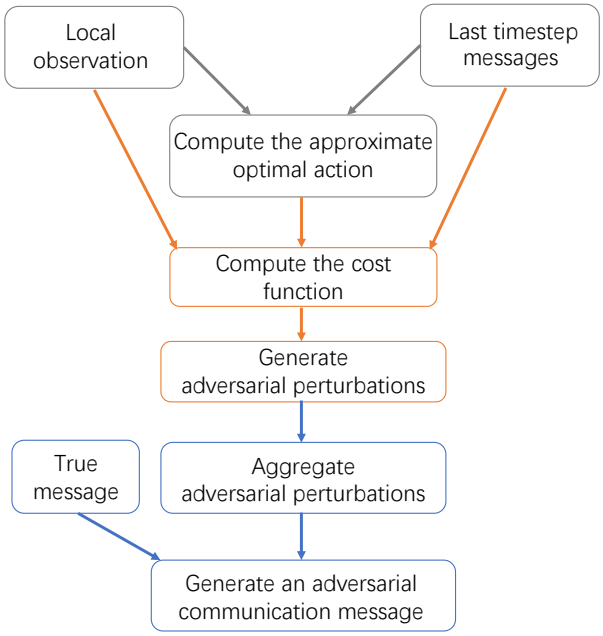


Fig. 3. The generation process of adversarial communication messages by VSAA. Best viewed in color.

Algorithm 1 Victim-Simulation based Adversarial Attack under ℓ_∞-norm

// Take malicious agent $i \in \mathcal{B}$ as an example.
Initialize pre-trained policy $\pi_{\theta_i^*}$, received messages $m_0^{-i} = \mathbf{0}$, ϵ_∞;
for $t = 1, \ldots, T$ **do**
  Observe $o_t^{(i)}$ and compute the true message $m_t^{(i)}$;
  Compute the approximate optimal action $\hat{a}_t^{(i),*}$ according to (3);
  Compute candidate perturbations $\{\eta_t^{(i,j)} | j \in -i\}$ according to (6);
  Aggregate candidate perturbations according to (10) and then get $\eta_t^{(i)}$;
  Generate the adversarial message $\tilde{m}_t^{(i)}$ according to (11);
  Send the adversarial message $\tilde{m}_t^{(i)}$ to other agents $-i$;
  Receive communication messages $\hat{\boldsymbol{m}}_t^{-i}$ from other agents $-i$;
  Compute and then execute the optimal action $\arg\max_a \pi_{\theta_i^*}(a | o_t^{(i)}, \hat{\boldsymbol{m}}_t^{-i})$;
  Receive the reward r_t from the environment;
end for

V. Experiment

In this section, we evaluate our proposed attack method, VSAA, on multiple tasks.

A. Experimental Setup

1) CMARL Methods: Following [23], we choose three CMARL methods, including CommNet [10], TarMAC [15] and NDQ [2] for experiments. These CMARL methods are used to obtain the policies deployed in the MAS. Here, we briefly introduce these three CMARL methods and their training details, shown as follows:

- CommNet [10]: CommNet is a widely used CMARL method. Each agent can use the average of incoming messages sent by other agents and learn to coordinate with other agents. The dimension of communication messages is 128. The other training hyper-parameters can be found in the original paper [10].
- TarMAC [15]: TarMAC is an extension version of CommNet. TarMAC adopts a signature-based soft-attention mechanism to allow agents to pay different attention to

different incoming communication messages. The dimension of the message signature and the message query is 32. The dimension of messages is 128. The other training hyper-parameters are also the same as those in CommNet [10].

- NDQ [2]: NDQ uses variation inference to decide what is contained in a message and whether the message is sent to other agents. NDQ can achieve good performance even if part of communication messages has been cut off. Here, the implementation of NDQ is based on the PyMARL [53] framework. Following the original paper [2], we adopt the default network structure and hyper-parameter settings of QMIX [54], unless otherwise stated. The dimension of communication messages is set to 3. We cut off 80% of messages according to the means of distribution during the evaluation phase.

For each CMARL method, we train all tasks for 20 million timesteps and obtain five shared policies [1] using five random seeds on each task separately. Please note that these shared policies are obtained without the intervention of malicious agents. When malicious agents are introduced, agents do not continue to refine their policies. These trained policies have been deployed in the MAS during the evaluation phase. The number of evaluated episodes is set to 1000 for each policy. We report win-rate as results for all tasks, where win-rate is the percentage of won episodes. All results are averaged over five random seeds.

2) Environments: We consider two environments, predator prey (PP) [20] and traffic junction (TJ) [10], for our analysis and experiments. These two environments are widely adopted to evaluate the effectiveness of communication mechanisms in CMARL.

a) Predator Prey (PP): In the predator-prey [20] environment, N predators (agents) try to reach a stationary prey on a $U \times U$ grid. The goal of agents is to reach the prey. The predators have a limited vision v. When $v = 0$, agents only know their own current locations. Each predator has five available actions: *Up*, *Down*, *Left*, *Right*, and *Stay*. When a predator reaches the prey, the predator will stay in the position of the prey until the episode ends at the maximum episode step T. The total reward of the team at each timestep t can be written as: $r_t = \frac{1}{n}\sum_{i=1}^{n} \delta_i r_{\text{explore}} + (1-\delta_i) n_t r_{\text{prey}}$, where δ_i denotes whether agent i has reached the prey or not and n_t is the number of agents reached any prey at timestep t. $r_{\text{explore}} = -0.05$ is a negative timestep reward to prevent agents from being lazy, and $r_{\text{prey}} = 0.05$ is a positive reward when an agent reaches the prey. We choose four tasks in this environment, including PP-2, PP-3, PP-7, and PP-10. The configurations of these tasks are shown in Table I.

b) Traffic Junction (TJ): In the traffic junction (TJ) [10] environment, N agents (cars) enter and cross junctions in a $U \times U$ grid, where each agent has a limited vision v. The goal of agents is to pass the junction without any collision. With a probability p_{arrive}, all cars enter a junction from one of the E possible entry points. Following [20], we also utilize curriculum learning [55] to make the training process easier. The p_{arrive} is kept at the start value till the first 12.5% of total training timesteps and is then linearly increased till the end value during the course from 12.5% to 62.5% of total training timesteps. At each timestep, each car in the grid has two available actions: *Gas* and *Brake*. Once a car reaches the edge of the grid, the car will be removed from the grid. The total reward of the team at each timestep t can be written as: $r_t = c_t r_{\text{coll}} + \sum_{i=1}^{n_t} t_i r_{\text{time}}$. Here, c_t is the number of car collisions at timestep t, $r_{\text{coll}} = -10$ is the penalty reward when two cars collide, t_i is time spent by car i in the grid, and $r_{\text{time}} = -0.01$ is a negative timestep reward. We use two tasks in this environment, including TJ-5 and TJ-10. The configurations of these two tasks are shown in Table II. Figure 4 shows a snapshot of the TJ-5 task.

TABLE I
CONFIGURATIONS OF DIFFERENT TASKS ON THE PREDATOR-PREY ENVIRONMENT.

Tasks	PP-2	PP-3	PP-7	PP-10
Number of agents N	2	3	7	10
Dimension of grid U	3	5	10	10
Vision v	0	0	1	1
Maximum episode step T	10	20	40	40

TABLE II
CONFIGURATIONS OF DIFFERENT TASKS ON THE TRAFFIC JUNCTION ENVIRONMENT.

Tasks	TJ-5	TJ-10
Number of agents N	5	10
Dimension of grid U	7	14
Vision v	3	3
Maximum episode step T	20	40
P_{arrive}	start: 0.1 end: 0.3	start: 0.05 end: 0.2
Type of roads	1-way	2-way
Number of possible arrival points E	2	4

3) Baselines: To the best of our knowledge, there exists only one work for studying attacks on communication in CMARL, which proposes a black-box attack called model-based message attack (MA) [23]. Please note that our work, VSAA, is the first grey-box attack method on communication in CMARL. Besides MA, we design two other black-box attack methods as baselines, random message attack (RA) and outdated message attack (OA). These two black-box attacks have a large possibility of occurring in communication in the real world. We also design a white-box attack method (WA), in which the attacker has access to all agents' models and all agents' interactive data from the environment. Please note that WA is actually a special case of our grey-box attack method when the attacker can access all agents' models and all agents' interactive data from the environment. Although WA is impractical in the real world, WA reflects the upper bound of the attack capability. Here, we take the malicious agent $i \in \mathcal{B}$ as an example to introduce how the adversarial communication message generated by these four attack methods at timestep t.

- RA: RA is a common method to generate perturbations. At timestep t, malicious agent $i \in \mathcal{B}$ uses a random vector as the perturbation $\xi_t^{(i),\text{RA}}$ on true communication

[1]CommNet, TarMAC, and NDQ all follow the corresponding original papers in using parameter sharing to speed up the training process.

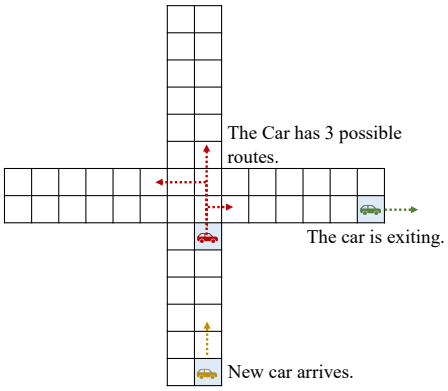


Fig. 4. A snapshot of the TJ-5 task.

message $m_t^{(i)}$. Here, each element of $\xi_t^{(i),\text{RA}}$ is sampled from a univariate Gaussian distribution of mean 0 and variance 1.

- OA: OA usually occurs when the communication is delayed, which might degrade the performance of the MAS. Here, at timestep t, malicious agent $i \in \mathcal{B}$ uses the communication $m_{t-1}^{(i)}$ to generate the perturbation of the current communication $m_t^{(i)}$. Formally, the perturbation on communication $m_t^{(i)}$ is defined as $\xi_t^{(i),\text{OA}} = m_{t-1}^{(i)}$.
- MA: MA [23] generates malicious messages by learning an adversarial policy model. The adversarial policy model is trained to generate the perturbation by minimizing the negative team reward. Here, we adopt PPO [56] to train the adversarial policy with $200K$ timesteps. For detailed training details, we follow the original paper [23]. Malicious agent $i \in \mathcal{B}$ uses the learned adversarial policy to generate a perturbation $\xi_t^{(i),\text{MA}}$ at timestep t. Please note that MA needs all agents' interactive data with the environment, which is not practical in the real world.
- WA: Here, we suppose that the whole MAS is a white box for the attacker. The attacker who controls malicious agent $i \in \mathcal{B}$ has access to all agents' models and interactive data with the environment. We introduce a new cost function, which is defined as follows:

$$G(\boldsymbol{o}_t^{-i}, m^{(i)}, \boldsymbol{m}_t^{-i}, \boldsymbol{a}_t^{-i,*}; \boldsymbol{\theta}_{-i}^*) = -\sum_{j \in -i} \log \pi_{\theta_j^*}(a_t^{(j),*} | o_t^{(j)}, m^{(i)}, \boldsymbol{m}_t^{-j \setminus \{i\}}), \tag{14}$$

where $\boldsymbol{a}_t^{-i,*} = \{a_t^{(j),*} | j \in -i\}$, and $a_t^{(j),*} = \arg\max_a \pi_{\theta_j^*}(a | o_t^{(j)}, \boldsymbol{m}_t^{-j})$, $j \in -i$. We still use FGSM [39] as the technique for crafting adversarial examples. At timestep t, by minimizing and linearizing $G(\boldsymbol{o}_t^{-i}, m^{(i)}, \boldsymbol{m}_t^{-i}, \boldsymbol{a}_t^{-i,*}; \boldsymbol{\theta}_{-i}^*)$, the perturbation around $m_t^{(i)}$ is defined as follows:

$$\xi_t^{(i),\text{WA}} = \left. \frac{\partial G(\boldsymbol{o}_t^{-i}, m^{(i)}, \boldsymbol{m}_t^{-i}, \boldsymbol{a}_t^{-i,*}; \boldsymbol{\theta}_{-i}^*)}{\partial m^{(i)}} \right|_{m^{(i)} = m_t^{(i)}}. \tag{15}$$

Please note that WA is a special case of VSAA in the ideal case, which reflects the upper bound of the VSAA's attack capability. However, WA is inapplicable in the realistic case.

When the perturbation $\xi_t^{(i)}$ around $m_t^{(i)}$ is generated by one of the above attack methods, we also use different norms to generate the adversarial communication messages similar to (11)-(13). Please note that the difference is using $\xi_t^{(i)}$ to replace $\eta_t^{(i)}$ in (11)-(13). Malicious agent $i \in \mathcal{B}$ sends the adversarial communication message $\tilde{m}_t^{(i)}$ to other agents.

B. Didactic Example

In this section, we use the PP-2 task to illustrate how VSAA, under different norms, degrades the performance of the MAS when the MAS has been deployed with the pre-trained policies by CommNet. PP-2 consists of two homogeneous agents, namely Agent 1 and Agent 2. Without loss of generality, we assume that Agent 1 is malicious. The results of all attack methods under different norms are shown in Figure 5, where $\epsilon_\infty = 0.3$, $\epsilon_1 = 0.1$, $\epsilon_2 = 0.3$ and $b = 1$. We find that the attack effect of the three black-box attack methods is limited, especially under ℓ_∞-norm and ℓ_2-norm. Although WA does exhibit a significant reduction in the win rate of the MAS, WA is impractical in real-world scenarios. We can find that our method, VSAA, can decrease the win rate of the MAS on the PP-2 task under all norms. This verifies that VSAA is an effective attack method on communication in CMARL. In the following content of this subsection, we compare the episodes, policies, and messages under two conditions: one with the VSAA method and one without the VSAA method. Unless otherwise specified, we focus on ℓ_∞-norm.

1) Comparing Episodes with and without VSAA: Figure 6a and Figure 6b respectively show the episode with VSAA (denoted by $\boldsymbol{\tau}_{\text{VSAA}}$) and the episode without attack (denoted by $\boldsymbol{\tau}$) starting from the same initial state. In $\boldsymbol{\tau}_{\text{VSAA}}$, the actions of malicious agent 1 remain unchanged compared to $\boldsymbol{\tau}$. However, Agent 2 takes actions that make itself deviate from all optimal paths, since Agent 2 utilizes adversarial communication messages $\{\tilde{m}_t^{(1)} | t = 1, ..., 10\}$. Figure 6b shows one of the optimal paths of Agent 2. We can find that in Figure 6a, even when Agent 2 is close enough to the prey, it fails to reach the prey until the episode ends. Comparing episodes with and without VSAA demonstrates that the malicious agent can still masquerade as a regular agent (Criteria 3) when it uses VSAA.

2) Comparing Policies with and without VSAA: In Figure 7b, we show the specific value of agents' policies with and without VSAA (ℓ_∞-norm) at the state s_3, which is chosen from $\boldsymbol{\tau}_{\text{VSAA}}$ and is shown in Figure 7a. When malicious Agent 1 sends the adversarial message $\tilde{m}_3^{(1)}$ generated by VSAA to Agent 2, the optimal action of Agent 2 is changed from *Down* to *Left*. However, at the current state, the action *Left* has no contribution to reach the prey for Agent 2. Furthermore, the malicious Agent 1's policy is the same for cases with and without VSAA, demonstrating that VSAA can meet Criteria 3.

Moreover, we use $\tilde{m}_3^{(1)}$ at state s_3 under different norms for VSAA as a didactic example to illustrate the

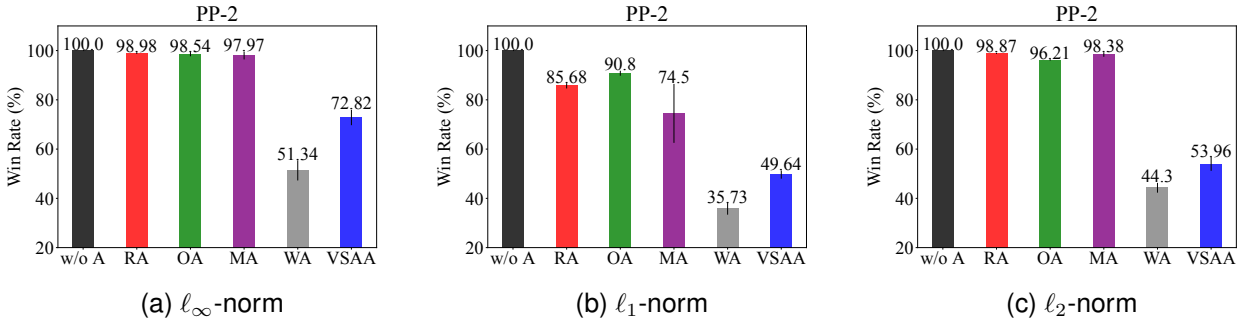


Fig. 5. Results of CommNet with different attack methods on the PP-2 task. The results are summarized over five random seeds. 'w/o A' denotes 'without attack'. Here, $\epsilon_\infty = 0.3$, $\epsilon_1 = 0.1$, and $\epsilon_2 = 0.3$.

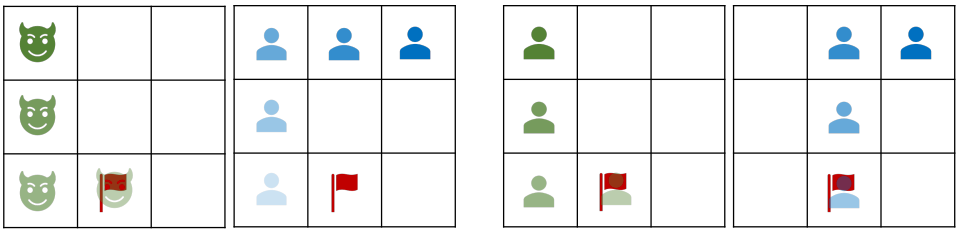
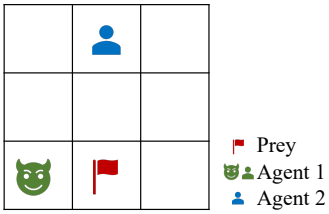


Fig. 6. (a) Behaviors of agents when malicious Agent 1 sends adversarial communication messages crafted by VSAA. The lighter icon represents the more recent agent position. (b) Behaviors of agents when there are no malicious agents. The lighter icon represents the more recent agent position.



w/o Attack	Up	Right	Down	Left	Stay
$\pi_{\theta_1^*}(a\|o_3^{(1)}, m_3^{(2)})$	6.92e-9	**9.99e-1**	1.36e-7	2.19e-9	1.69e-7
$\pi_{\theta_2^*}(a\|o_3^{(2)}, m_3^{(1)})$	1.12e-9	3.68e-9	**9.99e-1**	3.92e-5	4.19e-6

w/ VSAA	Up	Right	Down	Left	Stay
$\pi_{\theta_1^*}(a\|o_3^{(1)}, m_3^{(2)})$	6.92e-9	**9.99e-1**	1.36e-7	2.19e-9	1.69e-7
$\pi_{\theta_2^*}(a\|o_3^{(2)}, \tilde{m}_3^{(1)})$	1.45e-5	1.50e-9	7.22e-4	**9.99e-1**	1.39e-5

(b)

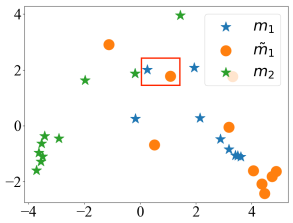


Fig. 7. (a) State s_3. (b) Comparing $\{\pi_{\theta_1^*}(a|o_3^{(1)}, m_3^{(2)}), \pi_{\theta_2^*}(a|o_3^{(2)}, m_3^{(1)})\}$ and $\{\pi_{\theta_1^*}(a|o_3^{(1)}, m_3^{(2)}), \pi_{\theta_2^*}(a|o_3^{(2)}, \tilde{m}_3^{(1)})\}$ at state s_3 on the PP-2 task. (c) The first two principal components of all messages $\{m_t^{(1)}, \tilde{m}_t^{(1)}, m_t^{(2)} | t = 1, \ldots, 10\}$ computed by PCA in the episode $\boldsymbol{\tau}_{\text{VSAA}}$. The points in the red box are $m_3^{(1)}$ and $\tilde{m}_3^{(1)}$.

specific process of generating the adversarial communication message. For the malicious Agent 1, its true communication message $m_3^{(1)}$, with a dimension of $D = 128$, is $[-0.7454, -0.8428, \ldots, 0.3381, \ldots]$. Here, for clarity, we present the value rounded to four decimal places and we only present portions of the $m_3^{(1)}$, specifically the first two dimensions and the 37-th dimension, where $f(\text{grad}_{1,2,3}) = 37$. Based on (7), we get $\text{grad}_{1,2,3} = [-0.0001, -0.0002, \ldots, -0.0023, \ldots]$. Please note that $\epsilon_\infty = 0.3$, $\epsilon_1 = 0.1$, and $\epsilon_2 = 0.3$. The detailed generation of the adversarial communication message $\tilde{m}_3^{(1)}$ under different norms is as follows:

- ℓ_∞- norm: Based on (6), we get the candidate perturbation [2]:

$$\eta_3^{(1,2)} = [-1.0, -1.0, \ldots, -1.0, \ldots].$$

[2]In this didactic example, the MAS has only two ($n = 2$) agents, resulting in one ($n - 1$) candidate perturbation.

Based on (10), we aggregate the candidate perturbation and get:

$$\eta_3^{(1)} = [-1.0, -1.0, \ldots, -1.0, \ldots].$$

Based on (11), we can get that:

$$\begin{aligned}\tilde{m}_3^{(1)} &= m_3^{(1)} + \epsilon_\infty \operatorname{sign}(\eta_3^{(1)}) \\ &= [-0.7454, -0.8428, \ldots, 0.3381, \ldots] + \\ &\quad 0.3 \times [-1.0, -1.0, \ldots, -1.0, \ldots] \\ &= [-1.0454, -1.1428, \ldots, 0.0381, \ldots].\end{aligned}$$

- ℓ_1- norm: Based on (8) and (10), we have:

$$\eta_3^{(1)} = \eta_3^{(1,2)} = [0.0, 0.0, \ldots, -128.0, \ldots].$$

TABLE III
RESULTS OF ALL ATTACK METHODS ON COMMNET. EACH ATTACK METHOD IS REPORTED WITH MEAN ± STANDARD DEVIATION, WHICH IS SUMMARIZED OVER FIVE RANDOM RUNS. 'W/O' DENOTES 'WITHOUT'.

Task (b)	Norm Type	ϵ	Win-Rate (%)					
			w/o Attack	RA	OA	MA	WA	VSAA
PP-3 (1)	ℓ_∞-norm	0.3	97.57 ± 3.79	95.73 ± 5.09	93.71 ± 4.37	89.80 ± 0.80	61.50 ± 6.05	$\mathbf{70.79 \pm 7.18}$
	ℓ_1-norm	0.1		82.77 ± 4.74	75.91 ± 14.61	72.14 ± 1.72	22.96 ± 3.93	$\mathbf{26.30 \pm 5.11}$
	ℓ_2-norm	0.3		95.55 ± 4.65	92.34 ± 5.41	89.34 ± 1.13	42.77 ± 6.40	$\mathbf{50.71 \pm 8.22}$
PP-7 (3)	ℓ_∞-norm	0.3	99.83 ± 0.19	99.77 ± 0.18	99.59 ± 0.21	98.31 ± 2.09	69.08 ± 5.1	$\mathbf{91.33 \pm 3.27}$
	ℓ_1-norm	0.1		95.14 ± 1.11	96.31 ± 3.66	65.54 ± 27.18	8.25 ± 4.35	$\mathbf{41.22 \pm 18.46}$
	ℓ_2-norm	0.3		99.78 ± 0.12	99.59 ± 0.54	99.69 ± 0.38	33.08 ± 6.89	$\mathbf{77.67 \pm 8.77}$
PP-10 (4)	ℓ_∞-norm	0.3	99.50 ± 0.75	99.40 ± 1.00	99.11 ± 1.03	97.85 ± 2.09	72.01 ± 8.09	$\mathbf{93.34 \pm 4.37}$
	ℓ_1-norm	0.1		95.95 ± 3.33	96.95 ± 2.95	68.46 ± 28.94	2.47 ± 0.94	$\mathbf{35.61 \pm 8.02}$
	ℓ_2-norm	0.3		99.36 ± 1.02	99.20 ± 1.29	96.00 ± 4.70	22.74 ± 5.30	$\mathbf{80.86 \pm 7.01}$
TJ-5 (2)	ℓ_∞-norm	0.3	99.38 ± 1.00	99.07 ± 1.28	98.10 ± 1.53	98.92 ± 1.62	65.61 ± 6.34	$\mathbf{88.97 \pm 1.97}$
	ℓ_1-norm	0.1		89.01 ± 2.56	93.49 ± 1.93	97.88 ± 1.34	56.13 ± 9.64	$\mathbf{66.53 \pm 10.63}$
	ℓ_2-norm	0.3		98.84 ± 1.19	97.97 ± 0.87	99.02 ± 1.14	74.68 ± 5.57	$\mathbf{75.08 \pm 7.63}$
TJ-10 (4)	ℓ_∞-norm	0.3	89.77 ± 4.62	89.17 ± 4.62	86.70 ± 6.26	89.23 ± 5.88	21.30 ± 2.54	$\mathbf{67.17 \pm 3.33}$
	ℓ_1-norm	0.1		70.05 ± 7.43	74.64 ± 6.26	77.54 ± 9.13	5.03 ± 2.90	$\mathbf{20.00 \pm 11.37}$
	ℓ_2-norm	0.3		88.39 ± 5.53	85.25 ± 5.35	88.31 ± 5.27	14.09 ± 2.38	$\mathbf{50.06 \pm 10.61}$

Based on (12), we have:

$$\begin{aligned}\tilde{m}_3^{(1)} &= m_3^{(1)} + \epsilon_1 D \cdot e_{f(\eta_3^{(1)})} \\ &= [-0.7454, -0.8428, \ldots, 0.3381, \ldots] + \\ &\quad 0.1 \times 128 \times [0.0, 0.0, \ldots, -1.0, \ldots] \\ &= [-0.7454, -0.8428, \ldots, 12.4619, \ldots].\end{aligned}$$

- ℓ_2- norm: Based on (9) and (10), we have:

$$\eta_3^{(1)} = \eta_3^{(1,2)} = [-0.1461, -0.3443, \ldots, -3.2992, \ldots].$$

Based on (13), we have:

$$\begin{aligned}\tilde{m}_3^{(1)} &= m_2^{(1)} + \epsilon_2 \sqrt{D} \frac{\eta_3^{(1)}}{\|\eta_3^{(1)}\|_2} \\ &= [-0.7454, -0.8428, \ldots, 0.3381, \ldots] + \\ &\quad 0.3 \times \sqrt{128} \times [-0.0129, -0.0304, \ldots, -0.2916, \ldots] \\ &= [-0.7893, -0.9461, \ldots, -0.6516, \ldots].\end{aligned}$$

3) Comparing Messages with and without VSAA: Figure 7c shows the first two principal components of all communication messages $\{m_t^{(1)}, \tilde{m}_t^{(1)}, m_t^{(2)}\}_{t=1}^{10}$ in $\boldsymbol{\tau}_{\text{VSAA}}$ calculated by principal component analysis (PCA). The adversarial communication messages $\{\tilde{m}_t^{(1)} | t = 1, \ldots, 10\}$ are still close to the true communication messages $\{m_t^{(1)} | t = 1, \ldots, 10\}$. In particular, the red box in Figure 7c highlights the communication messages $m_3^{(1)}$ and $\tilde{m}_3^{(1)}$ at state s_3 (shown in Figure 7a). These communication messages are visually similar, but the message $\tilde{m}_3^{(1)}$ generated by VSAA successfully misleads Agent 2 to take a non-optimal action. This observation demonstrates that the adversarial communication messages generated by VSAA are both imperceptible and effective in influencing the agent's behavior, which meets Criterion 2.

C. Results

In this section, we summarize the results of CMARL methods under different attack methods, which are shown in Table III, Table IV and Table V [3]. Here, the number of malicious agents b is set to $\lceil \frac{n}{3} \rceil$, where n represents the total number of agents in the MAS. The specific values of b for each task are provided in Table III, Table IV and Table V. We can find that in the PP and TJ environments, VSAA always surpasses other attack baselines, with the exception of the white-box method which is actually an unrealistic case. VSAA exhibits superior performance in degrading the effectiveness of all CMARL methods, regardless of the chosen norm for measuring perturbation. Hence, VSAA is an effective attack method on communication in CMARL.

D. Influencing Factors

In this section, we explore the influence of parameter sharing and communication dependency on the effectiveness of our VSAA method. Here, we utilize CommNet on the TJ-5 task as our test environment and ℓ_∞-norm for illustration. We aim to understand whether these factors have an influence on the performance and robustness of VSAA.

1) The Influence of Parameter Sharing: We conduct an experiment to investigate the influence of parameter sharing. Here, we assume that the model parameters between regular and malicious agents are different, indicating that the malicious agents do not have access to the model parameters of regular agents. Please note that malicious agents still share model parameters, as do regular agents. Malicious agents and regular agents only share the model architecture. Here, we simply deploy different model parameters trained by different seeds for malicious agents and regular agents. Figure 8 illustrates the performance of the MAS without any attack, the performance under our VSAA method, and the performance under other attack methods. We observe that even when we relax the assumption about parameter sharing, VSAA remains effective in degrading the performance of the MAS. This

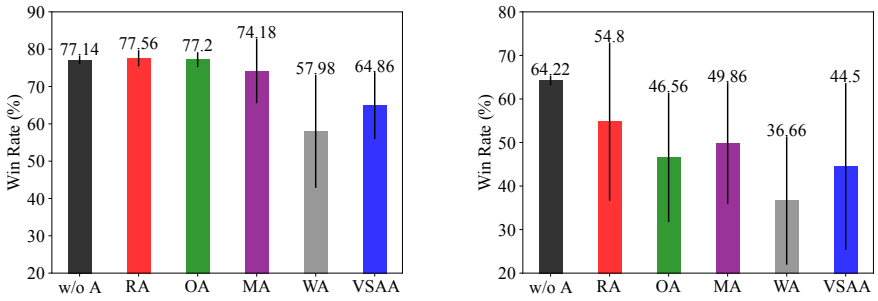
[3]On the PP-10 task, we observed that for NDQ [2], the performance when using VDN [1] as its network structure is better than that using QMIX [54]. Therefore, we choose VDN as the network structure for NDQ on the PP-10 task.

TABLE IV
RESULTS OF ALL ATTACK METHODS ON TARMAC. EACH ATTACK METHOD IS REPORTED WITH MEAN ± STANDARD DEVIATION, WHICH IS SUMMARIZED OVER FIVE RANDOM RUNS. 'W/O' DENOTES 'WITHOUT'.

Task (b)	Norm Type	ϵ	Win-Rate (%)					
			w/o Attack	RA	OA	MA	WA	VSAA
PP-3 (1)	ℓ_∞-norm	0.3	97.54 ± 1.40	89.29 ± 5.61	93.21 ± 6.96	92.96 ± 4.48	48.27 ± 4.89	$\mathbf{61.60 \pm 6.00}$
	ℓ_1-norm	0.1		73.75 ± 6.80	85.00 ± 3.87	82.00 ± 15.65	29.96 ± 3.59	$\mathbf{38.42 \pm 10.09}$
	ℓ_2-norm	0.3		88.90 ± 5.27	95.85 ± 2.22	92.80 ± 6.97	38.38 ± 4.02	$\mathbf{40.56 \pm 3.75}$
PP-7 (3)	ℓ_∞-norm	0.3	99.84 ± 0.15	98.57 ± 1.56	93.71 ± 5.54	99.40 ± 0.21	67.14 ± 8.81	$\mathbf{77.43 \pm 6.29}$
	ℓ_1-norm	0.1		91.14 ± 2.77	90.86 ± 4.10	92.12 ± 6.95	40.00 ± 13.31	$\mathbf{44.00 \pm 12.89}$
	ℓ_2-norm	0.3		99.14 ± 1.14	91.43 ± 5.19	99.56 ± 0.08	53.71 ± 6.86	$\mathbf{64.57 \pm 9.71}$
PP-10 (4)	ℓ_∞-norm	0.3	96.77 ± 4.88	96.73 ± 4.23	89.62 ± 12.32	96.72 ± 4.75	56.20 ± 12.96	$\mathbf{65.79 \pm 11.21}$
	ℓ_1-norm	0.1		74.46 ± 9.59	78.93 ± 13.52	92.08 ± 9.23	23.37 ± 5.72	$\mathbf{23.54 \pm 4.32}$
	ℓ_2-norm	0.3		96.50 ± 4.71	91.82 ± 8.33	96.56 ± 4.57	39.23 ± 9.34	$\mathbf{45.86 \pm 9.82}$
TJ-5 (2)	ℓ_∞-norm	0.3	99.98 ± 0.04	99.88 ± 0.15	99.52 ± 0.77	99.92 ± 0.16	72.16 ± 15.59	$\mathbf{92.14 \pm 5.98}$
	ℓ_1-norm	0.1		92.52 ± 5.20	96.36 ± 3.39	99.78 ± 0.19	40.56 ± 19.58	$\mathbf{61.18 \pm 18.90}$
	ℓ_2-norm	0.3		99.72 ± 0.31	99.48 ± 0.94	99.96 ± 0.08	64.36 ± 17.91	$\mathbf{82.26 \pm 11.42}$
TJ-10 (4)	ℓ_∞-norm	0.3	85.07 ± 7.41	83.43 ± 8.27	78.25 ± 7.69	84.84 ± 8.12	15.29 ± 10.95	$\mathbf{53.71 \pm 18.45}$
	ℓ_1-norm	0.1		35.52 ± 16.3	41.66 ± 7.06	79.04 ± 8.14	3.80 ± 3.50	$\mathbf{12.20 \pm 13.40}$
	ℓ_2-norm	0.3		83.23 ± 7.52	75.09 ± 8.47	83.60 ± 8.00	9.93 ± 8.97	$\mathbf{37.39 \pm 23.66}$

TABLE V
RESULTS OF ALL ATTACK METHODS ON NDQ. EACH ATTACK METHOD IS REPORTED WITH MEAN ± STANDARD DEVIATION, WHICH IS SUMMARIZED OVER FIVE RANDOM RUNS. 'W/O' DENOTES 'WITHOUT' AND $c = \|m\|_\infty$.

Task (b)	Norm Type	ϵ	Win-Rate (%)					
			w/o Attack	RA	OA	MA	WA	VSAA
PP-3 (1)	ℓ_∞-norm	$0.5c$	95.71 ± 1.92	79.35 ± 4.85	93.61 ± 4.25	87.92 ± 2.76	44.90 ± 4.10	$\mathbf{76.85 \pm 4.2}$
	ℓ_1-norm	$0.1c$		93.25 ± 3.05	95.02 ± 2.51	$\mathbf{90.69 \pm 3.89}$	90.52 ± 3.53	93.75 ± 2.67
	ℓ_2-norm	$0.5c$		78.65 ± 4.88	88.81 ± 4.47	85.15 ± 4.11	43.15 ± 5.4	$\mathbf{64.86 \pm 7.66}$
PP-7 (3)	ℓ_∞-norm	$0.1c$	71.67 ± 6.48	69.65 ± 3.95	74.24 ± 5.77	64.95 ± 2.39	62.64 ± 3.36	$\mathbf{64.65 \pm 5.22}$
	ℓ_1-norm	$0.1c$		69.24 ± 4.94	72.43 ± 4.30	64.55 ± 3.51	61.18 ± 6.81	$\mathbf{62.50 \pm 6.39}$
	ℓ_2-norm	$0.1c$		69.93 ± 2.31	74.44 ± 4.92	$\mathbf{65.35 \pm 1.08}$	60.90 ± 4.47	66.53 ± 5.03
PP-10 (4)	ℓ_∞-norm	$0.1c$	72.01 ± 6.30	72.08 ± 4.55	74.44 ± 5.98	70.89 ± 2.39	64.31 ± 3.08	$\mathbf{68.61 \pm 5.87}$
	ℓ_1-norm	$0.1c$		70.62 ± 6.28	72.57 ± 6.58	70.30 ± 5.77	44.17 ± 4.67	$\mathbf{62.78 \pm 9.07}$
	ℓ_2-norm	$0.1c$		72.15 ± 5.05	76.88 ± 5.17	70.69 ± 4.50	60.21 ± 4.96	$\mathbf{65.97 \pm 7.22}$
TJ-5 (2)	ℓ_∞-norm	$2c$	98.26 ± 2.63	82.50 ± 13.68	94.03 ± 3.60	86.73 ± 11.28	42.5 ± 30.87	$\mathbf{85.42 \pm 9.9}$
	ℓ_1-norm	c		91.04 ± 7.00	89.79 ± 7.97	90.10 ± 9.80	48.75 ± 35.16	$\mathbf{87.85 \pm 9.25}$
	ℓ_2-norm	$2c$		81.94 ± 11.84	85.49 ± 9.99	83.96 ± 15.36	47.15 ± 29.20	$\mathbf{81.94 \pm 11.84}$
TJ-10 (4)	ℓ_∞-norm	$2c$	99.10 ± 0.47	87.43 ± 6.60	97.78 ± 0.56	86.73 ± 11.28	20.35 ± 16.13	$\mathbf{85.07 \pm 7.28}$
	ℓ_1-norm	c		90.97 ± 5.27	93.75 ± 3.20	90.10 ± 9.80	29.17 ± 18.17	$\mathbf{88.68 \pm 9.02}$
	ℓ_2-norm	$2c$		84.10 ± 7.30	89.24 ± 4.88	83.96 ± 15.36	47.71 ± 31.72	$\mathbf{83.68 \pm 11.78}$



(a) $b = 1$ (b) $b = 2$

Fig. 8. Results of VSAA on the TJ-5 task when malicious agents and regular agents are deployed by different models separately. These models are trained by CommNet with different seeds separately. ϵ_∞ is set to 0.3. The number of malicious agents is chosen from $\{1, 2\}$. (a) $b = 1$. (b) $b = 2$. Here, 'w/o A' denotes 'without attack'.

finding illustrates the effectiveness of VSAA as an attack method when malicious agents and regular agents only share the model architecture. This finding also demonstrates that VSAA is not restricted by parameter sharing.

2) The Influence of Communication Dependency: Intuitively, the effect of VSAA might be more significant when agents are more dependent on the communication message. To illustrate the influence of communication dependency on

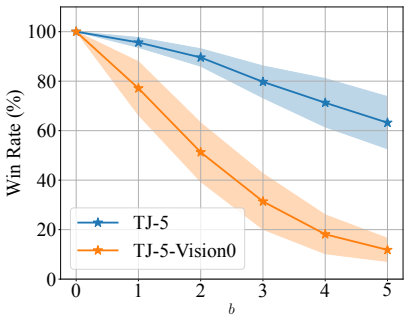


Fig. 9. Results of VSAA on the TJ-5 and TJ-5-Vision0 tasks when the number of malicious agents b is chosen from $\{0, 1, 2, 3, 4, 5\}$. The x-axis is the number of malicious agents b.

VSAA, we reduce the agent's vision range to 0 in the TJ-5 task, which is denoted as TJ-5-Vision0. Please note that the difference between TJ-5 and TJ-5-Vision0 is only the vision range of agents. A larger vision range implies that agents are less reliant on communication information, reducing their dependence on communication messages. When the assistance supplied by communication messages for agents is small, the influence of adversarial communication messages for agents might also be limited. For example, when agents have a global view, communication messages are unnecessary for agents, and the influence of adversarial communication messages might also be limited. In Figure 9, we compare the results of VSAA on TJ-5 and TJ-5-Vision0 with different b. Here, ϵ_∞ is set to 0.3. We can find that VSAA performs better on TJ-5-Vision0 than on TJ-5. This is because agents in TJ-5-Vision0 exhibit a higher dependency on communication messages. This observation highlights that the effect of VSAA is more significant when the agent relies more on communication.

E. Sensitivity Analysis of Hyper-parameters

In our method, the hyper-parameters include the number of malicious agents and the magnitude of perturbations under different norms. In this section, we still utilize CommNet on the TJ-5 task as our test environment and focus on ℓ_∞-norm except when investigating the magnitude of perturbations.

1) Number of Malicious Agents: Figure 9 summarizes the results of CommNet under our method, VSAA, with varying b. Here, b is chosen from $\{0, 1, 2, 3, 4, 5\}$ and ϵ_∞ is fixed at 0.3. We can find that, with the increase in the number of malicious agents b, the performance of CommNet decreases. This is intuitively reasonable because more attackers can increase the strength of VSAA.

2) Magnitude of Perturbations: ϵ_∞, ϵ_1 and ϵ_2 represent the perturbation magnitude of communication messages under ℓ_∞-norm, ℓ_1-norm and ℓ_2-norm, respectively. When $\epsilon = 0.0$, there are no perturbed communication messages. Figure 10 shows the results of VSAA with different magnitudes of perturbations. Here, ϵ is chosen from $\{0.0, 0.01, 0.05, 0.1, 0.5, 1.0, 2.0\}$ and b is fixed to 2. As we expected, a larger perturbation magnitude of communication messages will increase the strength of VSAA.

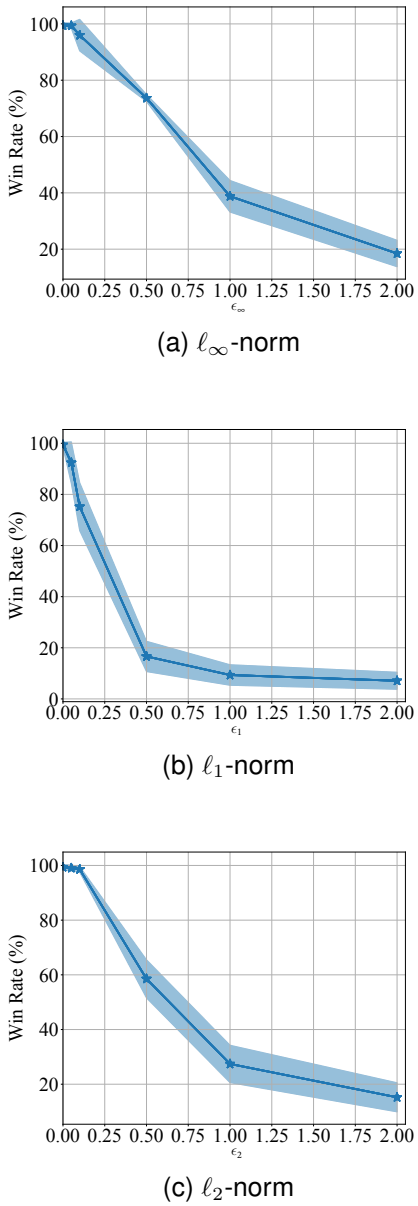


Fig. 10. Results of VSAA on the TJ-5 task when the magnitude of perturbations under different norms varies. ϵ_∞, ϵ_1 and ϵ_2 are chosen from $\{0.0, 0.01, 0.05, 0.1, 0.5, 1.0, 2.0\}$ and $b = 2$. (a) Under ℓ_∞-norm. The x-axis is the value of ϵ_∞. (b) Under ℓ_1-norm. The x-axis is the value of ϵ_1. (c) Under ℓ_2-norm. The x-axis is the value of ϵ_2.

F. Results under Defense

As discussed in Section II-C, AME [36] is a feasible defense method under grey-box attacks. Hence, to further illustrate the effectiveness of VSAA, we evaluate all attack methods on the TJ-5 task under AME. Here, for a fair comparison, we set all attack methods with $\epsilon = 1.0$ and $b = \lceil \frac{n}{3} \rceil$. In Figure 11, we can find that even under AME, VSAA is still the most effective attack method on communication among all attack methods.

VI. Conclusion

In this paper, we have proposed a novel and effective grey-box attack method, called VSAA, to make regular agents take

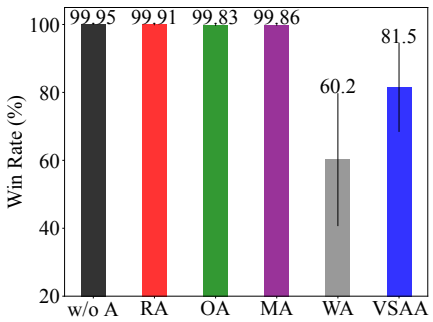


Fig. 11. Results of all attack methods on the TJ-5 task under AME defense mechanism.

sub-optimal actions by sending adversarial communication messages. VSAA is the first grey-box attack method on communication in CMARL. Experimental results on multiple tasks show that VSAA can effectively degrade the performance of MAS. The findings of this paper will make researchers aware of grey-box attacks in CMARL.

Furthermore, we also discuss the potential defense mechanisms against the attack proposed in this paper. Introducing the MR defense mechanism at a higher level in each agent might be a possible defense against the proposed attack by preventing malicious agents from knowing whether the message has been reconstructed. This high-level defense architecture can enhance the stealth of the system and further improve the ability to resist adversarial attacks. Furthermore, reducing the similarity of different agents, especially the similarity of the architecture of the model in different agents in MASs, might improve the ability to resist the proposed attack.

ACKNOWLEDGEMENT

This work is supported by the National Key R&D Program of China (No. 2020YFA0713900), NSFC Project (No. 62192783), and Fundamental Research Funds for the Central Universities (No.020214380108).

REFERENCES

[1] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *AAMAS*, 2018.

[2] T. Wang, J. Wang, C. Zheng, and C. Zhang, "Learning nearly decomposable value functions via communication minimization," in *ICLR*, 2020.

[3] X. Ma and W.-J. Li, "State-based episodic memory for multi-agent reinforcement learning," *Machine Learning*, vol. 112, no. 12, pp. 5163–5190, 2023.

[4] S. Gu, J. G. Kuba, Y. Chen, Y. Du, L. Yang, A. C. Knoll, and Y. Yang, "Safe multi-agent reinforcement learning for multi-robot control," *Artificial Intelligence*, vol. 319, p. 103905, 2023.

[5] D. L. Leottau, J. Ruiz-del-Solar, and R. Babuska, "Decentralized reinforcement learning of robot behaviors," *Artificial Intelligence*, vol. 256, pp. 130–159, 2018.

[6] C. Huang, G. Chen, and K. Wong, "Multi-agent reinforcement learning-based buffer-aided relay selection in irs-assisted secure cooperative networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4101–4112, 2021.

[7] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 427–438, 2013.

[8] L. Matignon, L. Jeanpierre, and A. Mouaddib, "Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes," in *AAAI*, 2012.

[9] F. Wu, S. Zilberstein, and X. Chen, "Online planning for multi-agent systems with bounded communication," *Artificial Intelligence*, vol. 175, no. 2, pp. 487–511, 2011.

[10] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *NeurIPS*, 2016.

[11] T. Chu, S. Chinchali, and S. Katti, "Multi-agent reinforcement learning for networked system control," in *ICLR*, 2020.

[12] A. Nedic and A. E. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[13] K. Zhang, Z. Yang, and T. Basar, "Networked multi-agent reinforcement learning in continuous spaces," in *CDC*, 2018.

[14] Y. Hoshen, "VAIN: attentional multi-agent predictive modeling," in *NeurIPS*, 2017.

[15] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted multi-agent communication," in *ICML*, 2019.

[16] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *NeurIPS*, 2016.

[17] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *NeurIPS*, 2018.

[18] W. Kim, J. Park, and Y. Sung, "Communication in multi-agent reinforcement learning: Intention sharing," in *ICLR*, 2021.

[19] Z. Ding, T. Huang, and Z. Lu, "Learning individually inferred communication for multi-agent cooperation," in *NeurIPS*, 2020.

[20] A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," in *ICLR*, 2019.

[21] S. Q. Zhang, Q. Zhang, and J. Lin, "Succinct and robust multi-agent communication with temporal message control," in *NeurIPS*, 2020.

[22] T. Yuan, H. Chung, J. Yuan, and X. Fu, "DACOM: learning delay-aware communication for multi-agent reinforcement learning," in *AAAI*, 2023.

[23] W. Xue, W. Qiu, B. An, Z. Rabinovich, S. Obraztsova, and C. K. Yeo, "Mis-spoke or mis-lead: Achieving robustness in multi-agent communicative reinforcement learning," in *AAMAS*, 2022.

[24] H. Wang, S. Wang, Z. Jin, Y. Wang, C. Chen, and M. Tistarelli, "Similarity-based gray-box adversarial attack against deep face recognition," *CoRR*, vol. abs/2201.04011, 2022.

[25] S. Z. Béguelin, S. Tople, A. Paverd, and B. Köpf, "Grey-box extraction of natural language models," in *ICML*, 2021.

[26] F. Ataiefard and H. Hemmati, "Gray-box adversarial attack of deep reinforcement learning-based trading agents*," in *ICMLA*, 2023.

[27] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *CoRR*, vol. abs/1605.07277, 2016.

[28] S. H. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," in *ICLR Workshop Track Proceedings*, 2017.

[29] X. Ma and W.-J. Li, "Grey-box adversarial attack on communication in multi-agent reinforcement learning," in *AAMAS*, 2023.

[30] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 895–943, 2022.

[31] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, and Y. Ni, "Learning agent communication under limited bandwidth by message pruning," in *AAAI*, 2020.

[32] J. Blumenkamp and A. Prorok, "The emergence of adversarial communication in multi-agent reinforcement learning," in *CoRL*, 2020.

[33] R. Mitchell, J. Blumenkamp, and A. Prorok, "Gaussian process based message filtering for robust multi-agent cooperation in the presence of adversarial communication," *CoRR*, vol. abs/2012.00508, 2020.

[34] N. Piazza and V. Behzadan, "A theory of mind approach as test-time mitigation against emergent adversarial communication," in *AAMAS*, 2023.

[35] Z. Lv, L. Xiao, Y. Chen, H. Chen, and X. Ji, "Safe multi-agent reinforcement learning for wireless applications against adversarial communications," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 6824–6839, 2024.

[36] Y. Sun, R. Zheng, P. Hassanzadeh, Y. Liang, S. Feizi, S. Ganesh, and F. Huang, "Certifiably robust policy learning against adversarial multi-agent communication," in *ICLR*, 2023.
[37] F. A. Oliehoek, "Decentralized pomdps," in *Reinforcement Learning*, ser. Adaptation, Learning, and Optimization. Springer, 2012, vol. 12, pp. 471–503.
[38] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *AsiaCCS*, 2017.
[39] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.
[40] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.
[41] Y.-R. Yang and W.-J. Li, "Buffered asynchronous SGD for byzantine learning," *Journal Machine Learning Research*, vol. 24, pp. 204:1–204:62, 2023.
[42] Y. Wang and S. Zou, "Policy gradient method for robust reinforcement learning," in *ICML*, 2022.
[43] J. Mei, C. Xiao, C. Szepesvári, and D. Schuurmans, "On the global convergence rates of softmax policy gradient methods," in *ICML*, 2020.
[44] G. Li, Y. Wei, Y. Chi, Y. Gu, and Y. Chen, "Softmax policy gradient methods can take exponential time to converge," in *COLT*, 2021.
[45] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *ICML*, 2019.
[46] B. Neyshabur, "Implicit regularization in deep learning," *CoRR*, vol. abs/1709.01953, 2017.
[47] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.
[48] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.
[49] X. Li, S. Ji, M. Han, J. Ji, Z. Ren, Y. Liu, and C. Wu, "Adversarial examples versus cloud-based detectors: A black-box empirical study," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, pp. 1933–1949, 2021.
[50] Y. Mao, C. Fu, S. Wang, S. Ji, X. Zhang, Z. Liu, J. Zhou, A. X. Liu, R. Beyah, and T. Wang, "Transfer attacks revisited: A large-scale empirical study in real computer vision settings," in *S&P*, 2022.
[51] Y. Zhong and W. Deng, "Towards transferable adversarial attack against deep face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1452–1466, 2021.
[52] Z. Li, W. Wang, J. Li, K. Chen, and S. Zhang, "UCG: A universal cross-domain generator for transferable adversarial examples," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3023–3037, 2024.
[53] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C. Hung, P. H. S. Torr, J. N. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," in *AAMAS*, 2019.
[54] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning," in *ICML*, 2018.
[55] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, 2009.
[56] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

Xiao Ma received the BSc degree in information and computing science from Xi'an Jiao Tong University, China. She is currently working toward the PhD degree in the Department of Computer Science and Technology, Nanjing University of China. Her research interests are in reinforcement learning, machine learning, and artificial intelligence. She is a student member of the IEEE.

Wu-Jun Li received the BSc and MEng degrees in computer science from the Nanjing University of China, and the PhD degree in computer science from the Hong Kong University of Science and Technology. He started his academic career as an assistant professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He then joined Nanjing University, where he is currently a professor in the Department of Computer Science and Technology. His research interests are in machine learning, big data, and artificial intelligence. He is a member of the IEEE.