ABSTRACT
Recent studies in multi-agent communicative reinforcement learning (MACRL) have demonstrated that multi-agent coordination can be greatly improved by allowing communication between agents. Meanwhile, adversarial machine learning (ML) has shown that ML models are vulnerable to attacks. Despite the increasing concern about the robustness of ML algorithms, how to achieve robust communication in multi-agent reinforcement learning has been largely neglected. In this paper, we systematically explore the problem of adversarial communication in MACRL. Our main contributions are threefold. First, we propose an effective method to perform attacks in MACRL by learning a model to generate optimal malicious messages. Second, we develop a defence method based on message reconstruction, to maintain multi-agent coordination under message attacks. Third, we formulate the adversarial communication problem as a two-player zero-sum game and propose a game-theoretical method \( \Phi \)-MACRL to improve the worst-case defending performance. Empirical results demonstrate that many state-of-the-art MACRL methods are vulnerable to message attacks, and our method can significantly improve their robustness.

KEYWORDS
Multi-Agent Reinforcement Learning; Robust Reinforcement Learning; Adversarial Reinforcement Learning

1 INTRODUCTION
Cooperative multi-agent reinforcement learning (MARL) has achieved remarkable success in a variety of challenging problems, such as urban systems [29], coordination of robot swarms [11] and real-time strategy video games [36]. To tackle the problems of scalability and non-stationarity in MARL, the framework of centralized training with decentralized execution (CTDE) is proposed [17, 24], where decentralized policies are learned in a centralized manner so that they can share experiences, parameters, etc., during training. Despite the advantages, CTDE-based methods still perform unsatisfactorily in scenarios where multi-agent coordination is necessary, mainly due to partial observability in decentralized execution. To mitigate partial observability, many multi-agent communicative reinforcement learning (MACRL) methods have been proposed, which allow agents to exchange information such as private observations, intentions during the execution phase. MACRL greatly improves multi-agent coordination in a wide range of tasks [6, 7, 13, 15, 30, 37, 38].

Meanwhile, adversarial machine learning has received extensive attention [2, 10]. Adversarial machine learning demonstrates that machine learning (ML) models are vulnerable to manipulation [8, 9]. As a result, ML models often suffer from performance degradation when under attack. For the same reason, many practical applications of ML models are at high risk. For instance, researchers have shown that, by placing a few small stickers on the ground at an intersection, self-driving cars can be tricked into making abnormal judgements and driving into the opposite lane [32]. Maliciously designed attacks on ML models can have serious consequences.

Unfortunately, despite great importance, adversarial problems, especially adversarial inter-agent communication problems, remain largely uninvestigated in MACRL. Blumenkamp et al. show that, by introducing random noise in communication, agents are able to deceive their opponents in competitive games [3]. However, the attacks are not artificially designed and therefore inefficient. Besides, cooperative cases, where communication is more crucial, are neglected. They also fail to propose an effective defence, but merely retrain the models to adapt to the attacks. Mitchell et al. propose to generate weights of Attention models [35] through Gaussian process for defending against random attacks in attention-based MACRL [21]. However, the applicability of this approach is unsatisfactory, being limited to attention-based MACRL, and its performance on maliciously designed attacks is unclear.

In this paper, we systematically explore the problem of adversarial communication in MACRL, where there are malicious agents that attempt to disrupt multi-agent cooperation by manipulating messages. Our contributions are in three aspects. First, we propose an effective learning approach to model the optimal attacking scheme in MACRL. Second, to defend against the adversary, we propose a two-stage message filter which works by first detecting...
the malicious message and then recovering the message. The defending scheme can greatly maintain the coordination of agents under message attacks. Third, to address the problem that the message filter can be exploited by learnable attackers, we formulate the attack-defense problem as a two-player zero-sum game and propose \( \mathcal{R}\)-MACRL, based on a game-theoretical framework Policy-Space Response Oracle (PSRO) \([18, 22]\), to approximate a Nash equilibrium policy in the adversarial communication game. \( \mathcal{R}\) -MACRL improves the defensive performance under the worst case and thus improves the robustness. Empirical experiments demonstrate that many state-of-the-art MACRL algorithms are vulnerable to attacks and our method can significantly improve their robustness.

## 2 PRELIMINARIES AND RELATED WORK

**Multi-Agent Communicative Reinforcement Learning.** There has been extensive research on encouraging communication between agents to improve performance on cooperative or competitive tasks. Among the recent advances, some design communication mechanisms to address the problem of when to communicate \([13, 14, 28]\); other lines of work, e.g., TarMac \([6] \), focus on who to communicate. These works determine the two fundamental elements in communication, i.e., the message sender and the receiver. Apart from the two elements, the message itself is another element which is crucial in communication, i.e., what to communicate: Jaques et al. propose to maximize the social influence of messages \([12]\). Kim et al. encode messages such that they contain the intention of agents \([15]\). Some other works learn to send succinct messages to meet the limitations of communication bandwidth \([37, 38, 42]\). Despite significant progress in MACRL, if some agents are adversarial and send maliciously designed messages, multi-agent coordination will rapidly disintegrate as these messages propagate.

**Adversarial Training.** Adversarial training is a prevalent paradigm for training robust models to defend against potential attacks \([9, 31]\). Recent literature has considered two types of attacks \([5, 25, 33]\): black-box attack and white-box attack. In black-box attack, the attacker does not have access to information about the attacked deep neural network (DNN) model; whereas in white-box attack, the attacker tries to attack a DNN by corrupting the input via \( \| \cdot \|_p - \text{norm} \) \((s.t. \min \| x + \delta \|_p) \) attack \([9]\). The attacker has complete knowledge, e.g., the architecture, the parameters and potential defense mechanisms, about the DNN model. We consider the black-box attack in our problem formulation, because the setting of the white-box attack is too idealistic and may not be applicable to many realistic adversarial scenarios. In adversarial training, the attacker tries to attack a DNN by corrupting the input via \( \ell_p \)-norm \((p \in \{1, 2, \infty\}) \) attack \([9]\). The attacker carefully generates artificial perturbations to manipulate the input of the model. In doing so, the DNN will be fooled into making incorrect predictions or decisions. The attacker finds the optimal perturbation \( \delta \) by optimizing:

\[
\max \delta \mathcal{L}_{\text{predict}}(f(x), f(x + \delta)) \quad \text{s.t.} \quad \min \mathcal{L}_{\text{norm}}(x, x + \delta)
\]

where \( x \) is the input, \( f \) is the DNN model, \( \mathcal{L}_{\text{predict}} \) is a metric to measure the distance between the outputs of the DNN model w/ and w/o being attacked, \( \mathcal{L}_{\text{norm}} \) is used to measure that for the inputs.

**Adversarial Reinforcement Learning (RL).** Recent advances in adversarial machine learning motivate researchers to investigate the adversarial problem in RL \([8, 19, 39]\), SA-MDP \([41]\) characterizes the problem of decision making under adversarial attacks on state observations. Lin et al. propose two tactics of attacks, i.e., the strategically-timed attack and the enchanting attack, which attack by injecting noise to states and luring the agent to a designated target \([19]\). Gleave et al. consider the problem of taking adversarial actions that change the environment and consequentially change the observation of agents \([8]\). ATLA \([40]\) propose to train the optimal adversary to perturb state observations and improve the worst-case agent reward. The settings of these works are different from ours: we consider the multi-agent scenario and restrict the attacking approach to adversarial messages, which makes the detection of anomalies difficult. Tu et al. propose to attack on multi-agent communication \([34]\). However, their focus is on the representation-level, whereas we focus on the policy-level. Recently, there are some works considering a similar setting as ours \([3, 21]\). However, they either focus on random attacks in specific competitive games or the defense of specific communication methods.

## 3 ACHIEVING ROBUSTNESS IN MACRL

In this section, we propose our method for achieving robustness in MACRL. We begin by proposing the problem formulation for adversarial communication in MACRL. Then, we introduce the method to build the optimal attacking scheme in MACRL. Next, we propose a learnable two-stage message filter to defend against possible attacks. Finally, we propose to formulate the attack-defense problem as a two-player zero-sum game, and design a game-theoretic method \( \mathcal{R}\)-MACRL to approximate a Nash equilibrium policy for the defender. In this way, we can improve the worst-case performance of the defender and thus enhance robustness.

### 3.1 Problem Formulation: Adversarial Communication in MACRL

An MACRL problem can be modeled by Decentralised Partially Observable Markov Decision Process with Communication (Dec-POMDP-Com) \([23]\), which is defined by a tuple \((\mathcal{S}, \mathcal{M}, \mathcal{A}, \mathcal{P}, R, Y, O, C, N, \gamma)\). \( \mathcal{S} \) denotes the state of the environment and \( \mathcal{M} \) is the message space. Each agent \( i \in N := \{1, ..., N\} \) chooses an action \( a_i \in \mathcal{A} \) at a state \( s \in \mathcal{S} \), giving rise to a joint action vector, \( a := [a_i]_{i=1}^N \in \mathcal{A}^N \). \( \mathcal{P}(s'| s, a) : \mathcal{S} \times \mathcal{A}^N \times \mathcal{S} \mapsto \mathcal{P}(S) \) is a Markovian transition function. Every agent shares the same joint reward function \( R(s, a) : \mathcal{S} \times \mathcal{A}^N \mapsto \mathbb{R} \) and \( \gamma \in (0, 1) \) is the discount factor. Due to partial observability, each agent has individual partial observation \( v \in \mathcal{Y} \), according to the observation function \( O(s, i) : \mathcal{S} \times N \mapsto \mathcal{Y} \). Each agent holds two policies, i.e., action policy \( p_t(a_i | r_i, m_{t}^{in}) : T \times \mathcal{M} \mapsto [0, 1] \) and message policy \( \nu_t(m_{t}^{out} | r_i, m_{t}^{in}) : T \times \mathcal{M} \times \mathcal{M} \mapsto [0, 1] \), both of which are conditioned on the observation-history \( r_i \in T = (T \times \mathcal{A}) \) and incoming messages \( m_{t}^{in} \) aggregated by the communication protocol \( C(m_{t}^{in}, m_{t}^{out}, i) : \mathcal{M}^{|N|} \times \mathcal{N} \times \mathcal{M} \mapsto [0, 1] \).

In adversarial communication where there are \( N_{adv} \) malicious agents, we assume that each malicious agent holds the third private adversarial policy, \( \xi(\delta_{adv}^{t} | r_i, m_{t}^{in}, m_{t}^{out}) : T \times \mathcal{M} \times \mathcal{M} \times \mathcal{M} \mapsto [0, 1] \), which generates adversarial message \( \delta_{adv}^{t} \) based on its action-observation history \( r_i \), received messages \( m_{t}^{in} \) and the message
**Attacker** Loss

actions. more properly and reduce the probability of selecting sub-optimal

\[ j \]

contaminated message \( m \) reconstruct the contaminated messages, to avoid distributing the

ule. The defender performs a transformation from \( m \) to \( \tilde{m} \), which is used to reconstruct the contaminated message (\( m \) to \( \tilde{m} \)). The benign agent \( j \) estimates Q-values based on \( \tilde{m} \), which can reduce the probability of selecting sub-optimal actions.

**3.2 Learning the Attacking Scheme**

To model the attack scheme in adversarial communication, we use a deep neural network (DNN) \( f_{\theta_{\text{adv}}} \), parameterized by \( \theta_{\text{adv}} \), to generate adversarial messages for a malicious agent. The adversarial policy \( \xi \) is a multivariate Gaussian distribution whose parameters are determined by the DNN \( f_{\theta_{\text{adv}}} \), i.e., \( \xi \sim \mathcal{N}(f_{\theta_{\text{adv}}}(\tau, m^\text{in}, m^\text{out}; \theta_{\text{adv}}), \Sigma) \) where \( \Sigma \) is a fixed covariance matrix. The reason of using Gaussian distribution as the prior is that it is the maximum entropy distribution under constraints of mean and variance. Each malicious agent generates adversarial messages by sampling from its adversarial policy. The optimization objective of the adversarial policy is to minimize the accumulated team rewards subject to a constraint on the distance between \( m^\text{out} \) and \( m^\text{adv} \). We utilize Proximal Policy Optimization (PPO) [27] to optimize the adversarial policy by maximizing the following objective:

\[
J_{\text{adv}}(\theta_{\text{adv}}) = \mathbb{E}_{\tau,m^\text{in},m^\text{out}} \left[ \min(\rho \cdot A^\text{eff}, \text{clip}(\rho, 1 - \varepsilon) \cdot A^\text{eff}) \right] - \alpha \cdot \mathbb{E}_{m^\text{out},\xi} \left[ (m^\text{out} - m^\text{adv})^2 \right]
\]

**3.3 Defending against Adversarial Communication**

We can train a DNN to model the attack scheme in adversarial communication. However, the defense of that is non-trivial because i) benign agents have no prior knowledge on which agents are malicious; ii) the malicious agents can inconsistently pretend to be cooperative or non-cooperative to avoid being detected; and iii)
Defender

\[ \pi \]

where tags which are estimated similarly as in the attack, between the messages before and after being attacked. To improve the of the ground-truth labels of messages (whether they are malicious).

optima induced by unguided exploration in large message space, eg. We utilize PPO to optimize the strategy. To mitigate the local the joint accumulated rewards under attack, i.e., \( m^{\text{out}} \) denotes the message sent by agent \( i \). We perform sampling from the generated distributions to determine the anomaly messages \( x \sim h_d(\cdot | \tilde{m}^{\text{out}}; \theta_d) \). Here, \( x \) is an indicator that records whether a message is predicted to be malicious or not. The predicted malicious messages \( m \) are recovered by the reconstructor \( g_r(\cdot | \tilde{m}; \theta_r) : M \mapsto M \) separately. The recovered message and other normal messages are aggregated and determine the messages that each agent will receive.

The optimization objective of the message filter is to maximize the joint accumulated rewards under attack, i.e., \( \max_{\pi} \mathbb{E} \left[ \sum_{t=0}^{\infty} y^t r_t(x) \right] \), where \( r_t \) is the team reward after performing the defending strategy. We utilize PPO to optimize the strategy. To mitigate the local optima induced by unguided exploration in large message space, we introduce a regularizer which is optimized under the guidance of the ground-truth labels of messages (whether they are malicious). For the reconstructor, we train it by minimizing the distance between the messages before and after being attacked. To improve the tolerance of the reconstructor to errors made by the detector, apart from malicious messages, we also use benign messages as training data, in which case the reconstructor is an identical mapping. Formally, the two-stage message filter is optimized by maximizing the following function:

\[
J_{\xi}(\theta_d, \theta_r) = \mathbb{E}_{(m,x)} \left[ \sum_{t=1}^{m} \min \left( \rho_1 \cdot A_t^{\text{adv}}, \text{clip} \left( \rho_1, 1 \pm \epsilon \right) \cdot A_t^{\text{adv}} \right) \right] + \beta_1 \cdot \mathbb{E}_{m} \left[ g_r \cdot \log(h_d(\cdot | m; \theta_d)) \right] - \beta_2 \cdot \mathbb{E}_{m} \left[ (\tilde{m} - g_r(\cdot | m; \theta_r))^2 \right] \tag{2}
\]

where \( \rho_1 = h_d(x_t|m_3; \theta_d)/h_d^{\text{old}}(x_t|m_3; \theta_d) \) and \( A_t^{\text{adv}} = \text{max} \{ 0, A_t \} \).

\[ \text{Figure 2: Structure of the communication protocol with the message filter (defender). The anomaly detector and the reconstructor are designed to determine and recover the potential malicious messages, respectively.} \]

recovering useful information from the contaminated messages is difficult. To address these challenges, we design a two-stage message filter for the communication protocol to perform defence. The message filter works by first determining the messages that are likely to be malicious and then recovering the potential malicious messages before distributing them to the corresponding agents. As in Fig. 2, the message filter \( \xi(h_d, g_r) \) consists of an anomaly detector \( h_d \) and a message reconstructor \( g_r \). The anomaly detector, parameterized by \( \theta_d \), outputs the probability that each incoming message needs to be recovered, i.e., \( h_d(\cdot | \tilde{m}^{\text{out}}; \theta_d) : M \mapsto \Psi \), where \( \Psi \) denotes a binomial distribution, \( m^{\text{out}} \) denotes the message sent by agent \( i \). We perform sampling from the generated distributions to determine the anomaly messages \( x \sim h_d(\cdot | \tilde{m}^{\text{out}}; \theta_d) \). Here, \( x \) is an indicator that records whether a message is predicted to be malicious or not. The predicted malicious messages \( m \) are recovered by the reconstructor \( g_r(\cdot | \tilde{m}; \theta_r) : M \mapsto M \) separately. The recovered message and other normal messages are aggregated and determine the messages that each agent will receive.

The optimization objective of the message filter is to maximize the joint accumulated rewards under attack, i.e., \( \max_{\pi} \mathbb{E} \left[ \sum_{t=0}^{\infty} y^t r_t(x) \right] \), where \( r_t \) is the team reward after performing the defending strategy. We utilize PPO to optimize the strategy. To mitigate the local optima induced by unguided exploration in large message space, we introduce a regularizer which is optimized under the guidance of the ground-truth labels of messages (whether they are malicious). For the reconstructor, we train it by minimizing the distance between the messages before and after being attacked. To improve the tolerance of the reconstructor to errors made by the detector, apart from malicious messages, we also use benign messages as training data, in which case the reconstructor is an identical mapping. Formally, the two-stage message filter is optimized by maximizing the following function:

\[
J_{\xi}(\theta_d, \theta_r) = \mathbb{E}_{(m,x)} \left[ \sum_{t=1}^{m} \min \left( \rho_1 \cdot A_t^{\text{adv}}, \text{clip} \left( \rho_1, 1 \pm \epsilon \right) \cdot A_t^{\text{adv}} \right) \right] + \beta_1 \cdot \mathbb{E}_{m} \left[ g_r \cdot \log(h_d(\cdot | m; \theta_d)) \right] - \beta_2 \cdot \mathbb{E}_{m} \left[ (\tilde{m} - g_r(\cdot | m; \theta_r))^2 \right] \tag{2}
\]

where \( \rho_1 = h_d(x_t|m_3; \theta_d)/h_d^{\text{old}}(x_t|m_3; \theta_d) \) and \( A_t^{\text{adv}} = \text{max} \{ 0, A_t \} \).

\[ \text{Figure 3: Workflow of \( \mathcal{R} \)-MACRL.} \]

ground-truth labels of messages, \( m \) is the messages that have not been attacked. \( \beta_1, \beta_2 \) and \( \epsilon \) are hyperparameters.

3.4 Achieving Robust MACRL

Despite the defensive message filter, the effectiveness of the defence system can rapidly decrease if malicious agents are aware of the defender and adjust their attack schemes. To mitigate this, we formulate the attack-defence problem as a two-player zero-sum game (malicious agents are controlled by the attacker) and improve the performance of the defender in the worst case. We define the adversarial communication game by a tuple \( (\Pi, U) \), where \( \Pi = (\Pi_x, \Pi_z) \) is the joint policy space of the players (\( \Pi_x \) and \( \Pi_z \) denote the policy space of the defender and the attacker respectively), \( U : \Pi \mapsto \mathbb{R}^2 \) is the utility function which is used to calculate utilities for the attacker and the defender given their joint policy \( \pi = (\pi_x, \pi_z) \in \Pi \). The utility of the defender is defined as the expected team return. The adversarial communication game is zero-sum, i.e., the utility of the defender \( U_{\xi}(\pi) \) must be the negative of the utility of the attacker \( U_{\xi}(\pi) \) for \( \forall \pi \in \Pi \). The solution concept of the adversarial communication game is Nash equilibrium (NE) and we can approach an NE by optimizing the following MaxMin objective:

\[
\max_{\pi_x, \pi_z} \min_{\pi_x, \pi_z} \mathbb{E} \left[ \sum_{t=0}^{\infty} r_t(x_t, \pi_t) \right] \tag{3}
\]

where \( Br(\pi_x) \) is the best response policy of the defender, i.e., \( Br(\pi_x) = \arg \min_{\pi_z} U_{\xi}(\pi_x, \pi_z) \).

We propose \( \mathcal{R} \)-MACRL to optimize the objective in the adversarial communication game. \( \mathcal{R} \)-MACRL is developed based on Policy-Space Response Oracle (PSRO) [18], a deep learning-based extension of the classic Double Oracle algorithm [20]. The workflow of \( \mathcal{R} \)-MACRL is depicted in Fig. 3. At each iteration, \( \mathcal{R} \)-MACRL keeps a population (set) of policies \( \Pi \subseteq \Pi \), e.g., \( \Pi_x = (\pi_{11}, \pi_{12}) \) and \( \Pi_z = (\pi_{21}, \pi_{22}) \). We can evaluate the utility table \( U(\Pi') \) for the current iteration and calculate a Nash equilibrium \( \pi_x^* = (\Delta(\Pi_x'), \Delta(\Pi_z')) \) for the game restricted to policies in \( \Pi' \) by, for example, linear programming, replicator dynamics, etc. \( \Delta \) denotes the meta-strategy which is an arbitrary categorical distribution. Next, we calculate the best response policy \( Br(\pi_x^*) \) to the NE in this restricted game for each player \( i \) (player \( \sim i \) denotes the opponent of player \( i \), \( i \in \{x, z\} \)), e.g., \( \pi_{13} = Br(\pi_x^*|_{z}) \) and \( \pi_{23} = Br(\pi_x^*|_{x}) \). We extend the population by adding the best response policies to the policy set:
Algorithm 1: Ψ-MACRL

1 Input: initial policy sets for both players $Π^0$, maximum number of iterations $T$;
2 Empirically estimate utilities $U(Π^0)$ for each joint policy $π ∈ Π^0$;
3 Initialize mixed strategies for both players $π^t_{i,j} = \text{UNIFORM}(Π^0_i)$ for $i ∈ \{ξ, \xi\}$;
4 Initialize number of iterations $t = 0$;
5 while not converge and $t ≤ T$ do
6     for $player \ i ∈ \{ξ, ξ\}$ do
7         Train $π^{t+1}_i$ by letting it exploit $π^{t}_{-i}$;
8         Extend policy set $Π^{t+1}_i = Π^t_i ∪ \{π^{t+1}_{i,-i}\}$;
9         Check convergence and $t = t + 1$;
10     Compute mixed strategies $π^{t+1}_i$ by solving the matrix game defined by $U(Π^{t+1})$;
11 Output: mixed strategies $π_i$ for both players .

$Π^{t+1}_i = Π^t_i ∪ \text{Br}(π^{t}_{-i})$ for $i ∈ \{ξ, ξ\}$. After extending the population, we complete the utility table $U(Π^{t+1})$ and perform the next iteration. The algorithm converges if the best response policies are already in the population. Practically, Ψ-MACRL approximates the utility function $U(\cdot)$ by having each policy in the population $Π^t_i$ playing with each other policy in $Π^{t-1}_j$ and tracking the average utilities. The approximation of the best response policies is performed by maximizing $J_ξ(θ_i|π^t_{-i})$ and $J_ξ(θ_d, d|π^t_{-d})$ for the attacker and the defender, where the full expressions of the two objectives are described in Eq. 1 and Eq. 2, respectively.

The overall algorithm of Ψ-MACRL is presented in Algorithm 1. We first initialize the meta-game and the mixed strategies (line 2 and line 3). Then, at each step, we train the attacker and the defender alternately by letting them best respond to their corresponding opponent (line 7). Next, the learned new policy $π^{t+1}_i$ is added to the policy set $Π^{t+1}_i$ for the two players respectively (line 8). After extending the policy set, we can check the convergence (line 9) and calculate the missing entries in $U(Π^{t+1})$ (line 10). Finally, we solve the new meta-game (line 11) and repeat the iteration.

4 EXPERIMENTS

We conduct extensive experiments on a variety of state-of-the-art MACRL algorithms to answer: Q1: Are MACRL methods vulnerable to message attacks and whether the two-stage message filter is able to consistently recover multi-agent coordination? Q2: Whether Ψ-MACRL is able to improve the robustness of MACRL algorithms under message attacks? Q3: Whether our method is able to scale to scenarios where there are multiple attackers? Q4: Which components contribute to the performance of the method and how does the proposed method work? We first categorize existing MACRL algorithms and then select representative algorithms to perform the evaluation. All experiments are conducted on a server with 8 NVIDIA Tesla V100 GPUs and a 44-core 2.20GHz Intel(R) Xeon(R) E5-2699 v4 CPU.

Table 1: The chosen algorithms and environments.

<table>
<thead>
<tr>
<th>Categories</th>
<th>Methods</th>
<th>Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>NDQ [38]</td>
<td>StarCraft II (SCII) [26]</td>
</tr>
</tbody>
</table>

4.1 Experimental Setting

MACRL methods are commonly categorized by whether they are implicit or explicit [1, 23]. In implicit communication, the actions of agents influence the observations of the other agents. Whereas in explicit communication, agents have a separate set of communication actions and exchange information via communication actions. In this paper, we focus on explicit communication because in implicit communication, to carry out an attack, the attacker’s behaviour is often more subtle, making the attack trivial and easily detectable. We consider the following three realistic types of explicit communication:

- **Communication with delay (CD):** Communication in real world is usually not real-time. We can model this by assuming that it takes one or a few time steps for the messages being received by the targeted agents [30].
- **Local communication (LC):** Messages sometimes cannot be broadcast to all agents due to communication distance limitations or privacy concerns. Therefore, agents need to learn to communicate locally, affecting only those agents within the same communication group [4, 6].
- **Communication with cost or limited bandwidth (CC):** Agents should avoid sending redundant messages because communication in real world is costly and communication channels often have a limited bandwidth [38, 42].

Following the above taxonomy, we select some representative state-of-the-art algorithms to perform evaluation. As in Table 1, we select CommNet [30], TarMAC [6] and NDQ [38] for CD, LC and CC respectively. A brief introduction to each of the chosen algorithms is provided in Appendix A. We evaluate in the following environments, which are similar to those in the paper that first introduced the algorithms:

**Predator Prey (PP).** There are 3 predators, trying to catch 6 prey in a 7 × 7 grid. Each predator has local observation of a 3 × 3 sub-grid centered around it and can move in one of the 4 compass directions, remain still, or perform catch action. The prey moves randomly and is caught if at least two nearby predators try to catch it simultaneously. The predator will obtain a team reward at 10 for each successful catch and will be punished $p$ for each losing catch action. There are two tasks with $p = 0$ and $p = -0.5$, respectively. A prey will be removed from the grid after it being caught. An episode ends after 60 steps or all preys have been caught.

**Traffic Junction (TJ).** In TJ, there are $N_{\text{max}}$ cars and the aim of each car is to complete its pre-assigned route. Each car can observe a 3 × 3 region around it, but is free to communicate with other cars. The action space for each car at every time step is {gas, brake}. The reward function is $r = -0.01t + r_{\text{collision}}$, where $t$ is the number of time steps since the activation of the car, and $r_{\text{collision}} = -10$ is a
We first evaluate the performance of our attacking method on the where the enemy units are controlled by StarCraft II built-in AI (difficulty level is set to hard), and each of the ally units is controlled by a learning agent. The units of the two groups can be asymmetric. The action space contains no-op, move[direction], attack[enemy id], and stop. Agents receive a globally shared reward which calculates the total damage made to the enemy units at each time step. We conduct experiments on four SMAC tasks: 3bane_vs_hM, 4bane_vs_hM, 1o_2r_vs_4r and 1o_3r_vs_4r. More details about the tasks are in Appendix B.

For each of the tasks, we first train the chosen MACRL methods to obtain the action policy and the message policy for each agent. After that, we randomly select an agent to be malicious and train its adversarial policy to examine whether MACRL methods are vulnerable to message attacks. Then we perform defence by training the message filter, during which the adversarial policy of the malicious agent is fixed but keeps working. To show that a single message filter is brittle and can be easily exploited if the attacker adapts to it, we freeze the learned message filter and retrain the adversarial policy. Finally, we integrate the message filter into the framework of $\mathcal{R}$-MACRL and justify if $\mathcal{R}$-MACRL is helpful to improve the robustness. All experiments are carried out with five random seeds and results are shown with a 95% confidence interval.

### 4.2 Recovering Multi-Agent Coordination

We first evaluate the performance of our attacking method on the three selected MACRL algorithms. Then we try to recover multi-agent coordination for the attacked algorithms by applying the message filter.

**CommNet.** The experiments for CommNet are conducted on predator prey (PP) [4]. We set the punishment as $p = 0$ and $p = -0.5$ to create two PP tasks with different difficulties. As in Fig. 4, at the beginning of the attack, the performance of CommNet does not have obvious decrease, indicating that injecting random noise into the message is hard to disrupt agent coordination. As we gradually train the adversarial policy, there is a significant drop in the test return, with 40% and 33% decreases in the task of $p = 0$ and $p = -0.5$, respectively. Multi-agent cooperation has been severely affected due to the malicious messages. When the test return decreases to preset thresholds, i.e., 23 for $p = 0$ and 20 for $p = -0.5$, we freeze the adversarial policy network and start to train the message filter. As shown in the blue curves in Fig 4, with the message filter, test return steadily approaches the converged value of CommNet (red line), indicating that the message filter can effectively recover multi-agent coordination under attack.

**TarMAC.** We conduct message attack and defence on the Traffic Junction (TJ) environment [30]. There are two modes in TJ, i.e., easy and hard. The easy task has one junction of two one-way roads on a $7 \times 7$ grid with $N_{\text{max}} = 5$. In the hard task, its map has four junctions of two-way roads on a $18 \times 18$ grid and $N_{\text{max}} = 20$. As shown in Table 2, under attack, the success rate of TarMAC decreases in both the easy and the hard scenarios, demonstrating the vulnerability of TarMAC under malicious messages. After that, we equip TarMAC with the defender, then its performance improves considerably, demonstrating the merit of the message filter.

**NDQ.** We further examine the adversarial communication problem in NDQ. We perform evaluation on two StarCraft II Multi-Agent Challenge (SMAC) [26] scenarios, i.e., 3bane_vs_hM and 1o_2r_vs_4r, in which the communication between cooperative agents is necessary [38]. As presented in Fig. 5, under attack, the test win rate of NDQ decreases dramatically, demonstrating that NDQ is also vulnerable to message attacks. After applying the message filter as the defender, we can find the test win rate quickly reaches to around 60% in 3bane_vs_hM and 55% in 1o_2r_vs_4r, demonstrating that, once again, our defence methods succeed in restoring multi-agent coordination under message attacks.

### 4.3 Improving Robustness with $\mathcal{R}$-MACRL

We have demonstrated that many state-of-the-art MACRL algorithms are vulnerable to message attacks, and after applying the
message filter, multi-agent coordination can be recovered. In this part, we integrate the message filter into the framework of \(\mathcal{R}\)-MACRL and show that the robustness of MACRL algorithms can be significantly improved with \(\mathcal{R}\)-MACRL.

**Exploiting the message filter.** We first show that a single message filter is brittle and can be easily exploited if the attacker adapts to it. We perform experiments on CommNet by freezing the message filter and retraining the adversarial policy. As depicted in Fig. 6, the test return of the team gradually decreases as the training proceeds, with around 30% and 20% decreases in the task of \(\rho = 0\) and \(\rho = -0.5\) respectively. We also conduct experiments on NDQ and similar phenomenon is observed (see Appendix D). We conclude that even though the designed message filter is able to recover multi-agent cooperation under message attacks, its performance can degrade if faced with an adaptive attacker.

**Improving robustness.** To illustrate the improvement in robustness, we make comparisons between the defender trained by \(\mathcal{R}\)-MACRL and the vanilla defending method. For the defender trained with \(\mathcal{R}\)-MACRL, a population of attack policies are learned together with the defender, whose policy is also a mixture of sub-policies. In the vanilla training method, only a single defending policy is trained for the defender. We use the expected utility value (the accumulated team return) as the metric to compare the performance of the defender. Specifically, the larger the expected utility, the better the robustness. We denote the expected utility of the defender trained by \(\mathcal{R}\)-MACRL as \(u^R\), and the result for the vanilla one as \(u^V\). As shown in Table 3, \(\mathcal{R}\)-MACRL consistently outperforms the vanilla method over all the algorithms and environments. The improvement of expected utilities indicates that the defender trained by \(\mathcal{R}\)-MACRL is more robust. Intuitively, the defender benefits from exploring a wider range of the policy space with \(\mathcal{R}\)-MACRL, which enables the defender to maintain multi-agent coordination when faced with attacks.

**4.4 Scaling to Multiple Attackers**

To examine the performance of the method in scenarios where there is more than one attacker, we conduct experiments on NDQ in two new challenge tasks: 4bane_vs_hM and 1o_3r_vs_4r. In the former maps, i.e., 3bane_vs_hM and 1o_2r_vs_4r, there are only three agents in the team, making multiple attackers unreasonable. To mitigate this, we create 4bane_vs_hM and 1o_3r_vs_4r based on 3bane_vs_hM and 1o_2r_vs_4r by increasing one more agent to the team. We randomly sample two agents to be malicious. According to our assumptions (agents have no knowledge about who are malicious and malicious agents cannot communicate with each other), we can directly apply the attacking method to each of the malicious agents. As shown in Fig. 7, the attackers can quickly learn effective adversarial policies with less learning steps. On the other hand, the message filter can still successfully recover multi-agent cooperation under multiple attackers, though it takes much more learning steps to converge. We further evaluate the effectiveness of \(\mathcal{R}\)-MACRL on the two tasks and consistent results are observed: trained with \(\mathcal{R}\)-MACRL, the agents can obtain larger expected utilities. More results on other algorithms and environments are provided in Appendix D.

**4.5 Ablation Study and Analysis**

We have shown that by integrating the message filter into the framework of \(\mathcal{R}\)-MACRL, the robustness of MACRL algorithms can be improved. However, the performance of \(\mathcal{R}\)-MACRL directly is affected by the message filter. In this part, we will take a deeper look at how the message filter works.

**Which components contribute to the performance?** The message filter consists of two components: the anomaly detector and the message reconstructor. Here, we alternatively disable these two components in the message filter to study how they contribute to the performance. We run experiments on three scenarios, with each scenario corresponding to an MACRL algorithm. Following the former training procedure, we first train the original MACRL algorithm to obtain the action policy and the message policy; then we sample an attacker and train its adversarial policy by PPO; next we freeze the policy of the attacker and train the message filter (with the anomaly detector or the reconstructor disabled). As in Fig. 8, if we disable the message reconstructor in the message filter and replace it with a random message generator, after being attacked, multi-agent coordination is hard to be recovered, demonstrating the importance of the reconstructor. We further disable the anomaly detector and randomly choose an agent to reconstruct its message, as in Fig. 9, the defending performance is also poor. The ablation illustrates that both the anomaly detector and the reconstructor are critical in the message filter.

**How does the proposed method work?** Now we take a deeper look at how the message filter works. We conduct experiments

<table>
<thead>
<tr>
<th>Methods</th>
<th>Scenarios</th>
<th>(u^R)</th>
<th>(u^V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommNet</td>
<td>(\rho = 0)</td>
<td>41.75 ± 0.00</td>
<td>40.74 ± 0.23</td>
</tr>
<tr>
<td></td>
<td>(\rho = -0.5)</td>
<td>35.38 ± 1.28</td>
<td>31.91 ± 0.94</td>
</tr>
<tr>
<td>TarMAC</td>
<td>Easy</td>
<td>-4.08 ± 0.04</td>
<td>-4.24 ± 0.08</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>-9.33 ± 0.29</td>
<td>-9.80 ± 0.08</td>
</tr>
<tr>
<td>NDQ</td>
<td>3bane_vs_hM</td>
<td>12.36 ± 0.26</td>
<td>11.32 ± 0.15</td>
</tr>
<tr>
<td></td>
<td>1o_2r_vs_4r</td>
<td>17.70 ± 0.16</td>
<td>16.33 ± 0.54</td>
</tr>
</tbody>
</table>

**Table 3: Expected utilities for the defender trained with \(\mathcal{R}\)-MACRL and the vanilla approach.**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Scenarios</th>
<th>(u^R)</th>
<th>(u^V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDQ</td>
<td>4bane_vs_hM</td>
<td>15.86 ± 0.08</td>
<td>15.50 ± 0.29</td>
</tr>
<tr>
<td></td>
<td>1o_3r_vs_4r</td>
<td>18.39 ± 0.80</td>
<td>17.03 ± 0.53</td>
</tr>
</tbody>
</table>

**Table 4: Multiple attackers: Expected utilities for the defender trained with \(\mathcal{R}\)-MACRL and the vanilla approach.**

![Figure 7: Multiple attackers: Attack and defence on NDQ.](image)
on NDQ in the 3bane_vs_hM task. There are three agents in the team, of which agent 0 is the attacker and the others, namely agent 1 and agent 2, are benign agents. As shown in Fig. 10 (a)-(c), the action space of each agent contains eight elements. White cells in Q values correspond to illegal actions at the current state, and the action with red Q value is optimal, e.g., \( a_5 \) of agent 2. When attacked by the agent 0, the optimal action of agent 1 shifts from \( a_3 \) to \( a_1 \), leading to sub-optimality. After applying the message filter, the decision of agent 1 is corrected to \( a_3 \). We further examine the messages received by agent 1 and agent 2. As in Fig. 10 (d)-(e), the attacked messages are quite different from the original messages, while after applying the message filter, the distance between the original messages and the reconstructed messages becomes closer.

5 CONCLUSION

In this paper, we systematically examine the problem of adversarial communication in MACRL. The problem is of importance but has been largely neglected before. We first provide the formulation of adversarial communication. Then we propose an effective method to model message attacks in MACRL. Following that, we design a two-stage message filter to defend against message attacks. Finally, to improve robustness, we formulate the adversarial communication problem as a two-player zero-sum game and propose \( \mathcal{R} \)-MACRL to improve the robustness. Experiments on a wide range of algorithms and tasks show that many state-of-the-art MACRL methods are vulnerable to message attacks, while our algorithm can consistently recover multi-agent cooperation and improve the robustness of MACRL algorithms under message attacks.

ACKNOWLEDGMENTS

This research is partially supported by Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU), which is a collaboration between Singapore Telecommunications Limited (Singtel) and Nanyang Technological University (NTU) that is funded by the Singapore Government through the Industry Alignment Fund – Industry Collaboration Projects Grant.