Countering Attacker Data Manipulation in Security Games

Andrew R. Butler[1], Thanh H. Nguyen[1], and Arunesh Sinha[2]

[1] University of Oregon {arbutler, thanhhng}@cs.uoregon.edu
[2] Singapore Management University aruneshs@smu.edu.sg

Abstract. Defending against attackers with unknown behavior is an important area of research in security games. A well-established approach is to utilize historical attack data to create a behavioral model of the attacker. However, this presents a vulnerability: a clever attacker may change its own behavior during learning, leading to an inaccurate model and ineffective defender strategies. In this paper, we investigate how a wary defender can defend against such deceptive attacker. We provide four main contributions. First, we develop a new technique to estimate attacker true behavior despite data manipulation by the clever adversary. Second, we extend this technique to be viable even when the defender has access to a minimal amount of historical data. Third, we utilize a maximin approach to optimize the defender's strategy against the worst-case within the estimate uncertainty. Finally, we demonstrate the effectiveness of our counter-deception methods by performing extensive experiments, showing clear gain for the defender and loss for the deceptive attacker.

1 Introduction

Learning adversary behavior from historical attack data is a firmly established methodology in adversarial settings, both in academic literature [15,19], and in real world applications such as wildlife security [4,24]. Herein lies a vulnerability: a clever attacker may modify its own behavior in order to conceal information or mislead the defender. This deceptive behavior can influence the defender's learning process, creating future gainful opportunities for the attacker. Indeed, such deception has received considerable attention in security games literature [6,28,18]. However, robustness of the defender to the adversary's deceit is much less explored.

In this work, we investigate the defender's counteraction against attacker deception in a Stackelberg security game setting. Our work builds upon the *partial behavior deception* model [16] in which the defender models the behavior of the entire attacker population using a single Quantal Response (QR) [14] model of which the parameter $\lambda \in \mathbb{R}$ is learned from past attack data. Among the attackers, however, there is a rational attacker who can cause harm to the defender by manipulating part of attack data. Such manipulation makes the defender learn a wrong λ, leading to an ineffective defender strategy. Addressing the attacker deception is still an open problem, which is the focus of our paper.

As our *first contribution*, we develop a new technique to estimate the true behavior of the non-deceptive attackers (represented by a parameter value λ^{true} of QR), given the

perturbed training data. Our technique leverages the Karush-Kuhn-Tucker conditions of the rational attacker's optimization to formally express the relation between true behavior of non-deceptive attackers (λ^{true}) and learning outcome (λ^{learnt}) forced by the deceptive attacker. Based on this relation, we find that there is an interval of possible values for λ^{true} which leads to the same deception outcome λ^{learnt}. Moreover, bounds of this interval are increasing in λ^{learnt}. We thus propose a binary-search based method which uses λ^{learnt} to guide the search for these bounds within an ϵ-error.

As our *second contribution*, we extend our first contribution, perhaps surprisingly, to apply in scenarios with small number of attacks. The core issue is that the empirical attack distribution induced by limited attack samples may be far different from the true attack distribution induced by λ^{true}, making it challenging to characterize the relation between the true behavior and the deceptive outcome. We overcome this challenge by re-formulating the attack sampling process as choosing random *seeds* $\mathbf{u}$ drawn from the uniform distribution on $[0, 1]$ followed by a deterministic computation on $\mathbf{u}$.

We first prove that given any fixed $\mathbf{u}$, all mathematical results (from our first contribution) hold for small number of attacks. As the random seed chosen by nature is unknown, we then leverage the above result to perform binary search for *multiple* random seeds and construct a new interval spanning all found intervals as our final estimate for the range of λ^{true}.

As our *third contribution*, we propose a maximin approach to optimize the defender strategy against the worst case within the uncertainty interval for λ^{true}. We formulate this maximin problem as a multiple non-linear programs, each corresponds to a particular optimal attack choice of the deceptive attacker. *Finally*, via extensive experiments, we show that, even when optimizing against a wide uncertainty interval of λ^{true}, our algorithm gives significantly higher utility for the defender, and less benefit for the deceptive attacker.

2 Related Work

Adversarial Learning Adversarial learning is a field within machine learning that has become increasingly popular [12,23,9,13,29]. The attacker deception here is analogous to a *causative attack* (or poisoning attack) in adversarial learning [9]. A significant difference between our work and adversarial learning is that we seek to maximize defender utility *through* predicting the attacker's behavior, whereas in adversarial learning, the end goal is prediction accuracy.

Attacker Behavior Inference Learning the behavior of bounded rational attackers is crucial, and a major area of interest in security games. Various models including QR have been explored [27,10,28,22,20]. As this learning is used to create a defender strategy, the training attack pool is vulnerable to manipulation by a clever attacker. This paper focuses on addressing this challenge in security games. Our work overlaps with settings in which one or more players has limited information [1].

Deception in Security Games Historically, most work has focused on deception from the defender side [30,7]. In this scenario, the defender typically exploits information

asymmetry to fool the attacker (e.g. in network security, concealing some system characteristics). More recently, research has investigated deception from the attacker side [6,18,28] in SSGs, and the follower side in general Stackelberg games [5]. Much of this work concentrates on a single attacker whose payoff values are unknown to the defender. The attacker-deception model we utilize [16], on the other hand, describes a realistic scenario in which the defender must contend with multiple attackers of *unknown* behavior.

3 Preliminaries

3.1 Stackelberg Security Games (SSGs)

In SSGs [24], the *defender* must protect a set of T targets from one or more *attackers*. The defender has a limited number ($K < T$) of *resources* that each can be allocated to protect a single target. A pure strategy of the defender is defined as a one-to-one allocation of resources to targets. A mixed defense strategy, $\mathbf{x}$, is a probability distribution over these pure strategies. For the purposes of this paper, we consider no scheduling constraints to the defender's strategy, meaning that a mixed strategy can be compactly represented as a coverage probability vector, given by $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ where $x_i \in [0, 1]$ represents the probability that target i is protected by the defender and $\sum_i x_i \leq K$. We denote by $\mathbf{X}$ the set of all feasible defense strategies. In SSGs, the attacker is fully aware of the defender's mixed strategy and chooses a target to attack based on this knowledge.

An attack on target i gives each player a reward or a penalty, depending on whether the defender is currently protecting target i. If i is unprotected, the attacker gains reward R_i^a and the defender receives penalty P_i^d. Conversely, if target i is protected, the attacker takes penalty $P_i^a < R_i^a$ and the defender gains reward $R_i^d > P_i^d$. Given coverage probability x_i, the expected utilities for the defender and the attacker for an attack on target i can be formulated as follows:

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

Quantal Response Behavior Model (QR). QR is an well-known model describing attacker behavior in SSGs [14,27]. Intuitively, QR provides a mechanism by which higher expected utility targets are attacked more frequently. Essentially, the probability of attacking target i is given as follows:

$$q_i(\mathbf{x}; \lambda) = \left(e^{\lambda U_i^a(x_i)}\right) \Big/ \left(\sum\nolimits_j e^{\lambda U_j^a(x_j)}\right) \tag{1}$$

3.2 Partial Behavior Deception Model

Our work on developing an optimal counter-deception strategy for the defender is built upon the partial behavior deception model introduced by [16]. In this model, multiple attackers are present, who have the same payoffs but different attack behavior due

to different rationality levels. Among these attackers, there is a rational attacker who intends to play deceptively to mislead the defender. The defender, on the other hand, is aware of the attackers' payoffs but is uncertain about the behavior of the attackers. The defender thus attempts to to build a behavior model, i.e., the QR model, to predict the attack distribution of the entire attacker population. Real-world applications such as wildlife conservation also use this single-behavior-modeling approach as park rangers usually cannot differentiate data collected, such as poaching signs, among multiple sources [10].

Two-phase learning-planning of defender. This model describes a *one-shot two-phase learning-planning* problem for the defender, consisting of a learning phase and a planning phase. This is the typical security game model used in literature [24,27]. Essentially, in the learning phase, the defender uses training attack data to estimate the parameter λ of QR using the Maximum Likelihood Estimation method (MLE), as formulated below:

$$\lambda^{\text{learnt}} \in \operatorname*{argmax}_{\lambda} \sum_{m} \sum_{i} z_i^m \log q_i(\mathbf{x}^m; \lambda) \tag{2}$$

where x_i^m is the defender's coverage probability at target i and step m and z_i^m is the corresponding number of attacks.

During the planning phase, the defender utilizes the learned λ^{learnt} value to optimize his defense against such an attacker. The optimal strategy, $\mathbf{x}^*$, is given by:

$$\mathbf{x}^* \in \operatorname*{argmax}_{\mathbf{x} \in \mathbf{X}} \sum_{i} q_i(\mathbf{x}; \lambda^{\text{learnt}}) U_i^d(x_i) \tag{3}$$

Behavior deception of attacker. [16] Since the (naive) defender uses the entire learning dataset to construct a single attacker model, a clever attacker might change its own behavior during the learning phase in order to benefit during the planning phase[3]. It is naturally assumed that only perfectly rational attackers display such deceptive behavior. Therefore, the partial behavior deception model centers on a single perfectly rational deceptive attacker, amongst the bounded rational attackers, that can alter some fraction of the training dataset. The bounded rational attackers attack non-deceptively according to a fixed unknown QR parameter λ^{true}. Essentially, the deceptive attacker wants to find the best perturbation of the training data to maximize its utility in the planning phase,

[3] In this paper, we focus on the one-shot game which only consists of a learning phase and planning phase—a commonly-used security game model in literature. Therefore, the deceptive attacker can simply play perfectly rationally in the planning phase after deceiving the defender in the learning phase. This model can also serve as the basis for repeated security games which involve multiple learning-planning rounds where the attacker plays deceptively in all rounds except the last round.

denoted by $U^a(\mathbf{x}^*(\mathbf{z}))$, as follows:

$$(\texttt{DecAlter}): \max_{\mathbf{z}=\{z_i^m\}} U^a(\mathbf{x}^*(\mathbf{z})) \tag{4}$$

$$\text{s.t. } z_i^m \geq n_i^m, \forall m, i \tag{5}$$

$$\sum_i z_i^m \leq (f+1) \cdot \sum_i n_i^m, \forall m. \tag{6}$$

where $\mathbf{x}^*(\mathbf{z})$ is the defender's strategy determined based on his learning-planning method in (2–3). In addition, n_i^m is the number of attacks by the non-deceptive attackers and $f \in \mathbb{R}$ is the ratio of deceptive attacks to non-deceptive attacks at each step m. Constraints (5–6) guarantee that the deceptive attacker can only control its own attacks. We denote by $\mathbf{z} = \{z_i^m\}$ the deception outcome of the deceptive attacker, which includes the non-deceptive attacks ($\mathbf{n} = \{n_i^m\}$). The defender learns a (deceptive) parameter λ^{learnt} using $\mathbf{z}$.

3.3 Cognitive Hierarchy Approach

In order to determine a counter-deception strategy for the defender, a possible approach is to compute a fixed point equilibrium of the deception game in which each player reasons about its opponent's strategy recursively till infinity. However, finding a fixed point equilibrium in our game is extremely challenging. This is because the defender has no information (or prior) about the behavior of the non-deceptive attackers. As a result, the defender has to relate the equilibrium outcome for every possible true behavior of these non-deceptive attackers to the observed (manipulated) attacks. This task is challenging (as well as impractical) given that the behavior space of attackers is infinite.

In real world settings, cognitive hierarchy models have been proven more effective than equilibrium based approaches at realistically modeling player behavior [3,2,8]. This is because human players do not exhibit infinite level strategic reasoning. Cognitive hierarchy theory states that players in games can be divided into different *levels* of thinkers, each assuming that no players are on levels above them [26]. In a mixed attacker deception setting, we can model the levels as follows:

- Level 1: The rational attacker plays truthfully. The defender follows the two-stage learning-planning approach to compute a defense strategy.
- Level 2: The rational attacker plays deceptively, assuming the defender is at level 1. The level 2 defender, on the other hand, attempts to counter the attacker deception, assuming the attackers are at levels 0, 1, or 2.
- Level $l > 2$: The strategic reasoning is similar to level 2. Specifically, the attacker assumes the defender is at level $l-1$ while the defender assumes the attackers are at any one of the levels *up to and including* l.

Previous work has shown that distributions of human players in normal form games mostly consist of lower level players [26]. The aforementioned partial behavior deception model focuses on the deception by a level 2 attacker [16]. Our paper studies the counter-deception by a level 2 defender.

4 Finding Non-Deceptive Attacker Behavior

In order to determine an effective defense strategy, we begin our analysis by characterizing the space of *possible* attack behavior (described by QR) of the non-deceptive attackers, given the perturbed data $\mathbf{z}$. Recall that the non-deceptive attackers respond according to a fixed λ^{true}, unknown to the defender. Instead, the defender obtains a learning outcome λ^{learnt} given perturbed training data. Our goal is to estimate the possible values of λ^{true} given observed learning outcome λ^{learnt}.

4.1 Characterizing Deceptive Attacker's Behavior

We first analyze the deception possibilities for the deceptive attacker *given any value* λ^{true} of the non-deceptive attackers. The results we establish here help us in our goal of estimating λ^{true}. For analysis sake, we assume that the number of attacks is large enough such that the sampled attacks is close to the actual attack probability distributions. We will relax this assumption later. Mathematically, we assume:

$$\big(n_i^m\big)/\big(\sum_j n_j^m\big) \approx q_i^m(\mathbf{x}^m, \lambda^{\text{true}}), \forall m \tag{7}$$

where n_i^m refers to the number of attacks committed by the *non-deceptive* attacker at target i. As shown in (DecAlter), the objective utility function of the deceptive attacker depends on the strategy of the defender, which in turn is governed by the training data $\{z_i^m\}$, and the training data contains attacks by the non-deceptive attacker too ($\{n_i^m\}$). Thus, the outcome of λ^{learnt} depends on the behavior of the non-deceptive attacker λ^{true} (or $\{n_i^m\}$). We thus also use the notion $\texttt{DecAlter}(\lambda^{\text{true}}) = \lambda^{\text{learnt}}$ to represent the dependence of the learning result (*altered* by deception) on λ^{true}.

For this portion of our analysis, we relax the domain of $\mathbf{z}$ to be continuous. This allows our proofs to be simpler and more concise. In practice, this value is limited to discrete integers; fractional attacks are nonsensical. Later, we will extend the methods to the discrete $\mathbf{z}$ case, and show why they still apply. We exploit the KKT condition for the optimality of the deceptive λ^{learnt} as the outcome of the defender's learning, formulated in optimization (2). Essentially, λ^{learnt} has to satisfy the following KKT condition:

$$\sum_m \Big[\sum_i z_i^m\Big]\Big[\frac{\sum_i z_i^m U_i^a(x_i^m)}{\sum_i z_i^m} - \underbrace{\sum_i q_i(\mathbf{x}^m; \lambda^{\text{learnt}}) U_i^a(x_i^m)}_{\text{Attacker utility } U^a(\mathbf{x}^m;\, \lambda^{\text{learnt}})}\Big] = 0$$

where $U^a(\mathbf{x}^m; \lambda^{\text{learnt}})$ is the attacker's expected utility when the defender plays $\mathbf{x}^m$ and the attacker plays according to λ^{learnt}. In our theoretical analysis, we leverage the following important monotonicity property of this utility function:

Observation 1 ([17]). *$U^a(\mathbf{x}^m, \lambda)$ is an increasing function of λ for any given strategy $\mathbf{x}^m$.*

Let's assume, WLOG, the attacker's utilities at each target has the following order: $U_1^a(x_1^m) \le U_2^a(x_2^m) \le \ldots \le U_T^a(x_T^m)$ for all m. Observation 1 aids us in showing that all

feasible (not necessarily optimal) deceptive λ values form an interval $[\lambda^{\text{learnt}}_{\min}, \lambda^{\text{learnt}}_{\max}]$ with $\lambda^{\text{learnt}}_{\min}$, $\lambda^{\text{learnt}}_{\max}$ specified as follows:

Theorem 1 (Characterization of Deception Space). *Given λ^{true} and the attack ratio f, the space of deceptive parameters inducible by the deceptive attacker forms an interval $[\lambda^{\text{learnt}}_{\min}, \lambda^{\text{learnt}}_{\max}]$, where $\lambda^{\text{learnt}}_{\max}$ is the unique solution of:*

$$\sum_{m,j} n^m_j [U^a(\mathbf{x}^m; \lambda^{\text{true}}) + f U^a_T(x^m_T) - (f+1) U^a(\mathbf{x}^m, \lambda^{\text{learnt}}_{\max})] = 0$$

and $\lambda^{\text{learnt}}_{\min}$ is the unique solution of:

$$\sum_{m,j} n^m_j [U^a(\mathbf{x}^m; \lambda^{\text{true}}) + f U^a_1(x^m_1) - (f+1) U^a(\mathbf{x}^m, \lambda^{\text{learnt}}_{\min})] = 0$$

All formal proofs are in the appendix. Essentially, Theorem 1 states that given some true behavior of the non-deceptive attacker λ^{true}, the deceptive attacker can force the deceptive λ to be any value in $[\lambda^{\text{learnt}}_{\min}, \lambda^{\text{learnt}}_{\max}]$. Further, the deceptive attacker cannot make the defender learn any λ outside of this range. Based on Theorem 1, we present the following corollaries which characterize the monotonicity of $\lambda^{\text{learnt}}_{\min}$ and $\lambda^{\text{learnt}}_{\max}$, as well as the monotonicity of the optimal deception $\lambda^{\text{learnt}} = \texttt{DecAlter}(\lambda^{\text{true}}) \in [\lambda^{\text{learnt}}_{\min}, \lambda^{\text{learnt}}_{\max}]$ with respect to the non-deceptive attacker behavior λ^{true}.

Corollary 1. *Consider two different behavior parameters, $\lambda^{\text{true}}_1 \le \lambda^{\text{true}}_2$. Denote by $[\lambda^{\text{learnt}}_{\min,1}, \lambda^{\text{learnt}}_{\max,1}]$ and $[\lambda^{\text{learnt}}_{\min,2}, \lambda^{\text{learnt}}_{\max,2}]$ the corresponding deceptive parameter ranges, we have: $\lambda^{\text{learnt}}_{\max,1} \le \lambda^{\text{learnt}}_{\max,2}$ and $\lambda^{\text{learnt}}_{\min,1} \le \lambda^{\text{learnt}}_{\min,2}$.*

Based on Corollary 1, we obtain Corollary 2 showing the monotonicity relation between λ^{learnt} and λ^{true}.

Corollary 2. *Consider two different behavior parameters, $\lambda^{\text{true}}_1 \ne \lambda^{\text{true}}_2$. Then, we have:*

$$\lambda^{\text{true}}_1 \le \lambda^{\text{true}}_2 \Longrightarrow \texttt{DecAlter}(\lambda^{\text{true}}_1) \le \texttt{DecAlter}(\lambda^{\text{true}}_2) \tag{8}$$

$$\texttt{DecAlter}(\lambda^{\text{true}}_1) < \texttt{DecAlter}(\lambda^{\text{true}}_2) \Longrightarrow \lambda^{\text{true}}_1 < \lambda^{\text{true}}_2 \tag{9}$$

Corollary 3. *Consider two different behavior parameters $\lambda^{\text{true}}_1 \le \lambda^{\text{true}}_2$. If the corresponding optimal deception solutions: $\texttt{DecAlter}(\lambda^{\text{true}}_1) = \texttt{DecAlter}(\lambda^{\text{true}}_2)$, then for any $\lambda^{\text{true}} \in [\lambda^{\text{true}}_1, \lambda^{\text{true}}_2]$, we also have its optimal deception solution: $\texttt{DecAlter}(\lambda^{\text{true}}) = \texttt{DecAlter}(\lambda^{\text{true}}_1)$.*

4.2 RaBiS: Characterizing Behavior of Non-Deceptive Attacker

In this section, we attempt to find the range of possible values for λ^{true}, which is unknown to the defender, as only the deceptively altered QR parameter λ^{learnt} is observed. We leverage the results of Corollaries 2 and 3 for this analysis.

Lemma 1. *Given some learned λ^{learnt}, there exists an interval $[\lambda^{\text{true}}_{\min}, \lambda^{\text{true}}_{\max}]$ such that all values $\lambda^{\text{true}} \in [\lambda^{\text{true}}_{\min}, \lambda^{\text{true}}_{\max}]$ leads to the same outcome λ^{learnt}. In addition, both bounds $\lambda^{\text{true}}_{\min}$ and $\lambda^{\text{true}}_{\max}$ are increasing in λ^{learnt}.*

Based on the above result, we propose a binary-search based approach, RaBiS (**Ra**nge-finding **Bi**nary **S**earch), to find the interval $[\lambda^{\text{true}}_{\min}, \lambda^{\text{true}}_{\max}]$ within an ϵ-error in a polynomial time for arbitrary small $\epsilon > 0$. RaBiS consists of two binary searches: the first binary search is to find the upper bound $\lambda^{\text{true}}_{\max}$ and the second binary search is to find the lower bound $\lambda^{\text{true}}_{\min}$. Both binary searches maintain a pair of bounds for binary search (lb, ub). While in theory the range of λ^{true} is $[0, \infty)$, in practice, a limited range of $[0, M]$, where M is a very large constant, ensures that the attacker's QR behavior with $\lambda^{\text{true}} = M$ is close enough to $\lambda^{\text{true}} = \infty$. Therefore, in our algorithm, we initialize $lb = 0$ and $ub = M$.

At each iteration, we examine the mid-value $r = (lb + ub)/2$ by comparing the deception calculation $\lambda' = \texttt{DecAlter}(r)$ with the actual deception outcome computed by the defender, λ^{learnt}. In particular, in the binary search for finding $\lambda^{\text{true}}_{\max}$, if $\lambda' \leq \lambda^{\text{learnt}}$, there must be a $\lambda^{\text{true}}_{\max} \in [r, ub]$ such that $\texttt{DecAlter}(\lambda^{\text{true}}_{\max}) = \lambda^{\text{learnt}}$ and any $\lambda > \lambda^{\text{true}}$ implies $\texttt{DecAlter}(\lambda) > \lambda^{\text{learnt}}$. Thus, in order to find $\lambda^{\text{true}}_{\max}$, we update the lower bound $lb = r$. Conversely, if $\lambda' > \lambda^{\text{learnt}}$, it means all $\lambda^{\text{true}} \in [r, ub]$ will lead to a deceptive parameter value strictly greater than λ^{learnt}. Therefore, we update the upper bound $ub = r$. This process stops when $ub - lb < \epsilon$. The binary search process for finding $\lambda^{\text{true}}_{\min}$ is similar.

4.3 Principled Approach for Low-Data Challenge

Thus far, our analysis of the range of the non-deceptive attacker λ^{true} was performed under the approximation assumption of Equation 7. However, in practice, this assumption may not hold true. This is because the attacker may conduct a limited number of attacks, which leads to a substantial difference between the empirical attack distribution and the true attack distribution, that is:

$$\big(n_i^m\big)\big/\big(\sum_j n_j^m\big) \neq q_i^m(\mathbf{x}^m; \lambda^{\text{true}}), \forall m$$

To address this challenge, we first investigate the generation of limited attack samples from the true distribution under a *static random seed*. We show that our previous theoretical results for the ideal scenario still hold in this "limited-attack" scenario. We then leverage this result for a static random seed to address the general case of *unknown* random seed.

Sampling by transformation. Sample generation from certain parameterized distributions can be split into a two step process by using a transformation of known distributions [21,11]. We show that such split generation is possible for our problem. Let u be a real valued random variable that is distributed uniformly between 0 and 1. Given a defense strategy, $\mathbf{x}^m$, and QR parameter λ, we define the function f_λ such that $P\big(f_\lambda(u) = i\big) = q_i(\mathbf{x}^m; \lambda)$. Note that f_λ is a deterministic function dependent on λ, which we define explicitly next. For any given $\mathbf{x}^m$, partition the interval $[0, 1]$ according to the attack probabilities $q_i(x^m; \lambda)$ specified by QR with parameter λ, with the following partition boundary points: $S(0; \lambda) = 0$, $S(i; \lambda) = \sum_{j=1}^{i} q_j(\mathbf{x}^m; \lambda)$, and $S(T; \lambda) = 1$. Figure 1 is an example when the number of targets is $T = 3$. Given this

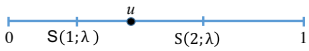


Fig. 1: Attack generation by transforming uniform dist.

division, we define $f_\lambda(u) = i$ when $u \in [S(i-1;\lambda), S(i;\lambda)]$; it can be readily verified that $P\big(f_\lambda(u) = i\big) = q_i(\mathbf{x}^m;\lambda)$. In the case of $N > 1$ attacks, we can view the attack generation process as N samples of u to get $\mathbf{u} = \{u_1, \ldots, u_N\}$ and then applying f_λ to each of those samples to obtain the targets attacked.

Static random seed generation. For our problem with parameter λ^{true}, after separating the randomness (u) and the effect of the parameter ($f_{\lambda^{\text{true}}}$) in attack generation, the main idea of a static random seed is to assume that the N uniformly sampled values $\mathbf{u}$ are the same for any value of λ^{true} that we consider in the binary search for $\lambda^{\text{true}}_{\min}$ or $\lambda^{\text{true}}_{\max}$. By controlling the randomness, we establish a deterministic baseline to compare the empirical distribution arising from the different λ^{true} that we consider. A big advantage of controlling randomness is that it allows us to carry over all the previous proofs to a low data setting, as described next.

Let $E(\mathbf{u}, \lambda^{\text{true}})$ be the empirical distribution when attacks are computed using $f_{\lambda^{\text{true}}}$ and the generated N samples $\mathbf{u}$. We can define the attacker expected utility w.r.t. this distribution, denoted by $U^a(\mathbf{x}^m; E(\mathbf{u}, \lambda^{\text{true}}))$, exactly analogously to how $U^a(\mathbf{x}^m; \lambda^{\text{true}})$ is defined w.r.t. the true distribution. We obtain Lemma 2 which is analogous to Observation 1.

Lemma 2. *For a fixed seed, $\mathbf{u}$, the attacker expected utility computed based on the corresponding empirical distribution, $U^a(\mathbf{x}^m; E(\mathbf{u}, \lambda^{\text{true}}))$, is an increasing function of λ^{true}.*

In all results previously (including corollaries), we only used the Observation 1 property of $U^a(\mathbf{x}^m; \lambda^{\text{true}})$. With the result above, we can replace $U^a(\mathbf{x}^m; \lambda^{\text{true}})$ by $U^a(\mathbf{x}^m; E(\mathbf{u}, \lambda^{\text{true}}))$ and all proofs still go through. Hence, our Theorem 1 holds with respect to $U^a(\mathbf{x}^m; E(\mathbf{u}, \lambda^{\text{true}}))$ (which replaces $U^a(\mathbf{x}^m; \lambda^{\text{true}})$ in the equations presented in Theorem 1). This result shows that for a fixed random seed $\mathbf{u}$ we can recover all previous results.

Extension to unknown random seed. The random seed used (by nature) in the generation of the training data is not known to the defender. To overcome this challenge, we extend our binary search to consider multiple random seeds. For each random seed, we run `RaBiS` to obtain an interval of possible values for λ^{true}. Taking a worst-case approach, we consider the smallest interval that spans all of these ranges as the uncertainty set containing all possible values of λ^{true}.

5 Maximin to Optimize Defender Utility

After finding the range $[\lambda^{\text{true}}_{\min}, \lambda^{\text{true}}_{\max}]$, the defender must optimize its strategy accordingly. Essentially, the defender is aware that there are attacks not only from a rational

(deceptive) attacker (who will act optimally in the defender's planning phase) but also from bounded rational attackers (whose λ^{true} can be any value within $[\lambda^{\text{true}}_{\text{min}}, \lambda^{\text{true}}_{\text{max}}]$). In order to overcome the uncertainty about the behavior of these attackers, we take a maximin approach where the defender seeks to *maximize* its utility against the *worst* possible (for the defender) λ value within the calculated range. In practice, to deal with the computational challenge due to an infinite number of possible values in $[\lambda^{\text{true}}_{\text{min}}, \lambda^{\text{true}}_{\text{max}}]$, we break down this range into a set of possible discrete values $\{\lambda^{\text{true}}_{\text{min}}, \lambda^1, \lambda^2, \ldots, \lambda^{\text{true}}_{\text{max}}\}$. Furthermore, since the rational attacker will choose an optimal target to attack in the planning phase, we decompose our defense problem into multiple non-linear programs, each corresponds to a particular optimal target to attacker for the rational attacker. In particular, our non-linear program corresponding to an optimal target j can be formulated as follows:

$$\max_{\mathbf{x}} f \cdot U^d_j(x_j) + U^d_{\text{worst-case}} \tag{10}$$

$$\text{s.t. } U^a_j(x_j) \geq U^a_i(x_i), \forall i \tag{11}$$

$$U^d_{\text{worst-case}} \leq \sum\nolimits_i q_i(\mathbf{x}; \lambda) U^d_i(x_i), \tag{12}$$

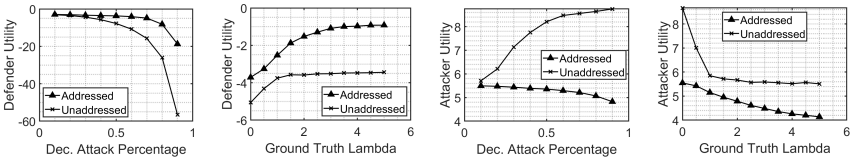
$$\forall \lambda \in \{\lambda^{\text{true}}_{\text{min}}, \lambda^1, \lambda^2, \ldots, \lambda^{\text{true}}_{\text{max}}\}$$

$$\sum\nolimits_i x_i \leq K, x_i \in [0, 1], \forall i \tag{13}$$

The objective (line 10) balances optimization against the fully rational attacker, $U^d_j(x_j)$, and the worst possible bounded rational attacker, $U^d_{\text{worst-case}}$, with multiplier f corresponding to the ratio of deceptive to non-deceptive attacks. Constraint (11) ensures that the target chosen by the fully rational attacker, j, is indeed the highest-utility target. Constraint (12) effectively iterates through the λ range, setting $U^d_{\text{worst-case}}$ equal to the lowest defender utility value among all possible lambdas. In a zero sum game, these lines could be replaced by simply setting $\lambda = \lambda^{\text{true}}_{\text{max}}$. Lastly, constraint (13) provides logical bounds to the defender's strategy: the total coverage percentage of all targets cannot exceed the number of resources, and all targets have coverage probability between 0 and 1.

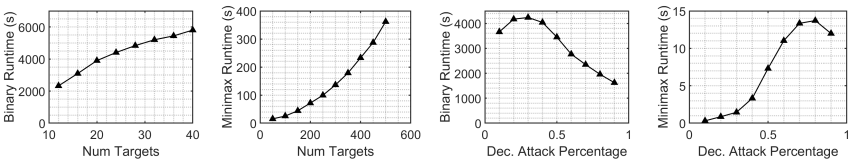
6 Experiments

In our experiments, we analyze: (i) the defender's utility gain by addressing deception, and (ii) the loss of utility for the devious attacker. The training data includes attacks from both the fully rational deceptive attacker and a boundedly rational attacker whose behavior is described by `QR`. We use 5 defender training strategies ($M = 5$) each with 50 non-deceptive attacks ($\sum_i n^m_i = 50$) sampled from the `QR` distribution with λ^{true} of the bounded rational attacker. Each data point is averaged over 200+ games, generated using GAMUT (http://gamut.stanford.edu). For our trials, we vary (i) the true non-deceptive lambda λ^{true} value and (ii) the fraction f of attacks done by the devious adversary. Due to limited space, we will only highlight important results. Additional results are included in our appendix. All utility results are statistically significant under bootstrap-t (α=0.05) [25].



(a) Vary % of dec. attacks (b) Vary λ^{true} (c) Vary % of dec. attacks (d) Vary λ^{true}

Fig. 2: Players Utility Evaluation



(a) Binary Search Runtime (b) Maximin Runtime (c) Binary Search Runtime (d) Maximin Runtime

Fig. 3: Runtime Evaluation

Figures 2a and 2b display the defender's utility in two cases: (i) `Addressed` — the defender addresses the attacker's deception using our counter-deception algorithm; and (ii) `Unaddressed` — the defender simply does not take the attacker's deception into account. In these two figures, the y-axis represents the defender's expected utility on average. Both figures show that the defender can significantly increase his utility for playing our maximin counter-deception strategy. In Figure 2a we observe that, when deception is unaddressed, the defender's utility decreases exponentially as the deceptive attack ratio increases. On the other hand, when the defender *does* address deception, the slope is far more gradual. Figure 2b shows how defender utility increases as the non-deceptive λ^{true} value does. This effect tapers off on the upper end of the spectrum. This result is expected because the non-deceptive attacker gets more rational as λ^{true} increases, leading to less changes in the defender's maximin strategy. Furthermore, in Figure 2b, the lowest utility point for the defender is when λ^{true} gets to zero. This makes sense: as the non-deceptive attackers become completely non-strategic ($i.e.$, $\lambda^{\text{true}} = 0$), the non-deceptive attackers will have less influence on the training data, or equivalently, the deceptive attacker has more power to manipulate the data.

Naturally, we observe an opposite trend in the attacker-utility graphs shown in Figures 2c and 2d. That is, the utility of the attacker reduces substantially when the defender addresses the attacker deception. Figure 2c shows that when the defender plays our maximin strategy, the attacker's utility actually decreases w.r.t. the percentage of attacks controlled by the deceptive attacker. This result appears to be counter-intuitive at first glance. However, it's logical: our maximin algorithm knows the attack ratio so it tailors more of the defense strategy towards a fully rational attacker (the actual rationality of the deceptive attacker).

Lastly, we analyze runtime performance of both portions of the algorithm in Figure 3. For the binary search, runtime is high across the board due to the sheer number of partial deception games (`DecAlter`) solved in each search. However, this runtime

scales linearly w.r.t. the number of targets (Figure 3a), implying that the algorithm can be scaled to large games. Furthermore, when varying the attack percentage (Figure 3c), we see that the runtime peaks with a percentage around 0.3. This peak is shifted compared to the runtime for solving (`DecAlter`) only, which peaks around 0.5 [16]. This is because the range, $[\lambda_{\min}^{\text{true}}, \lambda_{\max}^{\text{true}}]$ increases as the deceptive attack percentage does, meaning the total search time decreases as `RaBiS` exits earlier.

Figure 3b shows how the maximin runtime increases w.r.t. the number of targets. This is expected since the number of non-linear programs involved is equal to the number of targets. The maximin optimization can scale to large games: 500 target games are solved in less than 10 minutes. Observe that we examine a larger spread of targets here than for the binary search portion of the algorithm; the binary search runtime is orders of magnitude higher, reaching our 100 minute cut-off with far fewer targets. Figure 3d shows that maximin runtime initially increases as the percentage of attacks that are deceptive does, reflecting the wider range of possible values for λ^{true}. At higher values this effect diminishes and runtime ends up decreasing at the 0.9 marker, indicating that it is easier to optimize a strategy against mostly rational attacks.

7 Conclusion

We successfully addressed attacker deception in security games, showing both theoretically and experimentally the value of our approach. Through mathematical analysis we explored the characteristics of deception and defense and developed effective countermeasures: `RaBiS` allowed the defender to see through the deceptively altered historical attack data, after which a maximin approach yielded a robust strategy. Our experiments showed the wary defender receiving much higher utility than its naive counterpart.

Acknowledgement: This work was supported by ARO grant W911NF-20-1-0344 from the US Army Research Office.

References

1. Albarran, S.E., Clempner, J.B.: A stackelberg security markov game based on partial information for strategic decision making against unexpected attacks. Engineering Applications of Artificial Intelligence **81**, 408 – 419 (2019). https://doi.org/https://doi.org/10.1016/j.engappai.2019.03.010, http://www.sciencedirect.com/science/article/pii/S0952197619300600
2. Brown, A.L., Camerer, C.F., Lovallo, D.: To review or not to review? limited strategic thinking at the movie box office. American Economic Journal: Microeconomics **4**(2), 1–26 (May 2012). https://doi.org/10.1257/mic.4.2.1, https://www.aeaweb.org/articles?id=10.1257/mic.4.2.1
3. Camerer, C.F., Ho, T.H., Chong, J.K.: A Cognitive Hierarchy Model of Games*. The Quarterly Journal of Economics **119**(3), 861–898 (08 2004). https://doi.org/10.1162/0033553041502225, https://doi.org/10.1162/0033553041502225
4. Fang, F., Nguyen, T.H., Pickles, R., Lam, W.Y., Clements, G.R., An, B., Singh, A., Tambe, M., Lemieux, A.: Deploying paws: Field optimization of the protection assistant for wildlife security. In: IAAI-16 (2016)

5. Gan, J., Guo, Q., Tran-Thanh, L., An, B., Wooldridge, M.: Manipulating a learning defender and ways to counteract. In: NIPS-19 (2019)
6. Gan, J., Xu, H., Guo, Q., Tran-Thanh, L., Rabinovich, Z., Wooldridge, M.: Imitative follower deception in stackelberg games. In: EC '19 (2019)
7. Guo, Q., An, B., Bosansky, B., Kiekintveld, C.: Comparing strategic secrecy and Stackelberg commitment in security games. In: IJCAI (2017)
8. Hortaçsu, A., Luco, F., Puller, S.L., Zhu, D.: Does strategic ability affect efficiency? evidence from electricity markets. AER **109**(12), 4302–42 (December 2019). https://doi.org/10.1257/aer.20172015, https://www.aeaweb.org/articles?id=10.1257/aer.20172015
9. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.D.: Adversarial machine learning. In: AISec (2011)
10. Kar, D., Ford, B., Gholami, S., Fang, F., Plumptre, A., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Nsubaga, M.: Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In: AAMAS '17 (2017)
11. Kingma, D.P.: Auto-encoding variational bayes. In: ICLR (2014)
12. Lowd, D., Meek, C.: Adversarial learning. In: ACM SIGKDD (2005)
13. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks (2017)
14. McKelvey, R.D., Palfrey, T.R.: Quantal response equilibria for normal form games. In: Games and economic behavior (1995)
15. Nguyen, T.H., Sinha, A., Gholami, S., Plumptre, A., Joppa, L., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Critchlow, R., et al.: Capture: A new predictive anti-poaching tool for wildlife protection. In: AAMAS '16. pp. 767–775 (2016)
16. Nguyen, T.H., Sinha, A., He, H.: Partial adversarial behavior deception in security games. In: IJCAI (2020)
17. Nguyen, T.H., Vu, N., Yadav, A., Nguyen, U.: Decoding the imitation security game: Handling attacker imitative behavior deception. In: 24th European Conference on Artificial Intelligence (2020)
18. Nguyen, T.H., Wang, Y., Sinha, A., Wellman, M.P.: Deception in finitely repeated security games. In: AAAI-19 (2019)
19. Peng, B., Shen, W., Tang, P., Zuo, S.: Learning optimal strategies to commit to. In: 33th AAAI Conference on Artificial Intelligence (2019)
20. Perrault, A., Wilder, B., Ewing, E., Mate, A., Dilkina, B., Tambe, M.: Decision-focused learning of adversary behavior in security games. CoRR **abs/1903.00958** (2019), http://arxiv.org/abs/1903.00958
21. Price, R.: A useful theorem for nonlinear devices having gaussian inputs. IEEE Trans. Inf. Theory **4** (1958)
22. Sinha, A., Kar, D., Tambe, M.: Learning adversary behavior in security games: A pac model perspective. In: AAMAS '16 (2016)
23. Song, Y., Ma, C., Wu, X., Gong, L., Bao, L., Zuo, W., Shen, C., Lau, R.W., Yang, M.H.: Vital: Visual tracking via adversarial learning. In: IEEE CVPR (2018)
24. Tambe, M.: Security and game theory: algorithms, deployed systems, lessons learned. Cambridge University Press (2011)
25. Wilcox, R.: Applying contemporary statistical techniques. Academic Press (2002)
26. Wright, J.R., Leyton-Brown, K.: Level-0 meta-models for predicting human behavior in games. In: EC '14. ACM (2014). https://doi.org/10.1145/2600057.2602907, https://doi.org/10.1145/2600057.2602907
27. Yang, R., Kiekintveld, C., Ordonez, F., Tambe, M., John, R.: Improving resource allocation strategy against human adversaries in security games. In: IJCAI (2011)

28. Zhang, J., Wang, Y., Zhuang, J.: Modeling multi-target defender-attacker games with quantal response attack strategies. Reliability Engineering & System Safety **205** (2021)
29. Zhang, X., Zhu, X., Lessard, L.: Online data poisoning attack (2019)
30. Zhuang, J., Bier, V.M., Alagoz, O.: Modeling secrecy and deception in a multi-period attacker-defender signaling game. European Journal of Operational Research **203**, 409–418 (2010)

Appendix

7.1 Proof of Theorem 1

In order to prove this theorem, we introduce a series of lemmas (3–6). For the sake of analysis, we denote by:

$$y_i^m = \frac{z_i^m}{\sum_j z_j^m} \qquad\qquad c^m = \frac{1}{\sum_j z_j^m}$$

Intuitively, y_i^m is the empirical attack distribution estimated from the perturbed training data $\hat{\mathcal{D}} = \{x_i^m, z_i^m\}$ and c^m is the normalization term. Also, $\{y_i^m, c^m\}$ and $\{z_i^m\}$ are interchangeable. That is, given $\{y_i^m, c^m\}$, we can determine $z_i^m = \frac{y_i^m}{c^m}$. We first present the Lemma 1 which determines the deception capability of the deceptive attacker:

Lemma 3. *Given the true behavior λ^{true} of the non-deceptive attackers and the attack ratio f, the deceptive space for the deceptive attacker is specified as follows:*

$$\sum\nolimits_m \frac{1}{c^m}\left[\sum\nolimits_i y_i^m U_i^a(x_i^m) - U^a(\mathbf{x}^m, \lambda)\right] = 0 \tag{14}$$

$$\frac{y_i^m}{c^m} \geq n_i^m, \forall m, i \tag{15}$$

$$c^m \geq \frac{1}{(f+1)\sum_i n_i^m}, \forall m \tag{16}$$

$$y_i^m \in [0,1], \sum\nolimits_i y_i^m = 1, \forall m, i \tag{17}$$

That is, any deceptive λ that the defender learns has to be a part of a feasible solution (λ, y_i^m, c^m) of the system (14–17). Conversely, given any feasible (λ, y_i^m, c^m) satisfying (14–17), the deceptive attacker can make the defender learn λ by inducing the following perturbed data:

$$z_i^m = \frac{y_i^m}{c^m}$$

Proof. Equation (14) is simply the KKT condition presented in the previous section with y_i^m and c^m substituted in. Similarly, the constraints (15–16) correspond to the constraints for the deception capability of the deceptive attacker in (5–6). Finally, the constraint (17) follows from the definition of y_i^m and ensures that $\sum_i \frac{z_i^m}{\sum_j z_j^m} = 1$ and $\frac{z_i^m}{\sum_j z_j^m} \leq 1$.

According to Lemma 3, we now can prove Theorem 1 based on the characterization of the feasible solution domain of λ for the system (14–17). We denote by:

$$\mathcal{F}(\lambda, \{y_i^m, c^m\}) = \sum_m \frac{1}{c^m} \left[\sum_i y_i^m U_i^a(x_i^m) - U^a(\mathbf{x}^m, \lambda) \right]$$

the LHS of (14). In addition, we denote by $\mathbf{S} = \{(y_i^m, c^m) : \text{conditions (15–17) are satisfied}\}$ the feasible region of (y_i^m, c^m) which satisfy the conditions (15-17). In the following, we provide Lemmas 4 and 5 which specify the range of $\mathcal{F}$ as a function of λ. Essentially, if the value of $\mathcal{F}$ contains the point zero, then λ is a feasible solution of the system (14–17). We will use this property to characterize the feasible region of λ.

Lemma 4. *Assume that, WLOG, $U_1^a(x_1^m) \leq U_2^a(x_2^m) \leq \cdots \leq U_T^a(x_T^m)$ for all m. Given a λ, the optimal solution to*

$$\mathcal{F}^{\max}(\lambda) = \max_{\{y_i^m, c^m\} \in \mathbf{S}} \mathcal{F}(\lambda, \{y_i^m, c^m\}) \tag{18}$$

is determined as follows:

$$c^m = \frac{1}{(f+1)\sum_i n_i^m} \tag{19}$$

$$y_i^m = n_i^m c^m, \text{ when } i < T \tag{20}$$

$$y_i^m = 1 - c^m \sum_{i=1}^{T-1} n_i^m \text{ when } i = T \tag{21}$$

Proof. First, $\mathcal{F}(\lambda, \{y_i^m, c^m\})$ can be reformulated as:

$$\sum_m \frac{1}{c^m} \left[U_T^a(x_T^m) + \sum_{i=1}^{T-1} y_i^m [U_i^a(x_i^m) - U_T^a(x_T^m)] - U^a(\mathbf{x}^m, \lambda) \right]$$

Under our assumption that $U_1^a(x_1^m) \leq U_2^a(x_2^m) \leq \cdots \leq U_T^a(x_T^m)$, we know that $[U_i^a(x_i^m) - U_T^a(x_T^m)]$ is a strictly non-positive term for all i. Thus, maximizing $\mathcal{F}$ involves minimizing y_i^m when $i < T$. From constraint (15), the minimum y_i^m for all i is $n_i^m c^m$. This gives us $y_i^m = n_i^m c^m$ when $i < T$. From constraint (17), we know that this leaves us with $y_i^m = 1 - c^m \sum_{i=1}^{T-1} n_i^m$ when $i = T$.

Finally, given this specification of $\{y_i^m\}$, the optimization problem (18) is reduced to:

$$\max_{c^m} \sum_m \sum_{i<T} n_i^m [U_i^a(x_i^m) - U_T^a(x_T^m)] + \frac{U_T^a(x_T^m) - U^a(\mathbf{x}^m, \lambda)}{c^m}$$
$$\text{s.t. } c^m \geq \frac{1}{(f+1)\sum_i n_i^m} \text{ and } c^m \leq \frac{1}{\sum_i n_i^m}, \forall m$$

in which the objective function comprises of two terms: the first term does not depend on $\{c^m\}$ and the second term is a decreasing function of c^m (since $U_T^a(x_T^m) - U^a(\mathbf{x}^m, \lambda) > 0$). Therefore, it is maximized when c^m is minimized, which is $c^m = \frac{1}{(f+1)\sum_i n_i^m}$, concluding the proof.

Lemma 5. *Assume that, WLOG,* $U_1^a(x_1^m) \leq U_2^a(x_2^m) \leq \cdots \leq U_T^a(x_T^m)$ *for all* m*. Given a* λ*, the optimal solution to*

$$\mathcal{F}^{\min}(\lambda) = \min_{\{y_i^m, c^m\} \in \mathbf{S}} \mathcal{F}(\lambda, \{y_i^m, c^m\}) \tag{22}$$

is determined as follows:

$$c^m = \frac{1}{(f+1)\sum_i n_i^m} \tag{23}$$

$$y_i^m = n_i^m c^m, \text{ when } i > 1 \tag{24}$$

$$y_i^m = 1 - c^m \sum_{i=2}^{T} n_i^m \text{ when } i = 1 \tag{25}$$

The proof of Lemma 5 is similar. Finally, using Lemmas (4–5) and the approximation in Eq. 7, we obtain:

$$\mathcal{F}^{\max}(\lambda) = \sum_m \left[\sum_j n_j^m\right] \Bigg[U^a(\mathbf{x}^m, \lambda^{\text{true}}) + fU_T^a(x_T^m) - (f+1)U^a(\mathbf{x}^m, \lambda)\Bigg] \tag{26}$$

$$\mathcal{F}^{\min}(\lambda) = \sum_m \left[\sum_j n_j^m\right] \Bigg[U^a(\mathbf{x}^m, \lambda^{\text{true}}) + fU_1^a(x_1^m) - (f+1)U^a(\mathbf{x}^m, \lambda)\Bigg] \tag{27}$$

Observe that, given λ, $\mathcal{F}(\lambda, \cdot)$ is continuous in $\{y_i^m, c^m\}$. Therefore, given a λ', if $\mathcal{F}^{\max}(\lambda') \geq 0 \geq \mathcal{F}^{\min}(\lambda')$, there must exist $\{y_i^m, c^m\} \in \mathbf{S}$ such that $\mathcal{F}(\lambda', \{y_i^m, c^m\}) = 0$. In other words, λ' is a part of a feasible solution for (14–17). Conversely, if $\mathcal{F}^{\max}(\lambda') < 0$ or $\mathcal{F}^{\min}(\lambda') > 0$, it means λ' is not feasible for (14–17). Moreover, using Observation 1, we can infer that both $\mathcal{F}^{\max}$ and $\mathcal{F}^{\min}$ are continuous and decreasing in λ. We obtain Lemma 6 which states that feasible solutions of (14–17) form an interval.

Lemma 6. *Let us assume* $\lambda_1 < \lambda_2$ *are two feasible solutions of (14–17). Then any* $\lambda \in [\lambda_1, \lambda_2]$ *is also a feasible solution of the system.*

Proof. Since λ_1 and λ_2 are feasible solutions of (14–17), we obtain the inequalities:

$$\mathcal{F}^{\max}(\lambda_1) \geq 0 \geq \mathcal{F}^{\min}(\lambda_1)$$
$$\mathcal{F}^{\min}(\lambda_2) \geq 0 \geq \mathcal{F}^{\min}(\lambda_2)$$

For any $\lambda \in [\lambda_1, \lambda_2]$, since $\mathcal{F}^{\max}$ and $\mathcal{F}^{\min}$ are decreasing functions in λ, the following inequality holds true:

$$\mathcal{F}^{\max}(\lambda) \geq \mathcal{F}^{\max}(\lambda_2) \geq 0 \geq \mathcal{F}^{\min}(\lambda_1) \geq \mathcal{F}^{\min}(\lambda)$$

which implies that λ is also a feasible solution for (14–17), concluding the proof.

Lemma 7 specifies the interval $[\lambda^{\text{learnt}}_{\min}, \lambda^{\text{learnt}}_{\max}]$ of feasible λ values for (14–17).

Lemma 7. *There exist $\lambda^{\text{learnt}}_{\max} \geq \lambda^{\text{learnt}}_{\min}$ such that:*

$$\mathcal{F}^{\max}(\lambda^{\text{learnt}}_{\max}) = \mathcal{F}^{\min}(\lambda^{\text{learnt}}_{\min}) = 0,$$

which means $\lambda^{\text{learnt}}_{\min}$ and $\lambda^{\text{learnt}}_{\max}$ are feasible solutions for (14–17) and any $\lambda \notin [\lambda^{\text{learnt}}_{\min}, \lambda^{\text{learnt}}_{\max}]$ is not a feasible solution for (14–17).

Proof. As noted before, $\mathcal{F}^{\max}(\lambda)$ is a continuous and decreasing function in λ. On the other hand, we have:

$$\mathcal{F}^{\max}(\lambda{=}+\infty){=}\sum_m \left[\sum_j n^m_j\right]\left[U^a(\mathbf{x}^m, \lambda^{\text{true}}) - U^a_T(x^m_T)\right]{\leq}0$$

$$\mathcal{F}^{\max}(\lambda{=}-\infty){=}\sum_m \left[\sum_j n^m_j\right]\Big[U^a(\mathbf{x}^m, \lambda^{\text{true}}) + fU^a_T(x^m_T){-}(f+1)U^a_1(x^m_1))\Big] \geq 0$$

for all λ^{true} since $U^a(\mathbf{x}^m, \lambda^{\text{true}} = +\infty) = U^a_T(x^m_T)$ and $U^a(\mathbf{x}^m, \lambda^{\text{true}} = -\infty) = U^a_1(x^m_1)$ is the highest and lowest expected utilities for the attacker among all targets , respectively, and by Observation 1, $U^a(\mathbf{x}^m, \lambda^{\text{true}})$ is increasing in λ^{true}. Since $\mathcal{F}^{\max}(\lambda)$ is continuous, there must exist a value of $\lambda^{\text{learnt}}_{\max} \in (-\infty, +\infty)$ such that $\mathcal{F}^{\max}(\lambda^{\text{learnt}}_{\max}) = 0$. The proof for $\lambda^{\text{learnt}}_{\min}$ is similar.

Finally, for any $\lambda < \lambda^{\text{learnt}}_{\min}$, we have $\mathcal{F}^{\min}(\lambda) > \mathcal{F}^{\min}(\lambda^{\text{learnt}}_{\min}) = 0$ since $\mathcal{F}^{\min}$ is decreasing in λ. Similarly, for any $\lambda > \lambda^{\text{learnt}}_{\max}$, we have $\mathcal{F}^{\max}(\lambda) < \mathcal{F}^{\max}(\lambda^{\text{learnt}}_{\max}) = 0$. Both imply that λ is not feasible, concluding our proof.

By combining Lemmas 3,6, and 7, we obtain Theorem 1.

Proof of Corollary 1

Proof. Corollary 1 is deduced based on the monotonicity property of the attacker's utility (Observation 1). When $\lambda^{\text{true}}_1 \leq \lambda^{\text{true}}_2$, we have $U^a(\mathbf{x}^m; \lambda^{\text{true}}_1) \leq U^a(\mathbf{x}^m; \lambda^{\text{true}}_2)$ for all m. Based on the relationship between $U^a(\mathbf{x}^m; \lambda^{\text{true}})$ and $U^a(\mathbf{x}^m; \lambda^{\text{learnt}}_{\max})$ presented in Theorem 1, we readily obtain $\lambda^{\text{learnt}}_{\max,1} \leq \lambda^{\text{learnt}}_{\max,2}$. Similarly, we have: $\lambda^{\text{learnt}}_{\min,1} \leq \lambda^{\text{learnt}}_{\min,2}$.

Proof of Corollary 2

Proof. We first prove (8). Let's consider the true behavior parameters $\lambda^{\text{true}}_1 \leq \lambda^{\text{true}}_2$. Based on Corollary 1, the corresponding optimal deception solutions have to belong to the deception ranges: $\texttt{DecAlter}(\lambda^{\text{true}}_1) \in [\lambda^{\text{learnt}}_{\min,1}, \lambda^{\text{learnt}}_{\max,1}]$ and $\texttt{DecAlter}(\lambda^{\text{true}}_2) \in [\lambda^{\text{learnt}}_{\min,2}, \lambda^{\text{learnt}}_{\max,2}]$ where $\lambda^{\text{learnt}}_{\min,1} \leq \lambda^{\text{learnt}}_{\min,2}$ and $\lambda^{\text{learnt}}_{\max,1} \leq \lambda^{\text{learnt}}_{\max,2}$. We have two cases:

The first case is when the deception ranges do not overlap, i.e., ($\lambda^1_{\max} < \lambda^2_{\min}$). In this case, it is apparent that $\texttt{DecAlter}(\lambda^{\text{true}}_1) < \texttt{DecAlter}(\lambda^{\text{true}}_2)$.

The other case is when the ranges overlap (i.e., $\lambda_1^{max} \geq \lambda_2^{min}$). If the optimal deceptive value for one or both does not belong to the overlap, i.e., $\texttt{DecAlter}(\lambda_1^{\text{true}}) < \lambda_{\text{min},2}^{\text{learnt}}$ and/or $\texttt{DecAlter}(\lambda_2^{\text{true}}) > \lambda_{\text{max},1}^{\text{learnt}}$), the result is clearly the same as in our previous case ($\texttt{DecAlter}(\lambda_1^{\text{true}}) < \texttt{DecAlter}(\lambda_2^{\text{true}})$). On the other hand, if both values fall within the overlap, that is $\lambda_{\text{min},2}^{\text{learnt}} \leq \texttt{DecAlter}(\lambda_1^{\text{true}}), \texttt{DecAlter}(\lambda_2^{\text{true}}) \leq \lambda_{\text{max},1}^{\text{learnt}}$, both will take on the same value ($\texttt{DecAlter}(\lambda_1^{\text{true}}) = \texttt{DecAlter}(\lambda_2^{\text{true}})$). This is true because both deceptive values $\texttt{DecAlter}(\lambda_1^{\text{true}})$ and $\texttt{DecAlter}(\lambda_2^{\text{true}})$ are being optimized to maximize the same objective: the utility of the deceptive attacker (as shown in DecAlter).

Finally, (9) can be easily deduced based on (8). Let's consider $\texttt{DecAlter}(\lambda_1^{\text{true}}) < \texttt{DecAlter}(\lambda_2^{\text{true}})$. We can prove $\lambda_1^{\text{true}} < \lambda_2^{\text{true}}$ by contradiction. That is, we assume $\lambda_1^{\text{true}} \geq \lambda_2^{\text{true}}$. According to (8), it means $\texttt{DecAlter}(\lambda_1^{\text{true}}) \geq \texttt{DecAlter}(\lambda_2^{\text{true}})$, which is a contradiction.

Proof of Corollary 3

Proof. Corollary 3 is a direct result of Corollary 2. Indeed, since $\lambda_1^{\text{true}} \leq \lambda^{\text{true}} \leq \lambda_2^{\text{true}}$, we obtain the inequality among optimal deception solutions $\texttt{DecAlter}(\lambda_1^{\text{true}}) \leq \texttt{DecAlter}(\lambda^{\text{true}}) \leq \texttt{DecAlter}(\lambda_2^{\text{true}})$ as a result of Corollary 2. Therefore if $\texttt{DecAlter}(\lambda_1^{\text{true}}) = \texttt{DecAlter}(\lambda_2^{\text{true}})$, we obtain the optimal deception solution: $\texttt{DecAlter}(\lambda^{\text{true}}) = \texttt{DecAlter}(\lambda_1^{\text{true}})$.

Proof of Lemma 1

Proof. Corollary 2 says that the deception outcome $\lambda^{\text{learnt}} = \texttt{DecAlter}(\lambda^{\text{true}})$ is an increasing (not strict) function of λ^{true}, and additionally using Corollary 3, we can say that given some deception outcome λ^{learnt}, there exists (unknown) $\lambda_{\min}^{\text{true}}, \lambda_{\max}^{\text{true}}$ such that any $\lambda^{\text{true}} \in [\lambda_{\min}^{\text{true}}, \lambda_{\max}^{\text{true}}]$ leads to the same outcome $\lambda^{\text{learnt}} = \texttt{DecAlter}(\lambda^{\text{true}})$. Any λ outside of the range $[\lambda_{\min}^{\text{true}}, \lambda_{\max}^{\text{true}}]$ cannot lead to the deception outcome λ^{learnt}. Corollary 2 further implies that $\lambda_{\min}^{\text{true}}$ and $\lambda_{\max}^{\text{true}}$ are increasing functions of λ^{learnt}.

Proof of Lemma 2

Proof. Assume WLOG, $U_1^a(x_1^m) \leq U_2^a(x_2^m) \leq \cdots \leq U_T^a(x_T^m)$. We claim that $S(i, \lambda^{\text{true}}) = \sum_{j=1}^{i} q_j(\mathbf{x}^m; \lambda^{\text{true}})$ for $T > i \geq 1$ is decreasing (not strictly) in λ^{true}, or in other words, the upper bound of the i^{th} segment is decreasing (not strictly) for all i except $i = T$. This means that for any single fixed u value, increasing λ^{true} implies that $f_{\lambda^{\text{true}}}(u)$ is also increasing (or stays same) because the upper bound of the interval that u lies in shifts downwards as λ^{true} increases. $f_{\lambda^{\text{true}}}(u)$ increasing means a higher value target is chosen for attack. Thus, for fixed u, a higher λ^{true} implies that the empirical distribution places more (or equal) attacks on higher utility targets and hence $U^a(\mathbf{x}^m, E(u; \lambda^{\text{true}}))$ increases (not strictly) with λ^{true}. Finally, to prove our claim at the start of the proof, we show that the derivative of $S(i, \lambda^{\text{true}})$ is non-positive every-

where. Indeed, its derivative is computed as follows:

$$\sum_{j=1}^{i} q_j(\mathbf{x}^m;\lambda^{\text{true}})U_j^a(x_j^m) - S(i,\lambda^{\text{true}})U^a(\mathbf{x}^m;\lambda^{\text{true}})$$
$$= S(i,\lambda^{\text{true}})\Big[\sum_{j=1}^{i}\frac{q_j(\mathbf{x}^m;\lambda^{\text{true}})}{S(i,\lambda^{\text{true}})}U_j^a(x_j^m) - U^a(\mathbf{x}^m;\lambda^{\text{true}})\Big] \tag{28}$$

decomposing the attacker utility function $U^a(\mathbf{x}^m;\lambda^{\text{true}})$, as follows:

$$S(i,\lambda^{\text{true}})\sum_{j=1}^{i}\frac{q_j(\mathbf{x}^m;\lambda^{\text{true}})}{S(i,\lambda^{\text{true}})}U_j^a(x_j^m)+$$
$$\Big(\sum_{j=i+1}^{T} q_j(\mathbf{x}^m;\lambda^{\text{true}})\Big)\sum_{j=i+1}^{T}\frac{q_j(\mathbf{x}^m;\lambda^{\text{true}})}{\sum_{j=i+1}^{T} q_j(\mathbf{x}^m;\lambda^{\text{true}})}U_j^a(x_j^m)$$

As we know that $U_1^a(x_1^m) \le U_2^a(x_2^m) \ldots \le U_T^a(x_T^m)$, the following inequality holds:

$$\sum_{j=i+1}^{T}\frac{q_j(\mathbf{x}^m;\lambda^{\text{true}})}{\sum_{j=i+1}^{T} q_j(\mathbf{x}^m;\lambda^{\text{true}})}U_j^a(x_j^m) \quad \ge U_i^a(x_i^m) \ge \sum_{j=1}^{i}\frac{q_j(\mathbf{x}^m;\lambda^{\text{true}})}{S(i,\lambda^{\text{true}})}U_j^a(x_j^m)$$

Using this we get:

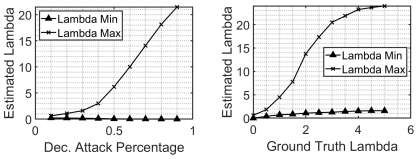
$$U^a(\mathbf{x}^m;\lambda^{\text{true}}) \ge \Big(S(i,\lambda^{\text{true}})+\sum_{j=i+1}^{T} q_j(\mathbf{x}^m;\lambda^{\text{true}})\Big)\sum_{j=1}^{i}\frac{q_j(\mathbf{x}^m;\lambda^{\text{true}})}{S(i,\lambda^{\text{true}})}U_j^a(x_j^m)$$
$$= 1\cdot\sum_{j=1}^{i}\frac{q_j(\mathbf{x}^m;\lambda^{\text{true}})}{S(i,\lambda^{\text{true}})}U_j^a(x_j^m)$$

Using the above in the derivative Eq. 28, we get that the derivative of $S(i,\lambda^{\text{true}})$ is non-positive, hence it is decreasing w.r.t. λ, concluding our proof.

Supplemental Experiments First, in Figure 4, we examine the range $[\lambda_{\min}^{\text{true}}, \lambda_{\max}^{\text{true}}]$ that the defender learns. Figure 4a shows that the range increases w.r.t. the percentage of attacks controlled by the deceptive attacker. This is intuitive, as more manipulation gives more power to the deceptive attacker. Figure 4b displays how this range also increases with the ground truth λ^{true} value of the non-deceptive attackers. As λ^{true} increases, the deceptive attacker produces a larger uncertainty range.

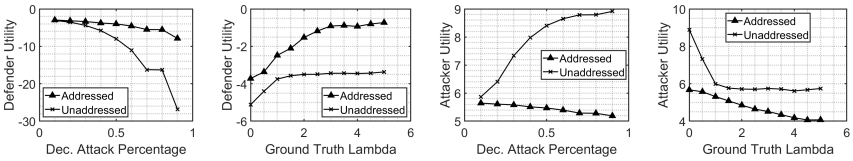
Lastly, Figures 6 through 5 are for 30-target games, and each corresponds to a previously discussed 20-target figure. We observe the same trends in both cases.

Experimental Details All experiments were run on the same HPC cluster, on instances using dual E5-2690v4 processors (28 cores). Each process was allocated 16000



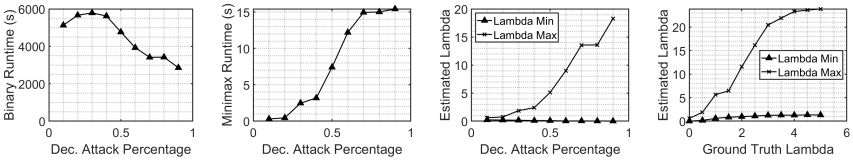
(a) Vary % of dec. attacks (b) Vary λ^{true}

Fig. 4: Lambda Range Evaluation with 20 Targets



(a) Vary % of dec. attacks (b) Vary λ^{true} (c) Vary % of dec. attacks (d) Vary λ^{true}

Fig. 5: Utility Evaluation with 30 Targets



(a) Binary Search Runtime (b) Minimax Runtime (c) Vary % of dec. attacks (d) Vary λ^{true}

Fig. 6: Runtime and λ Evaluation with 30 Targets

megabytes of RAM. Instances run Red Hat Enterprise Linux Server, version 7.8. The Matlab version used was R2018b.

All experiments used the L-Infinity norm with a value of 2 as a *rejection threshold* for non-deceptive attack samples. This is done to prevent outlying samples from compromising the binary search. Values between .5 and 5 for this metric were tested, along with the same value ranges for the L1 and L2 norms. This norm and value were shown to produce the best results, without drastically increasing the runtime of the algorithm.

Additionally, all experiments used a value of 0.05 as a *tolerance multiplier within the binary search itself*. This prevents the inherent inaccuracy of discrete attack samples from ruining binary search. For the sake of consistency, an initial random number generation seed of 1 was used across all experiments. After defender strategy generation and solving (`DecAlter`), the binary search is run 10 times, each with a different random seed. The superset of all resulting ranges forms our final uncertainty set for λ^{true}.

The trials shown in Figures 4a, 2a, 2c, 3, 6c, 5a, 5c, and 6 were conducted using a true lambda value of 0.4 and a resource/target ratio of 0.2. Those in Figures 4b, 2b, 2d, 6d, 5b, and 5d utilized a deceptive attack percentage of 0.3, and a resource/target ratio of 0.2. Experiments in Figures 3 and use deceptive attack percentage of 0.1, a true lambda value of 0.4, and a resource/target ratio of 0.2.