24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Decoding the Imitation Security Game: Handling Attacker Imitative Behavior Deception

Thanh H. Nguyen[1] and **Nam Vu**[2] and **Amulya Yadav**[3] and **Uy Nguyen**[4]

Abstract. Real-world security problems are generally characterized by uncertainty about attackers' preferences, behavior, or other characteristics. To handle such uncertainties, security agencies (defender) typically rely on historical attack data to build a behavior model of the attacker, and incorporate this model into generating an effective defense strategy. For example, in wildlife protection, rangers can collect poaching signs (e.g., snares) to learn the behavior of poachers. However, in the real-world, a clever attacker can manipulate its attacks to *fool* the learning algorithm of the defender towards its own benefit. Unfortunately, existing state-of-the-art algorithms for generating defense strategies are not equipped to handle such deceptive behavior by the attacker, and this could lead to arbitrary losses for the defender. To address these challenges, this paper investigates a basic deception strategy of the attacker, termed *imitative behavior deception*, in which the attacker intentionally pretends to follow a specific behavior model and consistently plays according to that model, in order to optimize its utility. We have three main contributions. First, built upon previous work on attacker-behavior modeling, we introduce new algorithms to compute an optimal imitative behavior deception strategy of the attacker. Second, we propose a novel game-theoretic counter-deception algorithm which determines effective defense strategies, taking into account the deceptive behavior of the attacker. Third, we conduct extensive experiments, which shows that under the attacker's deception, the defender accrues a significant loss whereas the attacker achieves a significant gain in utility. Our experimental results also demonstrate the impact of our counter-deception algorithm on substantially diminishing the attacker's deception.

1 Introduction

Security is a critical concern around the world in a variety of domains, including public safety and security, wildlife protection, and cyber security. Defender-attacker Stackelberg security games (SSGs) have been widely applied to address these security problems. In fact, there are several high-impact deployments of SSG-based algorithms in the real world [24]. A key challenge in these security problems is that security agencies (defender) are often uncertain about attackers' behaviors and/or preferences. Thus, prior work in SSGs has proposed several different models of attacker behavior [6, 10, 20, 23, 27]. Essentially, in the learning phase of the defender, these models of the attacker's behavior are trained based on historical attack data. In the defender's planning phase, new game-theoretic algorithms are then developed to generate optimal defense strategies to play, assuming that an attacker responds according to the learnt behavior model.

A crucial assumption in existing work is that the attacker always responds honestly to the defender's algorithm so that the true behavior model of the attacker can be learned. However, given the defender's reliance on historical attack data, a *deceptive* attacker can alter its attack behavior to mislead the defender's learning algorithm. That is, the attacker may select attack actions during the learning phase of the defender which could influence the learning outcome to the benefit of the attacker in the planning phase. Facing such a deceptive attacker, the defender may suffer an arbitrary loss in his utility if he does not address the attacker's deception (we provide a concrete example in Section 4.1). A shrewd defender should thus consider the attacker's deception explicitly in designing his defense strategy.

This paper studies the strategic deception of the attacker in SSGs given the defender attempts to learn a behavior model of the attacker based on historical attack data and the attacker is aware of the defender's learning. We investigate a basic deception strategy, *imitative behavior deception*, in which the attacker pretends to follow a behavior model (which may not represent the true behavior of the attacker) and consistently plays the game according to this model. As a result, the defender will eventually learn this (deceptive) behavior model of the attacker. The defender then decides on a defense strategy based on the learning outcome. The ultimate goal of the attacker is to find an optimal deceptive behavior model to imitate such that the attacker's utility is maximized in the planning phase.

This paper has three main contributions. First, based on previous work on attacker behavior modeling [27], we formulate the problem of finding an optimal attacker deception strategy as a bi-level optimization problem. We propose two new algorithms (named Discretization-based deception and KKT (Karush-Kuhn-Tucker)-based deception) to solve this problem. The former discretizes the domain of the behavior model parameter and iteratively searches through the discretized space to find an optimal deceptive model. The latter leverages KKT conditions [2] to reformulate the attacker-deception problem as a single-level optimization problem.

Second, we propose a novel game-theoretic *counter-deception* algorithm to generate effective defense strategies, taking into account the attacker's deception. Essentially, our algorithm generates a *mapping function* identifying which defense strategy to play against each possible (learnt) deceptive behavior model. Finding an optimal mapping function is challenging given that the domain of model parameters is continuous and the resulting optimization problem is non-convex. To tackle this challenge, we exploit intrinsic properties of extreme points and apply optimization techniques such as piece-wise linear approximation to convert the counter-deception problem to a Mixed Integer Linear Program (MILP), which can be solved exactly.

[1] University of Oregon, USA, email: tnguye11@uoregon.edu
[2] Posts and Telecommunications Institute of Technology, Vietnam, email: {vuhoainam7121998}@gmail.com
[3] Pennsylvania State University, USA, email: amulya@@psu.edu
[4] Posts and Telecommunications Institute of Technology, Vietnam, email: {nguyenquocuy2008}@gmail.com

Finally, we conduct a rigorous evaluation of our proposed methods on a variety of different game settings. We provide a detailed empirical analysis of the attacker's deception, showing that the attacker obtains a significant benefit by playing deceptively while the defender suffers a significant loss when he does not address the attacker's deception. Furthermore, our results on counter-deception demonstrate the advantage of our counter-deception algorithm in drastically reducing the defender's utility loss (due to the attacker's deception).

2 Related Work

Behavioral Game Theory. There is a long line of prior work on SSGs that focuses on developing behavior models for the attacker. Parameterized models of attacker behavior such as Quantal Response, and other machine learning models have been studied for SSGs [6, 11, 17]. These models provide general techniques for modeling the attacker decision making in security games. Prior work in this area assumes that the attacker always plays truthfully. Thus, existing algorithms for generating defense strategies would be vulnerable against deceptive attacks by an attacker who is aware of the defender's learning. Our work is different in that our proposed algorithms explicitly account for such a strategic deceptive attacker by planning strategies to counter the attacker's deception.

Deception in Security Games. Deception is a widely studied research area in security [3, 4, 8, 30]. In SSG literature, in particular, a lot of prior work has studied deception by the defender, i.e., the defender exploits his knowledge regarding uncertainties to mislead the attacker's decision making [7, 21, 22, 26, 29]. More recently, deception on the attacker's side has been studied. Existing work on attacker deception focuses on situations in which the defender is uncertain about the attacker type [5, 18, 19]. Our work, on the other hand, studies the attacker deception when the defender attempts to learn the attacker's behavior based on historical attack data.

Adversarial Machine Learning. Previous work on adversarial learning has investigated various types of attacks to machine learning algorithms in various learning domains [1, 9, 13, 14, 25]. Prediction accuracy is the main measure used in existing work. In particular, the learner attempts to find a robust learning algorithm which maximizes the prediction accuracy. The problem of attacker deception in SSGs can be considered as a type of causative attack to the defender's learning algorithms. However, unlike prior work in this space, the defender's goal in our problem is to find an optimal defense strategy (based on the learning outcomes) which maximizes his utility.

3 Background

Stackelberg security games (SSGs) [24]. In SSGs, there is a set of important targets $\mathbf{N} = \{1, 2, \ldots, N\}$. The defender has $K < N$ security resources to protect these targets against an attacker. A pure strategy of the defender is an allocation of these resources over the targets. A mixed strategy of the defender is a probability distribution over all pure strategies. Each mixed strategy of the defender can be compactly represented as a coverage probability vector $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ in which $x_i \in [0, 1]$ is the probability that the defender protects target i and $\sum_i x_i \leq K$ [12]. We denote by $\mathbf{X}$ the set of all mixed strategies of the defender. In SSGs, the defender plays first by committing to a mixed strategy, and the attacker responds against this strategy by choosing a single target to attack.

When the attacker attacks target i, it obtains a reward of R_i^a while the defender receives a penalty of P_i^d if the defender is not protecting that target. Conversely, if the defender is protecting target i, the attacker gets a penalty of $P_i^a < R_i^a$ while the defender receives a reward of $R_i^d > P_i^d$. The expected utility of the defender and attacker if the attacker attacks target i can be represented as follows:

$$U_i^d(x_i) = x_i R_i^d + (1 - x_i) P_i^d$$
$$U_i^a(x_i) = x_i P_i^a + (1 - x_i) R_i^a$$

Strong Stackelberg Equilibrium (SSE) [24]. In an SSE, both players play a best response against each other's strategies. In particular, a pair of strategies $(\mathbf{x}^*, \text{BR}(\mathbf{x}^*))$ are an SSE if and only if:

$$\mathbf{x}^* \in \text{argmax}_{\mathbf{x} \in \mathbf{X}} U^d_{\text{BR}(\mathbf{x})}(x_{\text{BR}(\mathbf{x})})$$
$$\text{BR}(\mathbf{x}) \in \text{argmax}_{i \in \mathbf{N}} U_i^a(x_i)$$

Quantal Response (QR) model [27]. QR is a well-known behavior model used to predict human (attacker) decision making in behavioral economics, game theory, and security games [15, 16, 27]. In SSGs in particular, QR predicts the probability the attacker attacks each target i, using the following softmax function:

$$q_i(\mathbf{x}, \lambda) = \frac{e^{\lambda U_i^a(x_i)}}{\sum_j e^{\lambda U_j^a(x_j)}} \tag{1}$$

where λ is the model parameter that governs the attacker's rationality. In particular, when $\lambda = 0$, the attacker attacks every target uniformly at random. When $\lambda = +\infty$, the attacker is perfectly rational. Given that the attacker follows QR, the defender and attacker's expected utility is computed as an expectation over all targets, as follows:

$$U^d(\mathbf{x}, \lambda) = \sum\nolimits_i q_i(\mathbf{x}, \lambda) U_i^d(x_i) \tag{2}$$
$$U^a(\mathbf{x}, \lambda) = \sum\nolimits_i q_i(\mathbf{x}, \lambda) U_i^a(x_i) \tag{3}$$

When the attacker follows a QR model, the defender's objective is to find an optimal strategy which maximizes his expected utility:

$$\max_{\mathbf{x} \in \mathbf{X}} U^d(\mathbf{x}, \lambda)$$

4 Attacker Imitative Behavior Deception

Since the defender relies on an attacker behavior model to plan his defense strategies [27], a clever attacker, who is aware of the defender's learning, can behave differently to mislead the defender. The attacker aims at obtaining the most benefit from such deception. This paper studies the attacker's *imitative behavior deception*. Essentially, the attacker pretends to have a behavior model (which may not represent its true behavior) and consistently plays the game according to that model. Eventually, the defender would learn a wrong model of the attacker's behavior, resulting in an ineffective defense strategy.

This work considers a security scenario in which the defender uses QR to learn the attacker's behavior and the attacker knows that. The deceptive attacker attempts to find a value of the parameter λ and plays based on the QR model with that deceptive λ. The defender will eventually learn this deceptive value of λ and play accordingly.

4.1 Example

To illustrate the attacker's benefit and the defender's loss as a result of the attacker's imitative behavior deception, we provide an example of a simple 2-target game (shown in Table 1). The row player is the defender and the column player is the attacker.

In each cell of this payoff matrix, the first number represents the defender's payoff and the second number is the attacker's payoff. For example, if the attacker attacks target t_1 and the defender also protects t_1, the defender obtains a payoff of 10 while the attacker receives a payoff of 1. The interpretation of other cells is similar. In this game, the defender has one security resource ($K = 1$). If the attacker is non-deceptive (i.e., the attacker always plays a best response to any strategy of the defender), then the defender will eventually learn to play an SSE strategy. In this game, the equilibrium strategy of the defender is to always protect target t_1 while leaving target t_2 unprotected. The attacker's corresponding best response is to attack target t_1. As a result, the defender receives a utility $U^d_{non-deception} = 10$ and the attacker obtains a utility of $U^a_{non-deception} = 1$.

	Target 1	Target 2
Target 1	10, 1	-100, 0
Target 2	-20, 10	1, -1

Table 1: An example of a simple 2-target game

On the other hand, the attacker can play deceptively. For example, the attacker intentionally attacks targets uniformly at random. In this case, the defender learns $\lambda = 0$. The corresponding optimal strategy of the defender is to always protect target t_2 while leaving target t_1 unprotected. As a result, the defender receives a utility of $0.5 \times (-20) + 0.5 \times 1 = -9.5 < U^d_{non-deception}$. The attacker obtains $0.5 \times 10 + 0.5 \times (-1) = 4.5 > U^a_{non-deception}$. This example shows that in the presence of the attacker deception, the defender would suffer a significant loss while the attacker obtains a significant benefit if the defender does not address the deception of the attacker.

4.2 Imitative Behavior Deception Formulation

The problem of finding an optimal deceptive parameter λ for the attacker can be formulated as the following optimization problem:

$$\max_\lambda \; U^a(\mathbf{x}^*(\lambda), \lambda) \tag{4}$$

$$\text{s.t. } \mathbf{x}^*(\lambda) \in \operatorname{argmax}_{\mathbf{x}\in\mathbf{X}} \; U^d(\mathbf{x}, \lambda) \tag{5}$$

which maximizes the attacker's expected utility. $\mathbf{x}^*(\lambda)$ is the defender's optimal strategy with respect to the QR parameter λ.

Proposition 1. *In zero-sum security games, the attacker's optimal imitative behavior deception strategy is to play truthfully.*

Proof. In zero-sum games, we have the zero-sum relation between players' payoffs: $R^a_i = -P^d_i$ and $P^a_i = -R^d_i$ for all targets i. Therefore, the inner optimization problem (5) is equivalent to:

$$\mathbf{x}^*(\lambda) \in \operatorname{argmin}_{\mathbf{x}\in\mathbf{X}} U^a(\mathbf{x}, \lambda)$$

As a result, finding an optimal deceptive λ becomes a maximin problem, represented as: $\max_\lambda \min_{\mathbf{x}\in\mathbf{X}} \; U^a(\mathbf{x}, \lambda)$. On the other hand, in zero-sum games, SSE strategies are equivalent to minimax strategies [28]. Therefore, the attacker's utility for playing truthfully is the objective of the following minimax problem: $\min_{\mathbf{x}\in\mathbf{X}} \max_i U^a_i(x_i)$.

Note that for every λ and every defender strategy $\mathbf{x}$, the attacker's expected utility with respect to $(\lambda, \mathbf{x})$ is less than its expected utility for playing a best response against $\mathbf{x}$. Therefore,

$$\min_{\mathbf{x}\in\mathbf{X}} U^a(\mathbf{x}, \lambda) \le \min_{\mathbf{x}\in\mathbf{X}} \max_i U^a_i(x_i), \forall\lambda$$

which means that the attacker's optimal deception is to always play a best response (i.e., the attacker is truthful). □

In general-sum games, finding an optimal deceptive λ is a bi-level optimization problem as shown in (4–5), which is not easy to solve. In the following, we propose two efficient algorithms to solve it.

4.3 Discretization-based Deception Algorithm

In our algorithm, we propose to discretize values of the model parameter λ into a finite set $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_L\}$ with an equal step size $\sigma_\lambda > 0$, i.e., $\lambda_1 = 0$, $\lambda_{l+1} - \lambda_l = \sigma_\lambda$ for all $l = 1, 2, \ldots, L-1$ where L is the number of values in consideration. We limit the search space of λ to Λ and iterate over this set to find an optimal deceptive parameter λ. At each iteration l, we compute the defender's optimal strategy $\mathbf{x}^*(\lambda_l)$ with respect to λ_l. This step can be done using existing game-theoretic methods [27]. We then computes the corresponding expected utility of the attacker `curAttU` and uses it to update the optimal deceptive value of λ, `optAttLambda`. Finally, our algorithm provides an approximation of the optimal value of λ and a lower bound of the attacker's maximum deceptive utility.

4.4 KKT-based Deception Algorithm

We propose to apply Karush-Kuhn-Tucker (KKT) conditions [2] to convert the inner optimization (5) into a set of mathematical constraints. As a result, we obtain the following single-level optimization problem to find an optimal deceptive λ, formulated as follows:

$$\max_{\lambda,\mathbf{x},\mu^1,\mu^0,\nu} \; U^a(\mathbf{x}, \lambda) \tag{6}$$

$$\text{s.t. } 0 \le x_i \le 1, \forall i, \text{ and } \sum\nolimits_i x_i = K \tag{7}$$

$$-\frac{\partial U^d(\mathbf{x}, \lambda)}{\partial x_i} + \mu^1_i - \mu^0_i + \nu = 0, \forall i \tag{8}$$

$$\mu^1_i(x_i - 1) = 0, \forall i \tag{9}$$

$$\mu^0_i x_i = 0, \forall i \tag{10}$$

$$\mu^1_i, \mu^0_i \ge 0, \forall i. \tag{11}$$

For each λ, constraints (7–11) are KKT conditions of the inner optimization problem (5). The variables $\{\mu^1_i\}$, $\{\mu^0_i\}$ and ν are dual optimal, which corresponds to the constraints $x_i \le 1$, $x_i \ge 0$, and $\sum_i x_i = K$, respectively. We denote by $(\mathbf{x}^*, \lambda^*)$ a part of the optimal solution of (6–11). Note that $(\mathbf{x}^*, \lambda^*)$ is not the optimal solution of (4–5) since (5) is non-convex and thus KKT conditions represented in (7–11) are not sufficient conditions for $\mathbf{x}^*$ to be an optimal solution of (5). Finally, we obtain the following proposition:

Proposition 2. *The optimization problem (6–11), denoted by $(\mathbf{x}^*, \lambda^*)$, returns a lower bound $U^a(\mathbf{x}^*(\lambda^*), \lambda^*)$ and an upper bound $U^a(\mathbf{x}^*, \lambda^*)$ of the attacker's optimal deceptive utility in (4–5).*

Proof. Given a λ, the optimization problem in (5) is generally non-convex and all of its local optimal solutions satisfy constraints (7–11) w.r.t λ. Thus, the feasible region of the defender strategy, $\mathbf{x}$, represented by the constraints (7–11) is a super set of the optimal solution set of (5). Therefore, the optimal objective of (6–11), $U^a(\mathbf{x}^*, \lambda^*)$, is an upper bound of the attacker's optimal deceptive utility.

Given λ^* obtained by solving (6–11), we can solve (5) to obtain a corresponding optimal strategy $\mathbf{x}^*(\lambda^*)$ [27]. This $(\mathbf{x}^*(\lambda^*), \lambda^*)$ is a feasible solution of (4–5). Thus, the attacker's expected utility, $U^a(\mathbf{x}^*(\lambda^*), \lambda^*)$, is a lower bound of the optimal objective value of (4–5) (i.e., the attacker's optimal deceptive utility). □

5 Defender Counter-Deception

To tackle the attacker's imitative behavior deception, the defender has to choose which defense strategy to play with respect to each possible learnt (deceptive) value of λ. In other words, the defender commits to a *mapping function* $\mathcal{H} : \mathbb{R} \rightarrow \mathbf{X}$ from a learnt (deceptive) value of λ to a defense strategy. The defender's goal is to find a mapping function which maximizes the defender's expected utility, given that the attacker would choose an optimal deceptive value of λ accordingly. In this work, we propose a novel game-theoretic counter-deception algorithm which determines an optimal mapping function for the defender. Essentially, finding an optimal mapping function $\mathcal{H}$ can be represented as the following bi-level optimization:

$$\max_{\mathcal{H}} U^d(\mathcal{H}(\lambda^*(\mathcal{H})), \lambda^*(\mathcal{H})) \tag{12}$$

$$\text{s.t. } \lambda^*(\mathcal{H}) \in \operatorname{argmax}_{\lambda} U^a(\mathcal{H}(\lambda), \lambda) \tag{13}$$

where $\lambda^*(\mathcal{H})$ is the optimal deceptive value of λ with respect to $\mathcal{H}$. However, finding a closed form for the mapping function $\mathcal{H}$ is challenging given that it involves a bi-level optimization problem (12–13). Therefore, we propose to divide the value range of λ into a finite number of intervals $\mathcal{I} = \{[\lambda_0, \lambda_1], [\lambda_1, \lambda_2], \dots, [\lambda_L, \lambda_{L+1}]\}$ where $\lambda_0 = 0$ and $\lambda_{L+1} = +\infty$. We then find an optimal mapping function which maps each interval $[\lambda_l, \lambda_{l+1}]$ to a single defense strategy $\mathbf{x}^{l+1}$. Intuitively, for any learnt (deceptive) $\lambda \in [\lambda_l, \lambda_{l+1}]$, the defender will play the same strategy $\mathbf{x}^{l+1}$ accordingly. In the following, we first provide a proposition to find an optimal deception strategy of the attacker given the function $\mathcal{H}$.

Proposition 3. *Given the mapping function $\mathcal{H} : \mathcal{I} \rightarrow \mathbf{X}$, the attacker's optimal deception belongs to $\{\lambda_1, \lambda_2, \dots, \lambda_L, \lambda_{L+1}\}$.*

Proof. For any value of λ in each interval $[\lambda_l, \lambda_{l+1}]$, the defender commits to the same strategy $\mathbf{x}^{l+1}$. Finding an optimal deceptive value of λ in an interval $[\lambda_l, \lambda_{l+1}]$ is thus formulated as follows:

$$\max_{\lambda \in [\lambda_l, \lambda_{l+1}]} U^a(\mathbf{x}^{l+1}, \lambda) \tag{14}$$

The attacker's expected utility $U^a(\mathbf{x}^{l+1}, \lambda)$ is increasing in λ since its derivative, $\frac{\partial U^a(\mathbf{x}^{l+1}, \lambda)}{\partial \lambda}$, is equal to:

$$\frac{1}{2} \sum\nolimits_{i,k} q_i(\mathbf{x}^{l+1}, \lambda) q_k(\mathbf{x}^{l+1}, \lambda) \left[U_i^a(x_i^{l+1}) - U_k^a(x_k^{l+1}) \right]^2 \geq 0$$

Thus, the optimal solution of (14) is $\lambda = \lambda_{l+1}$. □

5.1 Mixed Integer Non-Linear Program

According to the Proposition 3, the attacker's optimal deception strategy is to choose a value of λ in the set $\{\lambda_1, \lambda_2, \dots, \lambda_L, \lambda_{L+1}\}$ such that the attacker's expected utility is maximized. Therefore, we propose a Mixed Integer Non-Linear Program (`MINLP`) (15–18) to find an optimal mapping function $\mathcal{H}$ which maximizes the defender's expected utility. The binary variable $z_l \in \{0, 1\}$ indicates if the attacker chooses λ_l to imitate ($z_l = 1$) or not ($z_l = 0$) where $l \in \{1, 2, \dots, L+1\}$. The variable r represents the attacker's optimal deceptive utility and T is a very large constant. Constraint (16) ensures that the attacker will mimic one of the parameter values in $\{\lambda_1, \lambda_2, \dots, \lambda_{L+1}\}$. Constraint (17) indicates that r is greater than expected utility of the attacker with respect to every deception choice λ_l. Constraint (18) ensures that r is less than the attacker's utility at λ_l if λ_l is the best deception choice for the attacker (i.e., $z_l = 1$). Therefore, the combination of constraints (17-18) guarantees that the attacker will select the best deceptive value of λ.

$$\max_{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^{L+1}} \sum\nolimits_l z_l U^d(\mathbf{x}^l, \lambda_l) \tag{15}$$

$$\text{s.t.} \sum\nolimits_l z_l = 1, z_l \in \{0, 1\}, \forall l \tag{16}$$

$$r \geq U^a(\mathbf{x}^l, \lambda_l), \forall l \tag{17}$$

$$r \leq U^a(\mathbf{x}^l, \lambda_l) + (1 - z_l)T, \forall l \tag{18}$$

We can solve (15–18) using any non-convex solver. However, solving this `MINLP` is computationally expensive. Moreover, there is no guarantee of the quality of the returned solution. Therefore, we propose a novel scalable counter deception algorithm to find an approximately optimal mapping function $\mathcal{H}$.

5.2 Scalable Counter-Deception Algorithm

Overall, our algorithm has three main components: (i) Dividing the problem into multiple sub-problems — each sub-problem corresponds to a particular optimal deception strategy of the attacker; (ii) Applying binary search to convert the objective and constraint functions of each sub-problem into variable-separable functions — each resulting function is a sum of multiple uni-variate terms in which each of the variables represents an individual coverage probability of the defender; and (iii) Applying piecewise linear approximation to reformulate each sub-problem as a `MILP`. In the following, we elaborate on each of the three components of our algorithm.

5.2.1 Multiple non-linear programs

We divide the problem of finding an optimal mapping function $\mathcal{H}$ into multiple non-linear sub-problems, each of which corresponds to a different optimal deception choice of the attacker. In particular, conditioned on an optimal deception choice λ_l, we obtain:

$$\mathbf{NLP}_l : \max_{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^{L+1}} U^d(\mathbf{x}^l, \lambda_l) \tag{19}$$

$$\text{s.t. } U^a(\mathbf{x}^l, \lambda_l) \geq U^a(\mathbf{x}^j, \lambda_j), \forall j \tag{20}$$

which maximizes the defender's expected utility given that the attacker chooses λ_l to imitate. Constraint (20) guarantees that λ_l is the optimal deceptive value of λ for the attacker. By solving $\mathbf{NLP}_l$, we obtain a mapping function $\mathcal{H}_l$ that optimizes the defender's expected utility when the attacker deception choice is λ_l.

Finally, the optimal mapping function $\mathcal{H}$ is chosen as the best one among $\{\mathcal{H}_l\}$, i.e., the $\mathcal{H}_l$ which returns the highest utility for the defender. In the following, we provide Proposition 4 which determines part of the optimal solution of $\mathbf{NLP}_l$, for every l.

Proposition 4. *Denote by $\{\mathbf{x}^{j,*}\}_{j=1}^{L+1}$, the optimal solutions of the following optimization problems:*

$$\mathbf{x}^{j,*} \in \operatorname{argmin}_{\mathbf{x} \in \mathbf{X}} U^a(\mathbf{x}, \lambda_j), \forall j$$

Then, $\{\mathbf{x}^{j,*}\}_{j \neq l}$ is part of the optimal solution of $\mathbf{NLP}_l$ for all l.

Proof. In $\mathbf{NLP}_l$, strategies of the defender, $\{\mathbf{x}^j\}$ for all $j \neq l$, only appear in the RHSs of constraint (20). Therefore, for any feasible solution $(\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^{L+1})$ of $\mathbf{NLP}_l$, then $(\{\mathbf{x}^{j,*}\}_{j \neq l}, \mathbf{x}^l)$ is also a feasible solution since $\{\mathbf{x}^{j,*}\}_{j \neq l}$ provides lowest values of the RHSs. As a result, $\{\mathbf{x}^{j,*}\}_{j \neq l}$ is part of the optimal solution for $\mathbf{NLP}_l$. □

Algorithm 1: Binary search

Initialize $lb = \min_i P_i^d$ and $ub = \max_i R_i^d$;
while $ub - lb \geq \epsilon$ **do**
  Set $v_0 = \frac{ub+lb}{2}$;
  Solve $(\mathbf{x}^*, \textit{isFeasible}) = \texttt{feasibility.check}(v_0)$;
  if *isFeasible* **then** $lb = v_0$; **else** $ub = v_0$;
if *isFeasible* **then** $\mathbf{x}^l = \mathbf{x}^*$; **else**
$(\mathbf{x}^l, \textit{isFeasible}) = \texttt{feasibility.check}(lb)$;

We denote by $r^* = \max_j U^a(\mathbf{x}^{j,*}, \lambda_j)$ which is a constant. Based on Proposition 4, the sub-problem $\mathbf{NLP}_l$ in (19–20) can be now simplified as follows (constraint (22) is equivalent to constraint (20)):

$$\mathbf{NLP}_l : \max_{\mathbf{x}^l} U^d(\mathbf{x}^l, \lambda_l) \tag{21}$$

$$\text{s.t. } U^a(\mathbf{x}^l, \lambda_l) \geq r^* \tag{22}$$

This resulting optimization problem is still non-convex. Thus, we propose to apply a binary search to convert the objective and constraint functions of (21–22) into variable-separable functions. Each resulting function is a sum of multiple terms, and each term only depends on the defender's coverage probability at a particular target. Based on this conversion, we can apply piecewise linear approximation to represent the feasibility problem of the binary search as a MILP, which can be solved exactly.

5.2.2 *Solving* $\mathbf{NLP}_l$*: Binary search*

The overview of binary search is illustrated in Algorithm 1. Given some lower and upper bound on the objective, say (lb, ub), it starts by solving the feasibility problem of whether there exists a feasible strategy $\mathbf{x}^l$ such that the defender's expected utility is no less than a value $v_0 = (lb + ub)/2$. That is, is there $\mathbf{x}^l$ such that:

$$U^a(\mathbf{x}^l, \lambda_l) \geq r^*$$
$$U^d(\mathbf{x}^l, \lambda_l) \geq v_0$$

which is equivalent to (see Equations (1–3)):

$$\mathbf{FSP}_l : \exists \mathbf{x}^l \text{ s.t. } \sum\nolimits_i e^{\lambda_l U_i^a(x_i^l)}(U_i^a(x_i^l) - r^*) \geq 0 \tag{23}$$

$$\sum\nolimits_i e^{\lambda_l U_i^a(x_i^l)}(U_i^d(x_i^l) - v_0) \geq 0? \tag{24}$$

If it is feasible, then binary search updates the lower bound as $lb = v_0$. Otherwise, it updates the upper bound as $ub = v_0$. Binary search then solves the feasibility problem again with the updated lower bound or upper bound. This feasibility-check process will continue until it reaches a stopping condition (i.e., $ub - lb < \epsilon$). To solve $\mathbf{FSP}_l$ given a v_0 in (23–24), we introduce the following corresponding optimization problem:

$$\max_{\{x_i^l\}_i, \delta} \delta \tag{25}$$

$$\delta \leq \sum\nolimits_i e^{\lambda_l U_i^a(x_i^l)}(U_i^a(x_i^l) - r^*) \tag{26}$$

$$\delta \leq \sum\nolimits_i e^{\lambda_l U_i^a(x_i^l)}(U_i^d(x_i^l) - v_0) \tag{27}$$

which finds the defender strategy that maximizes the minimum value (i.e., δ) between the LHSs of the constraints (23–24) of the feasibility problem $\mathbf{FSP}_l$. Proposition 5 shows the equivalence between $\mathbf{FSP}_l$ and the optimization problem (25–27).

Proposition 5. *We denote by δ^* the optimal objective value of the problem (25–27). If $\delta^* \geq 0$, then the problem $\mathbf{FSP}_l$ is feasible. Otherwise, if $\delta^* < 0$, then $\mathbf{FSP}_l$ is infeasible.*

The proof of Proposition 5 is straightforward. If $\delta^* \geq 0$, then the optimal solution $\{x_i^l\}_i$ returned by solving (25–27) is a feasible solution of $\mathbf{FSP}_l$. Conversely, if $\delta^* < 0$, it means the maximum of the minimum value between the LHSs of the constraints (23–24) is strictly less than zero. Therefore, $\mathbf{FSP}_l$ is infeasible.

5.2.3 *Solving* $\mathbf{FSP}_l$*: Piecewise linear approximation*

The RHSs of (26–27) are the sum of uni-variate functions:

$$f_i^a(x_i^l) = e^{\lambda_l U_i^a(x_i^l)}(U_i^a(x_i^l) - r^*), \forall i$$
$$f_i^d(x_i^l) = e^{\lambda_l U_i^a(x_i^l)}(U_i^d(x_i^l) - v_0), \forall i$$

Therefore, we can apply piecewise linear approximation to linearize these functions. Overall, each feasible region $[0, 1]$ of the defender's coverage probability at target i, x_i^l, is divided into M equal segments $\{[0, \frac{1}{M}], [\frac{1}{M}, \frac{2}{M}], \ldots, [\frac{M-1}{M}, 1]\}$. Then the functions $f_i^a(x_i^l)$ can be approximated by using the segments connecting pairs of consecutive points $\big(\frac{m-1}{M}, f_i(\frac{m-1}{M})\big)$ and $\big(\frac{m}{M}, f_i(\frac{m}{M})\big)$. In particular, $f_i^a(x_i^l)$ is piecewise linear approximated as follows:

$$f_i^a(x_i^l) = f_i^a(0) + \sum\nolimits_{m=1}^{M} s_m^a x_{i,m}^l$$

where s_m^a is the slope of the m^{th} segment, computed as follows: $s_m^a = M(f_i^a(\frac{m}{M}) - f_i^a(\frac{m-1}{M}))$. In addition, $x_{i,m}^l$ refers to the portion of the defender's coverage probability x_i^l at target i which belongs to the m^{th} segment. In other words, $x_i^l = \sum_m x_{i,m}^l$. Note that if $x_i^l \geq \frac{m}{M}$, the m^{th} segment is fully covered by the defender's coverage probability, x_i^l, which means $x_{i,m}^l = \frac{1}{M}$. If $x_i^l < \frac{m-1}{M}$, the m^{th} segment is completely uncovered and thus $x_{i,m}^l = 0$. Finally, if $\frac{m-1}{M} \leq x_i^l < \frac{m}{M}$, the m^{th} segment is partially covered, which means $x_{i,m}^l = x_i^l - \frac{m-1}{M}$. The function $f_i^d(x_i^l)$ is approximated similarly:

$$f_i^d(x_i^l) = f_i^d(0) + \sum\nolimits_{m=1}^{M} s_m^d x_{i,m}^l$$

where s_m^d is the slope of the m^{th} segment: $s_m^d = M(f_i^d(\frac{m}{M}) - f_i^d(\frac{m-1}{M}))$. Based on piecewise linear approximation, we introduce the following MILP representation of (25–27):

$$\mathbf{PLA}_l : \max \delta \tag{28}$$

$$\text{s.t. } \delta \leq \sum\nolimits_i [f_i^d(0) + \sum\nolimits_{m=1}^{M} s_m^d x_{i,m}^l] \tag{29}$$

$$\delta \leq \sum\nolimits_i [f_i^a(0) + \sum\nolimits_{m=1}^{M} s_m^a x_{i,m}^l] \tag{30}$$

$$\sum\nolimits_i x_{i,m}^l \leq K, 0 \leq x_{i,m}^l \leq \frac{1}{M}, \forall i, m \tag{31}$$

$$h_{i,m}\frac{1}{M} \leq x_{i,m}^l, \forall i, m \tag{32}$$

$$x_{i,m+1}^l \leq h_{i,m}, \forall i, m \leq M-1 \tag{33}$$

$$h_{i,m} \in \{0, 1\}, \forall i, m. \tag{34}$$

where RHSs of constraints (29–30) are the piecewise linear approximations of RHSs of the constraints (26–27). Constraints (31–34) guarantee that the segmentation of the coverage probability of the defender at each target is valid. In particular, the binary variable $h_{i,m}$ indicates if there is any portion of the defender's coverage probability x_i^l which fully covers the m^{th} segment (i.e., $h_{i,m} = 1$) or not

(i.e., $h_{i,m} = 0$). Constraints (31–32) ensure that if $h_{i,m} = 1$, then $x^l_{i,m} = \frac{1}{M}$. Finally, constraint (33) ensures that if $h_{i,m} = 0$, then no portion of the defender's coverage probability at i overlaps with the segments $\left[\frac{m'}{M}, \frac{m'+1}{M}\right]$ with $m' \geq m$ (i.e., $x^l_{i,m+1} = 0$).

6 Experiments

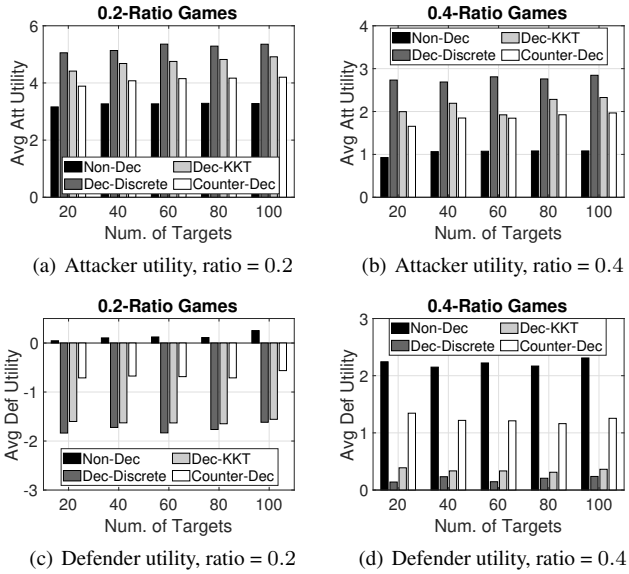
In our experiments, we evaluate both the solution quality and runtime performance of our proposed algorithms. We focus on analyzing the impact of the attacker's deception and the defender's counter deception in terms of utility of both players. In addition, we examine the attacker's deception behavior when the defender addresses (and does not address) the attacker's deception. We compare four algorithms:

1. `Non-Dec`: the attacker is non deceptive and the defender also assumes so. As a result, both play `SSE` strategies.
2. `Dec-Discrete`: the attacker is deceptive while the defender assumes the attacker is not. The attacker's deception strategy is generated based on our discretization-based deception algorithm.
3. `Dec-KKT`: the attacker is deceptive while the defender assumes the attacker is not. The attacker's deception strategy is generated based on our `KKT`-based deception algorithm.
4. `Counter-Dec`: the attacker is deceptive while the defender also assumes so. The defender's counter-deception strategies (i.e., the mapping function) are generated based on our scalable approximate game-theoretic counter-deception algorithm. Note that, we do not show the results of the `MINLP` (15–18) formulation since it is not scalable (it takes approximately more than four hours to solve one single 20-target game instance).

We use `Matlab` (`https://www.mathworks.com`) to solve non-linear programs and `Cplex` (`https://www.ibm.com/analytics/cplex-optimizer`) to solve `MILP`s involved in the evaluated algorithms. We use the covariance game generator, `GAMUT` (`http://gamut.stanford/edu`) to generate rewards and penalties of players within the range of $[1, 10]$ and $[-10, -1]$, respectively. The covariance value $r \in [-1, 0]$ controls the correlations between the defender and the attacker's payoff. In particular, when $r = -1.0$, the generated games are zero-sum. When $r = 0.0$, there is no relation between the players' payoff. Since the attacker plays truthfully in zero-sum games (Proposition 1), we only consider the covariance value in the range of $[-0.8, 0]$ with the step size of 0.2: $r \in \{-0.8, -0.6, -0.4, -0.2, 0\}$. Regarding the value range for the parameter λ, we limit its range to $[0, 10]$ to avoid the issue of overflow due to the computation of exponential functions. We discretize this range into a set of 50 values for λ with a step size of 0.2: $\lambda \in \{0, 0.2, \ldots, 10\}$ for `Dec-Discrete` and `Counter-Dec`.

6.1 Varying number of targets

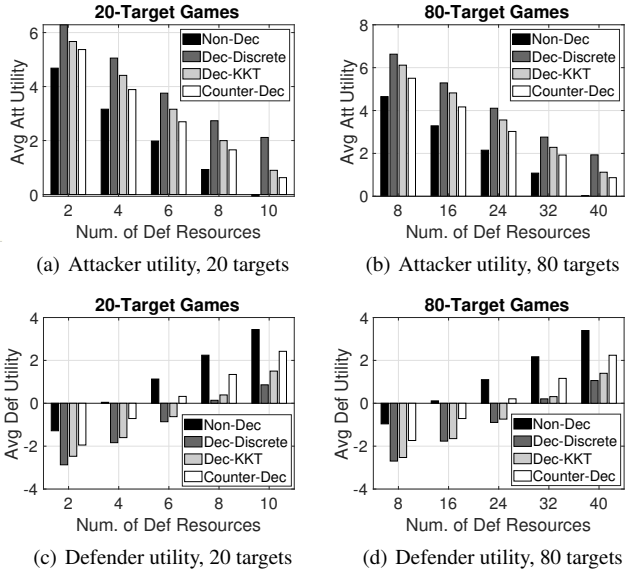
In our first set of experiments, we examine the players' expected utility when varying the number of targets. The results are shown in Figure 1 when the ratio of the number of security resources to the number of targets (i.e., $\frac{K}{N}$) is 0.2 and 0.4 respectively. The x-axis is the number of targets and the y-axis represented the average expected utility of the players. Each data point in Figure 1 is averaged over 100 different games; 20 games for each covariance value of r. Figure 1 shows that the attacker obtains a significantly higher average utility for playing deceptively compared with when the attacker is non-deceptive (`Non-Dec`). Comparing between our two approximate attacker-deception algorithms, `Dec-Discrete` results in a substantially higher utility for the attacker than `Dec-KKT`. This result shows



(a) Attacker utility, ratio = 0.2 (b) Attacker utility, ratio = 0.4

(c) Defender utility, ratio = 0.2 (d) Defender utility, ratio = 0.4

Figure 1: Evaluation on solution quality, varying number of targets



(a) Attacker utility, 20 targets (b) Attacker utility, 80 targets

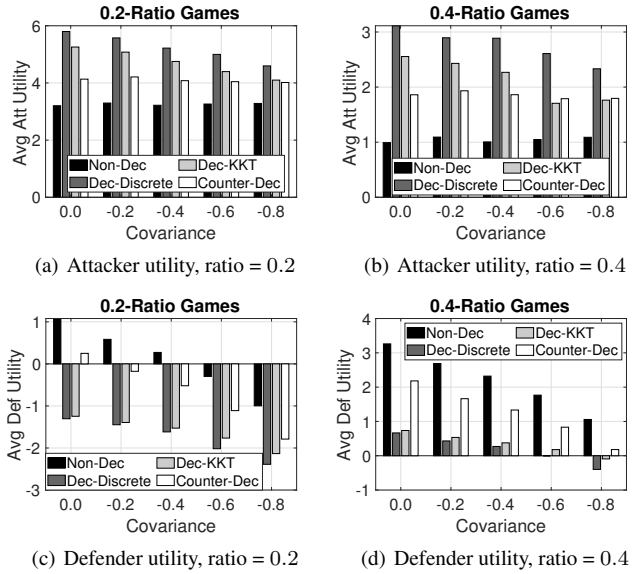
(c) Defender utility, 20 targets (d) Defender utility, 80 targets

Figure 2: Evaluation on solution quality, varying # of resources

that `Dec-Discrete` with a fine discretization of values for λ obtains a better approximation compared to `Dec-KKT` which only guarantees a lower bound for the attacker's optimal utility. The defender, on the other hand, suffers a significantly lower utility for not addressing the attacker's deception (`Non-Dec` versus `Dec-Discrete` and `Dec-KKT`). By taking the attacker's deception into account, `Counter-Dec` helps in drastically increasing the defender's utility. Finally, Figure 1 shows that the impact of deception and counter-deception does not qualitatively depend on the number of targets.

6.2 Varying number of defender resources

In our second set of experiments, we evaluate the solution quality of the examined algorithms when varying the number of security resources in the case of 20-target games and 80-target games. The results are shown in Figure 2. The x-axis represents the number of



(a) Attacker utility, ratio = 0.2 (b) Attacker utility, ratio = 0.4

(c) Defender utility, ratio = 0.2 (d) Defender utility, ratio = 0.4

Figure 3: Evaluation on solution quality, varying covariance value

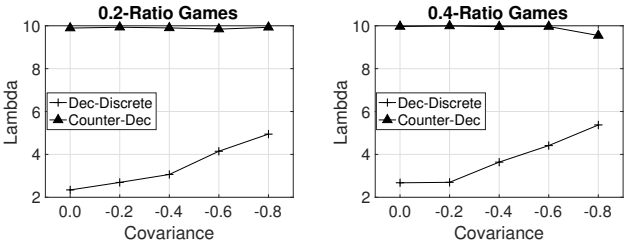
security resources in the game. Each data point is averaged over 100 games. In Figure 2, the decrease in attacker's average utility (regarding all four algorithms) when the number of defender resources increases is roughly linear. Similarly, the increase in the defender's average utility also exhibits a linearity property. This make senses as the defender's expected utility at each target is a linearly increasing function and the attacker's is a linearly decreasing function of the defender's coverage probability at that target. On the other hand, the relative loss in the defender's utility or the relative gain in utility of the attacker as a result of the attacker deception appears to be roughly constant when varying the number of defender resources.

6.3 Varying the covariance value

In our third set of experiments, we vary the covariance value. As noted, in zero-sum games, the attacker has no incentive to be deceptive. Therefore, we only plot the results of $r \in [-0.8, 0]$ with a step size of 0.2. The result is shown in Figure 3. Each data point is averaged over 100 games. Figure 3 shows that when the covariance value r gets closer to -1.0 (which implies to zero-sum games), the attacker's average utility (as result of its deception (`Dec-Discrete` and `Dec-KKT`) also gradually decreases, reflecting that the attacker has less incentive to play deceptively.

Furthermore, the defender's average utility in all cases (`Non-Dec`, `Dec-Discrete`, `Dec-KKT` and `Counter-Dec`) gradually decreases when r gets closer to -1.0. This results show that in SSGs, the defender's utility is always governed by the *adversarial* level (i.e., the payoff correlations) between the defender and the attacker, regardless of whether the attacker is deceptive or not.

Finally, we examine the deceptive value of λ for the attacker in two cases: (i) `Dec-Discrete` and (ii) `Counter-Dec`. The results are shown in Figure 4. Note that λ represents the attacker's rationality. The higher λ is, the more rational the attacker is. Figure 4 shows that when the defender does not take into account the attacker's deception, the deceptive attacker aims at mimicking a `QR` model with small value of λ. The deceptive value of λ increases when r decreases, implying the attacker is less deceptive when the game is closer to zero-sum. On the other hand, our counter-deception mechanism substantially diminishes the attacker's deception. The deceptive λ of the

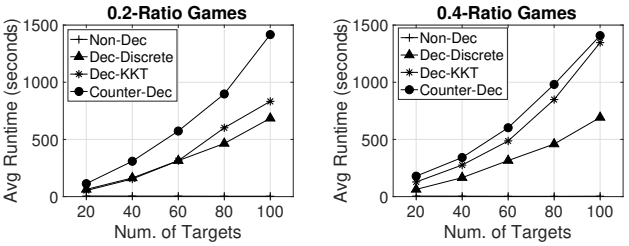


(a) Lambda, ratio = 0.2 (b) Lambda, ratio = 0.4

Figure 4: Analysis on `QR`'s parameter λ

attacker in this counter-deception case is close to 10 which is the upper bound we set for λ to avoid overflow.

6.4 Runtime performance



(a) Runtime, ratio = 0.2 (b) Runtime, ratio = 0.4

Figure 5: Runtime performance

Our evaluation on runtime performance is shown in Figure 5 in which the x-axis is the number of targets and the y-axis is the runtime on average. The runtime of our proposed algorithms gradually increases when the number of targets increases. For example, `Counter-Dec` reaches approximately 25 minutes when $N = 100$. This result shows that our proposed algorithms can scale up for large games.

7 Summary

This paper studies the attacker imitative deception in SSGs in the security scenario that the defender attempts to learn the attacker behavior based on historical attack data and the attacker is aware of the defender's learning. We introduce two new algorithms to compute an optimal deceptive strategy of the attacker: `Discretization-based` and `KKT-based` algorithms. Our empirical results shows that the attacker deception has a great impact in terms of providing a significant benefit for the attacker and loss for the defender. To handle the attacker's deception, we propose a novel counter deception algorithm which generate effective defense strategies with respect to learning outcomes of the attacker deceptive behavior. Through extensive experiments, we show that our proposed counter-deception algorithm helps in reducing drastically the impact the attacker's deception on both players' utility.

REFERENCES

[1] Battista Biggio and Fabio Roli, 'Wild patterns: Ten years after the rise of adversarial machine learning', *Pattern Recognition*, **84**, 317–331, (2018).
[2] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.
[3] Thomas E Carroll and Daniel Grosu, 'A game theoretic investigation of deception in network security', *Security and Communication Networks*, **4**(10), 1162–1172, (2011).
[4] D. Fraunholz, S. D. Anton, C. Lipps, D. Reti, D. Krohmer, F. Pohl, M. Tammen, and H. D. Schotten, 'Demystifying deception technology: A survey', *arXiv preprint arXiv:1804.06196*, (2018).
[5] Jiarui Gan, Haifeng Xu, Qingyu Guo, Long Tran-Thanh, Zinovi Rabinovich, and Michael Wooldridge, 'Imitative follower deception in stackelberg games', *arXiv preprint arXiv:1903.02917*, (2019).
[6] Shahrzad Gholami, Amulya Yadav, Long Tran-Thanh, Bistra Dilkina, and Milind Tambe, 'Don't put all your strategies in one basket: Playing green security games with imperfect prior knowledge', in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 395–403. International Foundation for Autonomous Agents and Multiagent Systems, (2019).
[7] Qingyu Guo, Bo An, Branislav Bosansky, and C. Kiekintveld, 'Comparing strategic secrecy and Stackelberg commitment in security games', in *26th International Joint Conference on Artificial Intelligence*, (2017).
[8] Karel Horák, Quanyan Zhu, and Branislav Bošanskỳ, 'Manipulating adversarys belief: A dynamic game approach to deception by design for proactive network security', in *International Conference on Decision and Game Theory for Security*, pp. 273–294. Springer, (2017).
[9] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar, 'Adversarial machine learning', in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pp. 43–58. ACM, (2011).
[10] Debarun Kar, Fei Fang, Francesco Delle Fave, Nicole Sintov, and Milind Tambe, 'A game of thrones: when human behavior models compete in repeated stackelberg security games', in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1381–1390. International Foundation for Autonomous Agents and Multiagent Systems, (2015).
[11] Debarun Kar, Thanh H Nguyen, Fei Fang, Matthew Brown, Arunesh Sinha, Milind Tambe, and Albert Xin Jiang, 'Trends and applications in stackelberg security games', *Handbook of Dynamic Game Theory*, 1–47, (2017).
[12] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe, 'Computing optimal randomized resource allocations for massive security games', in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 689–696. AAMAS, (2009).
[13] Daniel Lowd and Christopher Meek, 'Adversarial learning', in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 641–647. ACM, (2005).
[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, 'Towards deep learning models resistant to adversarial attacks', *arXiv preprint arXiv:1706.06083*, (2017).
[15] Daniel McFadden et al., 'Conditional logit analysis of qualitative choice behavior', (1973).
[16] Richard D McKelvey and Thomas R Palfrey, 'Quantal response equilibria for normal form games', *Games and economic behavior*, **10**(1), 6–38, (1995).
[17] Thanh H Nguyen, Arunesh Sinha, Shahrzad Gholami, Andrew Plumptre, Lucas Joppa, Milind Tambe, Margaret Driciru, Fred Wanyama, Aggrey Rwetsiba, Rob Critchlow, et al., 'Capture: A new predictive anti-poaching tool for wildlife protection', in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 767–775. International Foundation for Autonomous Agents and Multiagent Systems, (2016).
[18] Thanh H Nguyen, Yongzhao Wang, Arunesh Sinha, and Michael P Wellman, 'Deception in finitely repeated security games', in *33th AAAI Conference on Artificial Intelligence*, (2019).
[19] Thanh H Nguyen and Haifeng Xu, 'Imitative attacker deception in stackelberg security games', (2019).
[20] Thanh Hong Nguyen, Rong Yang, Amos Azaria, Sarit Kraus, and Milind Tambe, 'Analyzing the effectiveness of adversary modeling in security games', in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, (2013).
[21] Zinovi Rabinovich, Albert Xin Jiang, Manish Jain, and Haifeng Xu, 'Information disclosure as a means to security', pp. 645–653, (2015).
[22] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe, 'Stackelberg security games: Looking beyond a decade of success.', in *IJCAI*, pp. 5494–5501, (2018).
[23] Arunesh Sinha, Debarun Kar, and Milind Tambe, 'Learning adversary behavior in security games: A pac model perspective', in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 214–222. International Foundation for Autonomous Agents and Multiagent Systems, (2016).
[24] Milind Tambe, *Security and game theory: algorithms, deployed systems, lessons learned*, Cambridge university press, 2011.
[25] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell, 'Adversarial discriminative domain adaptation', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7167–7176, (2017).
[26] Haifeng Xu, Zinovi Rabinovich, Shaddin Dughmi, and Milind Tambe, 'Exploring information asymmetry in two-stage security games.', in *29th AAAI Conference on Artificial Intelligence*, pp. 1057–1063, (2015).
[27] Rong Yang, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Richard John, 'Improving resource allocation strategy against human adversaries in security games', in *Twenty-Second International Joint Conference on Artificial Intelligence*, (2011).
[28] Zhengyu Yin, Dmytro Korzhyk, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe, 'Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness', in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pp. 1139–1146. International Foundation for Autonomous Agents and Multiagent Systems, (2010).
[29] Jun Zhuang, Vicki M. Bier, and Oguzhan Alagoz, 'Modeling secrecy and deception in a multi-period attacker-defender signaling game', *European Journal of Operational Research*, **203**, 409–418, (2010).
[30] Jun Zhuang, Vicki M Bier, and Oguzhan Alagoz, 'Modeling secrecy and deception in a multiple-period attacker–defender signaling game', *European Journal of Operational Research*, **203**(2), 409–418, (2010).