Learning and Planning in Feature Deception Games

Zheyuan Ryan Shi[1] , **Ariel D. Procaccia**[1] , **Kevin S. Chan**[2] , **Sridhar Venkatesan**[3] , **Noam Ben-Asher**[2] , **Nandi O. Leslie**[2] , **Charles Kamhoua**[2] and **Fei Fang**[1]

[1]Carnegie Mellon University
[2]U.S. Army Research Laboratory
[3]Perspecta Labs

Abstract

Today's high-stakes adversarial interactions feature attackers who constantly breach the ever-improving security measures. Deception mitigates the defender's loss by misleading the attacker to make suboptimal decisions. In order to formally reason about deception, we introduce the *feature deception game (FDG)*, a domain-independent game-theoretic model and present a learning and planning framework. We make the following contributions. (1) We show that we can uniformly learn the adversary's preferences using data from a modest number of deception strategies. (2) We propose an approximation algorithm for finding the optimal deception strategy and show that the problem is NP-hard. (3) We perform extensive experiments to empirically validate our methods and results.

1 Introduction

The world today poses more challenges to security than ever before. Consider cyberspace or the financial world where a defender is protecting a collection of targets, e.g. servers or accounts. Despite the ever-improving security measures, malicious attackers work diligently and creatively to outstrip the defense [Potter and Day, 2009]. Against an attacker with previously unseen exploits and abundant resources, the attempt to protect any target is almost surely a lost cause [Hurlburt, 2016]. However, the defender could induce the attacker to attack a less harmful, or even fake, target. This can be seen as a case of deception.

Deception has been recognized as an important tactic in military operations for millenia [Latimer, 2001]. More recently, deception has been extensively studied in cybersecurity [Jajodia *et al.*, 2016; Horák *et al.*, 2017]. Cyberattackers use tools such as Nmap [Lyon, 2009] to probe the target network. Security researchers have proposed many deceptive measures to manipulate the network's replies to these probes [Jajodia *et al.*, 2017; Albanese *et al.*, 2016], which could confound and mislead an attempt to attack. In addition, the use of honey-X, such as honey pots, honey users, and honey files have been proposed and implemented to attract and induce the attackers to attack these fake targets [Spitzner, 2003]. For example, Nakashima [2013] reported that country A once created encrypted, but fake, files labeled with the names of country B's military systems and put them in folders marked for sharing with country A's intelligence agency. Using these sensitive filenames as bait, country A successfully lured the hackers on the other end to these decoy targets.

Feature	Observable value	Hidden value
Operating system	Windows 2016	RHEL 7
Service version	v1.2	v1.4
IP address	10.0.1.2	10.0.2.1
Open ports	22, 445	22, 1433
Round trip time for probes [Shamsi *et al.*, 2014]	16 ms	84 ms

Table 1: Some relevant features for cybersecurity

Be it commanding an army or protecting a computer network, a common characteristic is that the attacker gathers information about the defender's system via surveillance to make decisions, and the defender can (partly) control how her system appears to the attacker. We formalize this view of the defender's system as features, and propose the *feature deception game (FDG)* to model the strategic interaction between the defender and the attacker.

It is evident that the FDG model could be applied to many domains, by appropriately defining the relevant set of features. To be concrete, we will ground our discussion in cybersecurity, where a cyberattacker observes the features of each network node and then chooses a node to compromise. The left column of Table 1 presents some relevant features. Note that these features can be continuous or discrete. If an intruder has an exploit for Windows machines, a Linux server might not be attractive to him. If the attacker is interested in exfiltration, he might choose a machine running database services. Based on such information, the defender could strategically make machines that lead to extensive harm if compromised appear undesirable to the attacker, by changing the feature values, e.g. Table 1. However, before doing so, she needs to learn the attacker's preferences from attack patterns in order to make an informed decision.

Our Contributions We make four key contributions. First, we propose the FDG model, which abstracts the information relevant to decision-making as features. In an FDG, the defender manipulates the features of each target in the system.

Upon observing the features, the attacker chooses a target to attack based on his preferences in a stochastic way.

Our second contribution is a sample complexity analysis of learning in FDG. We show that to learn a restricted class of attacker's preferences, the defender needs only gather a polynomial number of data points on a modest number of feature configurations. The sample complexity is dependent on the number of features and the norm of the inverse feature difference matrix (as introduced later).

Third, we analyze the planning problem of finding the optimal deception strategy once the attacker's preferences are learned. We show that the problem is NP-hard in general and propose an approximation algorithm.

Finally, we perform extensive experiments to validate our results. We demonstrate that we can learn the attacker's behavior model using a modest amount of data. Our planning algorithm can solve FDGs with 200 targets in 1 minute. Our combined learning and planning framework can find a deception strategy that is almost as good as the optimal strategy had we known the attacker's true preferences.

2 The Model

In an FDG, a defender aims to protect a set N of n targets from an adversary. Each target $i \in N$ has a set M of m features which the defender can strategically manipulate. The adversary observes these features and then chooses a target to attack. The defender incurs a loss $u_i \in [-1, 1]$ if the adversary chooses to attack target i.[1] The defender's objective is to minimize her expected loss.

Features Features are the key element of the FDG model. Each feature has an *observable* value and a *hidden* value. The hidden value is fixed, while the defender controls the observable value. Only the observable values are visible to the adversary. This ties into the notion of deception, where one may think of the hidden value as representing the 'ground truth,' whereas the observable value is what the defender would like the attacker to see. Table 1 shows an example of the observable and hidden values of different features in cybersecurity.

Deception means that the defender manipulates the attacker's perceived value of a target, not the actual value. Thus, changing the observable values does not affect the defender's loss u_i at each target.

Feature representation We represent the observable feature values of target i by a vector $x_i = (x_{ik})_{k \in M} \in [0, 1]^m$. We denote their corresponding hidden values as $\hat{x}_i \in [0, 1]^m$. We allow for both continuous and discrete features. In practice, we may have categorical features, such as the type of operating system, and they can be represented using one-hot encoding with binary features.

Feasibility constraints For a feature k with hidden value $\hat{x}_{ik}$, the defender can set its observable value $x_{ik} \in A(\hat{x}_{ik}) \subseteq [0, 1]$, where the feasible set $A(\hat{x}_{ik})$ is determined by the hidden value. In the sequel, for continuous features, we assume $A(\hat{x}_{ik})$ takes the form $[\hat{x}_{ik} - \tau_{ik}, \hat{x}_{ik} + \tau_{ik}] \cap [0, 1]$. This captures the feasibility constraint in setting up the observable value of a feature based on its hidden value. For binary features, $A(\hat{x}_{ik}) \subseteq \{0, 1\}$. In addition to these feasibility constraints for individual features, we also allow for linear constraints over multiple features, which could encode natural constraints for categorical features with one-hot encoding, e.g. $\sum_k x_{ik} = 1$. They may also encode the realistic considerations when setting up the observable features. For example, $x_{ik_1} + x_{ik_2} \leq 1$ could mean that a Linux machine cannot possibly have ActiveX available.

Budget constraint Deception comes at a cost. We assume the cost is additive across targets and features: $c = \sum_{i \in N} \sum_{k \in M} c_{ik}$, where $c_{ik} = \eta_{ik} |x_{ik} - \hat{x}_{ik}|$. For a continuous feature k, η_{ik} represents the cost associated with unit of change from the hidden value to the observable value. If k is binary, η_{ik} defines the cost of switching states. The defender has a budget B to cover these costs.

Defender strategies The defender's strategy is an observable feature configuration $x = \{x_i\}_{i \in N}$. We assume the defender uses only pure strategies. We discuss the relaxation of this assumption in Section 7.

Attacker strategies The attacker's pure strategy is to choose a target $i \in N$ to attack. Since human behavior is not perfectly rational, we reason about the adversary using a general class of bounded rationality models. We assume the attacker's utilities are characterized by a score function $f : [0, 1]^m \to \mathbb{R}_{>0}$ over the observable features of a target. Given the observable feature configuration $x = \{x_i\}_{i \in N}$, he attacks target i with probability $f(x_i) / \sum_{j \in N} f(x_j)$. We assume that f is parameterized by θ, and takes the form of various function classes, e.g., a neural network. Given the strong expressive power of neural networks, such model can approximate a large class of actual behavioral models of the attacker.

The ultimate goal of the defender is to find the optimal feature configuration against an unknown attacker. This may be decomposed into two subtasks. First, she learns the attacker's behavior model from attack data (*learning*). Then, she manipulates the feature configuration to minimize her expected loss (*planning*), based on the learned attacker's preferences. In the following sections, we first analyze the sample complexity of the learning task and then propose algorithms for the planning task.

3 Learning the Adversary's Preferences

The defender attempts to learn the adversary's score function f from a set D of d labeled data points. The j^{th} data point is denoted as (N^j, x^j, y^j). N^j is the set of targets in this example, and x^j is the observable feature configuration of each target in N^j. The label $y^j \in N^j$ indicates that the adversary chooses target y^j to attack.

A general approach for learning the score function f, parameterized by θ, is by maximum-likelihood estimation, i.e.,

$$\theta = \arg\max_{\theta'} \sum_j \left[L^j_{\theta'}(N^j, x^j, y^j) \right],$$

where

$$L^j_{\theta'}(N^j, x^j, y^j) = \log(f_{\theta'}(x^j_{y_j})) - \log(\sum_{i \in N^j} f_{\theta'}(x^j_i)).$$

[1] Typically, the loss u_i is non-negative, but it might be negative if, for example, the target is set up as a decoy or honeypot, and allows the defender to gain information about the attacker.

Assuming f_θ is Lipschitz and differentiable in θ, we can apply any gradient-based optimizer, e.g. RMSProp [Hinton *et al.*, 2012], to solve the optimization problem for θ. This approach can be applied to general score functions, though it is not guaranteed to find the optimal solution given the non-convexity of L.

To analyze the sample complexity of learning the adversary's preferences, we now proceed with a different learning algorithm and a special form of score function f — a one-layer neural network followed by an exponential function, parameterized by $\theta = w = (w_1, \ldots, w_m)$.

$$f(x_i) = \exp\left(\sum_{k \in M} w_k x_{ik}\right) \tag{1}$$

We show that, in an FDG with m features, the defender can learn the attacker's behavior model correctly with high probability, using only m observable feature configurations and a polynomial number of samples. We view this condition as very mild, because even if the network administrator's historical dataset does not meet the requirement, she could set up a honeynet to elicit attacks, where she can control the feature configurations at each target [Spitzner, 2003].

Haghtalab *et al.* [2016] studied a closely related problem on the sample complexity in Stackelberg security games. However, their techniques cannot be directly applied, as in security games the coverage probability is the only "feature" that the defender controls, while in FDG there could be an arbitrary number of features. We leverage the high-level idea in [Haghtalab *et al.*, 2016], and introduce the *inverse feature difference matrix* $(A^{st})^{-1}$. Specifically, given observable feature configurations $x^1, \ldots, x^m$, for any two targets $s, t \in N$, let A^{st} be the $m \times m$ matrix whose (i,j)-entry is $a^{st}_{ij} = x^i_{sj} - x^i_{tj}$. A^{st} captures the matrix-level correlation among different feature configurations. We use the matrix norm of $(A^{st})^{-1}$ to bound the learning error. In doing so, we also eliminate the technical conditions they imposed on defender strategies. Our result is formally stated as the following theorem; its proof is deferred to Appendix B.1.

Theorem 3.1. *Consider m observable feature configurations $x^1, x^2, \ldots, x^m \in [0,1]^{mn}$. Let $\alpha = \min_{s \neq t} ||(A^{st})^{-1}||$, where $||\cdot||$ is the matrix norm induced by the L^1 vector norm, i.e. $||(A^{st})^{-1}|| = \sup_{y \neq 0} \frac{|(A^{st})^{-1} y|}{|y|}$. With $\Omega(\frac{\alpha^4 m^4}{\rho \epsilon^2} \log \frac{nm}{\delta})$ samples for each of the m feature configurations, with probability $1-\delta$, we can uniformly learn $f(\cdot)$ within multiplicative error ϵ.*

Proof sketch. For feature configuration x, let $D^x(t) = \frac{f(x_t)}{\sum_{i \in N} f(x_i)}$ be the attack probability on target t, and assume $D^x(t) > \rho$ for all x and t. Construct a system of m equations $A^{st} w = b^{st}$, where $b^{st} = (\ln \frac{D^{x^1}(s)}{D^{x^1}(t)}, \ldots, \ln \frac{D^{x^m}(s)}{D^{x^m}(t)})$. To find w, we solve the equations $A^{st} \hat{w} = \hat{b}^{st}$, where $\hat{b}^{st}$ is based on empirical distributions $\hat{D}^x(\cdot)$. Using a concentration argument, we can bound the difference $|\hat{b}^{st} - b^{st}|$. With the norm of $(A^{st})^{-1}$, we can then bound the difference $|\hat{w} - w|$. □

The α in the theorem above need not be large. Consider a sequence of m feature configurations $x^1, \ldots, x^m$, and focus on targets 1 and 2. For each x^j, let the features on target 1 be identical to target 2, except for the j-th feature, where $x^j_{1j} = 1$ and $x^j_{2j} = 0$. This leads to $A^{12} = I$, and thus $\alpha \leq 1$. This is compatible with the binary encoding of categorical features, if we represent the default category as all 0's.

When the score function f is approximately learned, the optimal feature configuration assuming the learned score function $\hat{f}$ is also near optimal against the true score function f. Let $\hat{U}(x) = \frac{\sum_{i \in N} \hat{f}(x_i) u_i}{\sum_{j \in N} \hat{f}(x_j)}$ be the defender's expected loss using feature configuration x assuming $\hat{f}$. Define $U(x)$ as the corresponding defender's utility assuming f. We can adapt Theorem 3.7 of Haghtalab *et al.* [2016] to obtain the following result, whose proof is included in Appendix B.2.

Theorem 3.2. *Suppose that for some $\epsilon \leq 1/4$, $\frac{1}{1+\epsilon} < \hat{f}(x_i)/f(x_i) < 1 + \epsilon$ for all x_i. Then, $|\hat{U}(x) - U(x)| \leq 4\epsilon$ for all x. Furthermore, let $x' = \arg\max_x \hat{U}(x)$ and $x^* = \arg\max_x U(x)$, then $U(x^*) - U(x') \leq 8\epsilon$.*

4 Computing the Optimal Feature Configuration

We now embark on our second task: assuming the (learned) adversary's behavior model, compute the optimal observable feature configuration to minimize the defender's expected loss. This can be formulated as an optimization problem.

$$\begin{aligned} \min_x \quad & \frac{\sum_{i \in N} f(x_i) u_i}{\sum_{i \in N} f(x_i)} \\ s.t. \quad & \sum_{i \in N} \sum_{k \in M} \eta_{ik} |x_{ik} - \hat{x}_{ik}| \leq B \\ & \text{Categorical feature constraints} \\ & x_{ik} \in A(\hat{x}_{ik}) \qquad \forall i \in N, k \in M \end{aligned}$$

In general, this optimization problem is typically non-convex, and very difficult to solve. We show that the decision version of FDG is NP-complete, hence finding the optimal feature configuration is NP-hard. In fact, this holds even when there is only a single binary feature and the score function f takes the form in Eq. (1).

Theorem 4.1. *FDG is NP-complete.*

Proof. We reduce from the Knapsack problem. Given $v \in [0,1]^n$, $\omega \in \mathbb{R}^n_+, \Omega, V \in \mathbb{R}_+$, decide whether there exists $y \in \{0,1\}^n$ such that $\sum_{i=1}^n v_i y_i \geq V$ and $\sum_{i=1}^n \omega_i y_i \leq \Omega$.

We construct an instance of FDG. Let the set of targets be $N = \{1, \ldots, n+1\}$, and let there be a single binary feature, i.e. $M = \{1\}$ and $x_{i1} \in \{0,1\}$ for each $i \in N$. Since there is only one feature, we abuse the notation by using $x_i = x_{i1}$. Suppose each target's hidden value of the feature is $\hat{x}_i = 0$. Consider a score function f such that $f(0) = 1$ and $f(1) = 2$. For each $i \in N$, let $u_i = \frac{1 - v_i}{\delta}$ if $i \neq n+1$, and $u_{n+1} = \frac{1 + V + \sum_{i=1}^n v_i}{\delta}$. We chose a large enough $\delta \geq 1$ such that $u_{n+1} \leq 1$. In addition, for each $i \in N$, let $\eta_i = \omega_i$ if $i \neq n+1$, and $\eta_{n+1} = 0$. Finally, let the budget $B = \Omega$.

For a solution y to each Knapsack instance, we construct a solution x to the above FDG where $x_i = y_i$ for $i \neq n+1$, and $x_{n+1} = 0$. First, we know $\sum_{i \in N} \eta_i |x_i - \hat{x}_i| = \sum_{i \in N} \eta_i x_i \leq B$ if and only if $\sum_{i=1}^{n} \omega_i y_i \leq \Omega$. Since $f(x_i) > 0$ for all x_i, $\frac{\sum_{i \in N} f(x_i) u_i}{\sum_{i \in N} f(x_i)} \leq 1/\delta$ if and only if $\sum_{i \in N} (1 - \delta u_i) f(x_i) \geq 0$. Note that $\sum_{i \in N} (1 - \delta u_i) = \sum_{i=1}^{n} v_i (y_i + 1) - \sum_{i=1}^{n} v_i - V$. Thus, y is a certificate of Knapsack if and only if x is feasible for FDG and the defender's expected loss is at most $1/\delta$. □

We present an approximation algorithm based on mixed integer linear programming (MILP). Given $f(x_i) = \exp(\sum_{k \in M} w_k x_{ik})$, scaling the score by a factor of e^{-W} does not affect the attack probability, where $W = |w|$ is the L^1 norm of $w = (w_1, \ldots, w_m)$. Thus, we treat the score function as $f(x_i) = \exp(\sum_{k \in M} w_k x_{ik} - W)$.

Let $z_i = \sum_{k \in M} w_k x_{ik} - W \in [-2W, 0]$. We divide the interval $[-2W, 0]$ into $2W/\epsilon$ subintervals, each of length ϵ. On interval $[-l\epsilon, -(l-1)\epsilon]$ with $l = 0, 1, \ldots, 2W/\epsilon$, we approximate the function e^{z_i} with the line segment of slope γ_l connecting the points $(-l\epsilon, e^{-l\epsilon})$ and $(-(l-1)\epsilon, e^{-(l-1)\epsilon})$. We use this method to approximate the score of each target, f_i, in the following mathematical program. We represent $z_i = -\sum_l z_{il}$, where each variable z_{il} indicates the quantity z_i takes up on the interval $[-l\epsilon, -(l-1)\epsilon]$. The constraints in Eq. (6)-(7) ensure that $z_{i(l+1)} > 0$ only if $z_{il} = \epsilon$. While the formulation presented in Eq. (2)-(9) is not technically a MILP, we can linearize the objective and the constraint involving absolute value, but avoid doing so here for clarity. The MILP formulation is relegated to Appendix A.

$$\min_{f,z,x,y} \quad \frac{\sum_i f_i u_i}{\sum_i f_i} \tag{2}$$

$$s.t. \quad f_i = e^{-2W} + \sum_l \gamma_l (\epsilon - z_{il}) \qquad \forall i \in N \tag{3}$$

$$\sum_{k \in M} w_k x_{ik} - W = -\sum_l z_{il} \qquad \forall i \in N \tag{4}$$

$$\sum_{i,k} \eta_{ik} |x_{ik} - \hat{x}_{ik}| \leq B \tag{5}$$

$$\epsilon y_{il} \leq z_{il}, z_{i(l+1)} \leq \epsilon y_{il} \qquad \forall l, \forall i \in N \tag{6}$$

$$z_{il} \in [0, \epsilon], y_{il} \in \{0, 1\} \qquad \forall l, \forall i \in N \tag{7}$$

$$\text{Categorical constraints} \tag{8}$$

$$x_{ik} \in A(\hat{x}_{ik}) \qquad \forall i \in N, k \in M \tag{9}$$

We can now establish the following bound, whose proof appears in Appendix B.3.

Theorem 4.2. *Given $\epsilon < 1$, The MILP above is a $2\epsilon^2$-approximation to the original problem.*

Proof sketch. We first need to analyze the tightness of the linear approximation. Using inequalities $e^{\epsilon} \leq \epsilon^2 + \epsilon + 1$ for $\epsilon < 1.7$ and $ve^{1/v-1} \leq 1 + (v-1)^2/2$ for $v \geq 1$, we can uniformly bound the approximation error $\frac{\hat{f}(x_i) - f(x_i)}{f(x_i)} \leq \frac{\epsilon^2}{2}$. This allows us to bound the difference between true and approximated defender's expected utility on any feature configuration by $|\hat{U}(x) - U(x)| \leq \epsilon^2$. We use the triangle inequality to connect the approximate solution and the true solution. □

While the mathematical program in Eq. (2)-(9) could be transformed into a MILP, the necessary linearization introduces many additional variables, increasing the size of the problem. To improve scalability, we perform binary search on the objective value δ. Specifically, the objective at each iteration of binary search becomes

$$\min_{f,z,x,y} \quad \sum_i f_i u_i - \delta \sum_i f_i. \tag{10}$$

The complete procedure is given as Alg. 1 in Appendix A. With the objective in Eq. (10), we no longer need to perform linearization to obtain a MILP. This leads to significant performance improvement as we show later in the experiments. We also preserve the approximation bound; the proof appears in Appendix B.4.

Theorem 4.3. *Given $\epsilon < 1$ and tolerance ϵ_{bs}, binary search gives a $(2\epsilon^2 + \epsilon_{bs})$-approximation.*

In addition, we propose two exact algorithms for special cases of FDG, which can be found in Appendix C. When the deception cost is associated with discrete features only, we provide an exact MILP formulation. When there is no budget and feasibility constraints, we can find the optimal defender strategy in $O(n \log n + m)$ time.

5 Experiments

We present the experimental results for our learning and planning algorithms separately, and then combine them to demonstrate the effectiveness of our learning and planning framework. All experiments are carried out on a 3.8GHz Intel Core i5 CPU with 32GB RAM. We use RMSProp as our gradient-based optimizer, Ipopt as our non-convex solver, and CPLEX 12.8 as the MILP solver. All results are averaged over 20 instances; error bars represent the standard deviations.

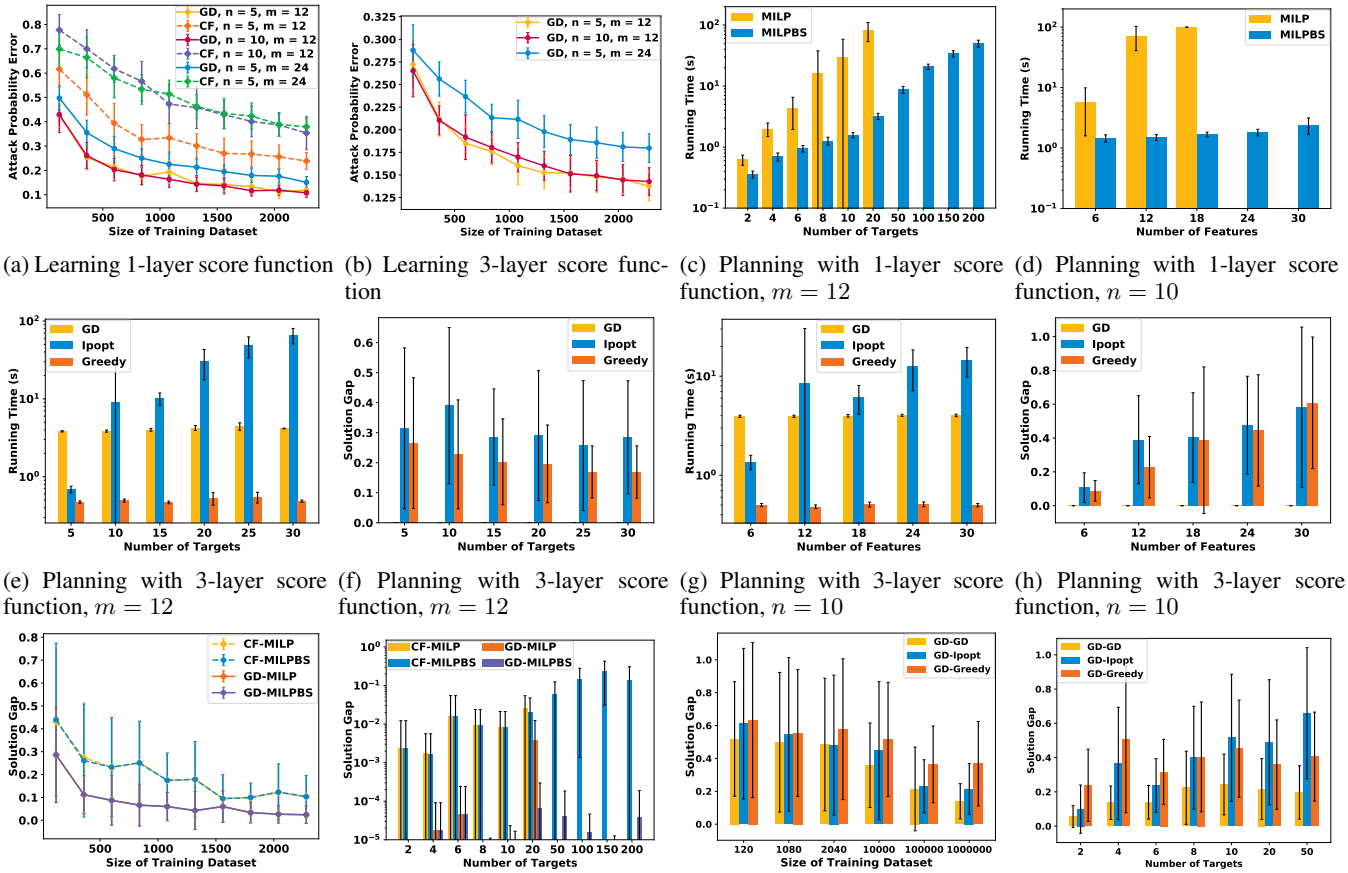
5.1 Learning

Simple score function First, we assume the adversary uses the score function specified in Eq (1). The defender learns this score function using either the closed-form estimation (CF) in Theorem 3.1, or a gradient-based algorithm (GD).

We study how the learning accuracy changes with the size of training sample d. We sample the parameters of the true score function f uniformly at random from $[-0.5, 0.5]$. We then generate m feature configurations uniformly at random. For each of them, we sample the attacked target d/m times according to f, obtaining a training set of d samples. We also generate a test set $\tilde{D}$ of 5×10^5 feature configurations sampled uniformly at random. We measure the learning error as the mean total variation distance between the attack distribution from the learned model $\hat{f}$ and that of the true model f:

$$\frac{1}{|\tilde{D}|} \sum_{j=1}^{|\tilde{D}|} d_{TV} \left(\left(\frac{f(x_i^j)}{\sum_{t \in N} f(x_t^j)} \right)_{i \in N}, \left(\frac{\hat{f}(x_i^j)}{\sum_{t \in N} \hat{f}(x_t^j)} \right)_{i \in N} \right).$$

Figure 1a shows that for both learning approaches, the learning error decreases as we increase the number of samples. In general, GD outperforms CF. This is expected as the size of the training dataset is far smaller than that implied by Theorem 3.1. The learning error increases for both methods when we have more features, and for CF, more targets as well.



(a) Learning 1-layer score function (b) Learning 3-layer score function (c) Planning with 1-layer score function, $m = 12$ (d) Planning with 1-layer score function, $n = 10$

(e) Planning with 3-layer score function, $m = 12$ (f) Planning with 3-layer score function, $m = 12$ (g) Planning with 3-layer score function, $n = 10$ (h) Planning with 3-layer score function, $n = 10$

(i) Learning + Planning, 1-layer score function, $n = 5, m = 12$ (j) Learning + Planning, 1-layer score function, $m = 12$ (k) Learning + Planning, 3-layer score function, $n = 5, m = 12$ (l) Learning + Planning, 3-layer score function, $m = 12$

Figure 1: Experimental results

Complex score function We assume the adversary uses a 3-layer neural network score function, whose details are in Appendix E. Since CF is no longer applicable, for each sample size d, we generate d feature configurations and sample an attacked target for each of them in the training set. Fig. 1b shows GD can minimize the learning error to below 0.15.

We also measured $|\hat{\theta} - \theta|$, the L_1 error in the score function parameter θ, which directly relates to the sample complexity bound in Theorem 3.1. We include the results in Appendix D.

5.2 Planning

We test our algorithms on finding the optimal feature configuration against a known attacker model. The FDG parameter distributions are included in Appendix E. Experiments on our special case algorithms are included in Appendix D.

Simple score function Fig. 1c shows that the MILP in Eq (2)-(9) becomes impractical to test when $n \geq 50$, whereas the binary search extension (MILPBS) scales up to problems with 200 targets. We note that this is already at the scale of many real-world problems. Fig. 1d shows that MILPBS also scales better on the number of features. We set the MILP's error bound at 0.005; the difference in the two algorithms' results is negligible.

Complex score function When the features are continuous without feasibility constraints, planning becomes a non-convex optimization problem. We can apply the gradient-based optimizer or non-convex solver. We also introduce a greedy heuristic inspired by the special case algorithm in Appendix C.2. GREEDY (Alg. 2 in Appendix A) finds the feature vectors that maximize and minimize the score, respectively, by GD. It then greedily applies these features to targets of extreme losses. Recall that $U(x)$ is the defender's expected loss using feature configuration x. We measure the solution gap of alg $\in$ {Ipopt, GD, GREEDY} as $\frac{U(x^{\text{alg}}) - U(x^{\text{GD}})}{U(x^{\text{GD}})}$, where x^{alg} is the solution from alg. We choose GD as baseline as it typically yields the best solution.

Fig. 1e–1h show the running time and solution gap fixing either $m = 12$ or $n = 10$. GD does well in both dimensions. GREEDY lags in solution quality, while non-convex solver struggles in both. However, these algorithms do not provide any solution guarantees, thus we cannot conclude that any of them (including GD) yields a good defender strategy.

5.3 Combining Learning and Planning

We integrate the learning and planning algorithms to examine our full framework. The defender learns a score function

$\hat{f}$ using algorithm L. Then, she uses planning algorithm P to find an optimal configuration $x^{\mathrm{L,P}}$ assuming $\hat{f}$. We measure the solution gap as $\frac{U(x^{\mathrm{L,P}})-U(x^*)}{U(x^*)}$, where x^* is the optimal feature configuration against the true attacker model, computed using MILPBS or GD. We choose them as baselines as they have better solution quality in the previous experiments.

Simple score function We test learning algorithms L $\in$ {CF, GD} and planning algorithms P $\in$ {MILP, MILPBS}. As shown in Fig. 1i, more training data lead to smaller solution gaps. Consistently with the results in Fig. 1a, GD learning has better performance. Consistently with the planning results, the difference between MILP and MILPBS in solution quality is negligible. With $n \leq 20$ targets, all algorithms yield solution gaps below 0.1 (Fig. 1j). With $n \geq 50$, CF learning shows unsatisfactory results. Compared to Fig. 1a, we have more targets, which increases the sample complexity. GD learning performs well on all problem sizes.

Complex score function We test learning algorithm GD and planning algorithms P $\in$ {GD, Ipopt, GREEDY}. Fig. 1k shows that all algorithms benefit from having more training data, and GD performs the best. Compared to the case with 1-layer score functions, more data are required here to achieve a small solution gap. Since we have small learning error for both cases (Fig. 1a,1b), this suggests planning is more sensitive to complex score functions than simple score functions. When given enough data, Fig. 1l shows that GD can achieve a solution gap below 0.2 with as many as 50 targets.

6 Related Work

Deception as a game Deception has been studied in many domains, and of immediate relevance is its use in cybersecurity [Rowe, 2007]. Studies have suggested that deceptively responding to an attacker's scanning and probing could be a useful defensive measure [Jajodia *et al.*, 2017; Albanese *et al.*, 2016]. Schlenker *et al.* [2018] and Wang and Zeng [2018] propose game-theoretic models where the defender strategically manipulates the query response to a known attacker. In addition to proposing a domain-independent model, we advance the state of the art by (1) providing a unified learning and planning framework with theoretical guarantee which can deal with unknown attackers, (2) extending the finite "type" space in both papers, where "type" is defined by the combination of feature values, to an infinite feature space that allows for both continuous and discrete features, and (3) incorporating a highly expressive bounded rationality model whereas both papers assume perfectly rational attackers.

On the game-theoretic modeling of deception in general, Horák *et al.* [2017] study a defender that engages an attacker in a sequential interaction. Nguyen *et al.* [2019] offer a complementary view where the attacker aims at deceiving the defender. In [Yin *et al.*, 2014; Guo *et al.*, 2017; Nguyen *et al.*, 2019] deception is defined as deceptively allocating defensive resources. We study feature deception where no effective tools can thwart an attack, which is arguably more realistic in high-stakes interactions. When such tools exist, feature deception is still valuable for strategic defense.

Learning in Stackelberg games Much work has also been devoted to learning in Stackelberg games. Our work is most directly related to that of Haghtalab *et al.* [2016]. They consider an SUQR adversary behavior model, and provide a similar learning guarantee as our Theorem 3.1. The only decision variable in their model, the coverage probability, may be viewed as a single feature in FDG. FDG allows for an arbitrary number of features, and this realistic extension makes their key technique inapplicable. Our main learning result also removes the technical constraints on defender strategies present in their work. Sinha *et al.* [2016] study learning adversary's preferences in a probably approximately correct (PAC) setting. However, their learning accuracy depends heavily on the quality of distribution from which they sample the defender's strategies. We provide a uniform guarantee in a distribution-free context. Other papers [Blum *et al.*, 2014; Marecki *et al.*, 2012; Letchford *et al.*, 2009; Peng *et al.*, 2019] study the online learning setting with rational attackers. As pointed out by Haghtalab *et al.* [2016], considering the more realistic bounded rationality scenario allows us to make use of historical data and use our algorithm more easily in practice.

Planning with boundedly rational attackers Yang *et al.* [2012] propose a MILP-based solution similar to our approximation algorithm. We generalize the coverage probability to features, and adopt a more expressive behavior model. The subsequent papers that incorporate learning with such bounded rationality models do not provide any theoretical guarantee [Yang *et al.*, 2014; Fang *et al.*, 2015]. In these papers, manipulating coverage probabilities affects the defender's utility. This is not the case in FDG due to the very notion of deception. However, this does not make our model technically easier to analyze in any substantial way.

7 Discussion and Conclusion

We conclude with a few remarks regarding the generality and limitations of our work. First, our model allows for more sophisticated attackers who can outstrip deception. A singleton feasible set $A(\hat{x}_{ik})$ could indicate that the defender knows the attacker is able to find out the hidden value of a feature.

Second, we assumed the hidden feature values are fixed, as they often represent environmental parameters beyond the defender's control, or at least present high cost of manipulation. Altering them also does not align conceptually with deception. As a result, we treat defender's losses u_i as fixed.

Third, we assumed the defender uses only pure strategies. In many domains such as cybersecurity, it is often too costly to frequently perform system reconfiguration. Thus, the system appears to the attacker as static. We leave to future work to explore the additional strength of mixed strategies in applications where they are appropriate.

Finally, it would be interesting to consider the attacker's ability to recognize deception after repeated interactions.

References

[Albanese *et al.*, 2016] Massimiliano Albanese, Ermanno Battista, and Sushil Jajodia. Deceiving attackers by creating a virtual attack surface. In *Cyber Deception*. Springer, 2016.

[Blum *et al.*, 2014] Avrim Blum, Nika Haghtalab, and Ariel D Procaccia. Learning optimal commitment to overcome insecurity. In *NIPS*, 2014.

[Fang *et al.*, 2015] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI*, 2015.

[Guo *et al.*, 2017] Qingyu Guo, Bo An, Branislav Bosanskỳ, and Christopher Kiekintveld. Comparing strategic secrecy and stackelberg commitment in security games. In *IJCAI*, pages 3691–3699, 2017.

[Haghtalab *et al.*, 2016] Nika Haghtalab, Fei Fang, Thanh H Nguyen, Arunesh Sinha, Ariel D Procaccia, and Milind Tambe. Three strategies to success: Learning adversary models in security games. In *IJCAI*, 2016.

[Hinton *et al.*, 2012] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. 2012.

[Horák *et al.*, 2017] Karel Horák, Quanyan Zhu, and Branislav Bošanskỳ. Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security. In *GameSec*, 2017.

[Hurlburt, 2016] George Hurlburt. " good enough" security: The best we'll ever have. *Computer*, 49(7):98–101, 2016.

[Jajodia *et al.*, 2016] Sushil Jajodia, VS Subrahmanian, Vipin Swarup, and Cliff Wang. *Cyber deception*. Springer, 2016.

[Jajodia *et al.*, 2017] Sushil Jajodia, Noseong Park, Fabio Pierazzi, Andrea Pugliese, Edoardo Serra, Gerardo I Simari, and VS Subrahmanian. A probabilistic logic of cyber deception. *IEEE Transactions on Information Forensics and Security*, 12(11), 2017.

[Latimer, 2001] Jon Latimer. *Deception in War*. John Murray, 2001.

[Letchford *et al.*, 2009] Joshua Letchford, Vincent Conitzer, and Kamesh Munagala. Learning and approximating the optimal strategy to commit to. In *SAGT*, 2009.

[Lyon, 2009] Gordon Fyodor Lyon. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

[Marecki *et al.*, 2012] Janusz Marecki, Gerry Tesauro, and Richard Segal. Playing repeated stackelberg games with unknown opponents. In *AAMAS*, 2012.

[Nakashima, 2013] Ellen Nakashima. *To thwart hackers, firms salting their servers with fake data*, 2013. http://articles.washingtonpost.com/2013-01-02/world/36211654_1_hackers-servers-contract-negotiations.

[Nguyen *et al.*, 2019] Thanh Hong Nguyen, Yongzhao Wang, Arunesh Sinha, and Michael P. Wellman. Deception in finitely repeated security games. In *AAAI*, 2019.

[Peng *et al.*, 2019] Binghui Peng, Weiran Shen, Pingzhong Tang, and Song Zuo. Learning optimal strategies to commit to. In *AAAI*, 2019.

[Potter and Day, 2009] Bruce Potter and Greg Day. The effectiveness of anti-malware tools. *Computer Fraud & Security*, 2009(3):12–13, 2009.

[Rowe, 2007] Neil C Rowe. Deception in defense of computer systems from cyber attack. In *Cyber Warfare and Cyber Terrorism*. IGI Global, 2007.

[Schlenker *et al.*, 2018] Aaron Schlenker, Omkar Thakoor, Haifeng Xu, Fei Fang, Milind Tambe, Long Tran-Thanh, Phebe Vayanos, and Yevgeniy Vorobeychik. Deceiving cyber adversaries: A game theoretic approach. In *AAMAS*, 2018.

[Shamsi *et al.*, 2014] Zain Shamsi, Ankur Nandwani, Derek Leonard, and Dmitri Loguinov. Hershel: single-packet os fingerprinting. In *ACM SIGMETRICS Performance Evaluation Review*, 2014.

[Sinha *et al.*, 2016] Arunesh Sinha, Debarun Kar, and Milind Tambe. Learning adversary behavior in security games: A pac model perspective. In *AAMAS*, 2016.

[Spitzner, 2003] Lance Spitzner. The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 99(2):15–23, 2003.

[Wang and Zeng, 2018] Wei Wang and Bo Zeng. A two-stage deception game for network defense. In *GameSec*, 2018.

[Yang *et al.*, 2012] Rong Yang, Fernando Ordonez, and Milind Tambe. Computing optimal strategy against quantal response in security games. In *AAMAS*, 2012.

[Yang *et al.*, 2014] Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. In *AAMAS*, 2014.

[Yin *et al.*, 2014] Yue Yin, Bo An, Yevgeniy Vorobeychik, and Jun Zhuang. Optimal deceptive strategies in security games: A preliminary study. In *AAAI Spring Symposium on Applied Computational Game Theory*, 2014.

Appendix: Learning and Planning in Feature Deception Games

A Deferred Algorithms

We show the MILP formulation for the mathematical program in Eq. (2)- (9). We use $M_c \subseteq M$ to denote the set of continuous features, and $M_d = M - M_c$ denotes the set of discrete features. For discrete feature $k \in M_d$, we assume that η_{ik} and budget B have been processed such that Constraint (5) has been modified to $\sum_{i\in N}\left(\sum_{k\in M_c}\eta_{ik}|x_{ik}-\hat{x}_{ik}|+\sum_{k\in M_d}\eta_{ik}x_{ik}\right) \le B$. This transformation based on $\hat{x}_{ik}\in\{0,1\}$ simplifies our presentation below.

$$\max_{b,d,g,h,q,s,t,v,y} \quad \sum_{i\in N} t_i \tag{11}$$

$$s.t. \quad t_i = ve^{-2W} + \sum_l \gamma_l(v\epsilon - s_{il}) \tag{12}$$

$$\sum_{k\in M_c} w_k q_{ik} + \sum_{k\in M_d} w_k b_{ik} - Wv = -\sum_l s_{il} \tag{13}$$

$$h_{ik} \ge q_{ik} - \hat{x}_{ik}v, h_{ik} \ge \hat{x}_{ik}v - q_{ik} \qquad \forall k \in M_c \tag{14}$$

$$\sum_{i\in N}\left(\sum_{k\in M_d}\eta_{ik}b_{ik} + \sum_{k\in M_c}\eta_{ik}h_{ik}\right) \le Bv \tag{15}$$

$$\epsilon g_{il} \le s_{il}, s_{i(l+1)} \le \epsilon g_{il} \qquad \forall l \tag{16}$$

$$s_{il} \le v\epsilon \qquad \forall l \tag{17}$$

$$g_{il} \le v, g_{il} \le Zy_{il}, g_{il} \ge v - Z(1-y_{il}) \qquad \forall l \tag{18}$$

$$b_{ik} \le v, b_{ik} \le Zd_{ik}, b_{il} \ge v - Z(1-d_{ik}) \ \forall k \in M_d \tag{19}$$

$$q_{ik} \in [(\hat{x}_{ik}-\tau_{ik})v, (\hat{x}_{ik}+\tau_{ik})v] \cap [0,1] \quad \forall k \in M_c \tag{20}$$

$$\sum_{i\in N} u_i t_i = 1 \tag{21}$$

Categorical constraints (22)

$$t_i, v, s_{il}, q_{ik}, h_{ik}, g_{il} \ge 0, y_{il} \in \{0,1\} \quad \forall k \in M_c, \forall l \tag{23}$$

$$b_{ik} \ge 0, d_{ik} \in \{0,1\} \qquad \forall k \in M_d \tag{24}$$

We establish the variables in the MILP above with the FDG variables as below.

$$t_i = \frac{f_i}{\sum_{i\in N} f_i u_i}, \qquad v = \frac{1}{\sum_{i\in N} f_i u_i} \tag{25}$$

$$h_{ik} = \frac{|x_{ik}-\hat{x}_{ik}|}{\sum_{i\in N} f_i u_i}, \quad q_{ik} = \frac{x_{ik}}{\sum_{i\in N} f_i u_i}, \quad \forall k \in M_c \tag{26}$$

$$d_{ik} = x_{ik}, \qquad b_{ik} = \frac{x_{ik}}{\sum_{i\in N} f_i u_i}, \quad \forall k \in M_d \tag{27}$$

$$s_{il} = \frac{z_{il}}{\sum_{i\in N} f_i u_i}, \quad g_{il} = \frac{y_{il}}{\sum_{i\in N} f_i u_i}, \qquad \forall l \tag{28}$$

(29)

All equations above involving index i without summation should be interpreted as applying to all $i \in N$.

Algorithm 1: MILP-BS

```
Initialize L = −1, U = 1, δ = 0, ε_bs
while U − L > ε_bs do
    Solve the MILP in Eq. (2)-(9) with objective in Eq. (10).
    if objective value < 0 then
        Let U = δ
    else
        Let L = δ
return U, the MILP solution when U was last updated
```

Algorithm 2: GREEDY

```
Use gradient-based method to find x^max ≈ arg max_x f(x) and x^min ≈ arg min_x f(x).
Sort the targets such that u_1 ≤ u_2 ≤ ··· ≤ u_n.
Initialize i = 1, j = n.
while i < j and budget > 0 do
    Let x_i ← x^max if
    if Cost(x_i ← x^max) ≤ remaining budget then
        x_i ← x^max, decrease the budget, i = i + 1.
    if Cost(x_j ← x^min) ≤ remaining budget then
        x_j ← x^min, decrease the budget, j = j − 1.
return feature configuration x
```

B Deferred Proofs

B.1 Proof of Theorem 3.1

We require the following lemma.

Lemma B.1. *[Haghtalab et al., 2016] Given observable features $x \in [0,1]^{mn}$, and $\Omega(\frac{1}{\rho\epsilon^2}\log\frac{n}{\delta})$ samples, we have $\frac{1}{1+\epsilon} \le \frac{\hat{D}^x(t)}{D^x(t)} \le 1+\epsilon$ with probability $1-\delta$, for all $t \in N$.*

Proof of Theorem 3.1. Fix $\epsilon, \delta > 0$. Fix two nodes $s \ne t$. For each x^i where $i = 1, 2, \ldots, m$, we have

$$\sum_{j=1}^{m} w_j(x^i_{sj} - x^i_{tj}) = \ln\frac{D^{x^i}(s)}{D^{x^i}(t)}$$

Let

$$b^{st} = (\ln\frac{D^{x^1}(s)}{D^{x^1}(t)}, \ldots, \ln\frac{D^{x^m}(s)}{D^{x^m}(t)}).$$

The system of equations above can be represented by $A^{st}w = b^{st}$. Let $||\cdot||$ be the matrix norm induced by L^1 vector norm, that is,

$$||A^{st}|| = \sup_{x\ne 0}\frac{|A^{st}x|}{|x|}, \qquad \text{where } |x| = \sum_{j=1}^{m}|x_j|.$$

It is known that $||A^{st}|| = \max_{1\le j\le m}\sum_{i=1}^{m}|a^{st}_{ij}|$. In our case, the feature values are bounded in $[0,1]$ and thus $|a^{st}_{ij}| \le 1$. This yields $||A^{st}|| \le m$. Now, choose s, t such that $||(A^{st})^{-1}|| = \alpha$. Suppose A^{st} is invertible.

Let $\epsilon' = \frac{\epsilon}{4\alpha^2 m^2}$ and $\delta' = \frac{\delta}{m}$. Suppose we have $\Omega(\frac{1}{\rho\epsilon'^2}\log\frac{n}{\delta'})$ samples. From Lemma B.1, for any node $r \in N$ and any feature configuration x^i where $i = 1, 2, \ldots, m$, $\frac{1}{1+\epsilon'} \le \frac{\hat{D}^{x^i}(r)}{D^{x^i}(r)} \le 1+\epsilon'$ with probability $1-\delta'$. The bound holds for all strategies simultaneously with probability at least $1 - m\delta' = 1-\delta$, using a union bound argument. In particular, for our chosen nodes s and t, we have

$$\frac{1}{(1+\epsilon')^2} \le \frac{\hat{D}^{x^i}(s)}{\hat{D}^{x^i}(t)}\frac{D^{x^i}(t)}{D^{x^i}(s)} \le (1+\epsilon')^2, \quad \forall i = 1, \ldots, m$$

Define $\hat{b}^{st}$ similarly as b^{st} but using empirical distribution $\hat{D}$ instead of true distribution D. Let $e = \hat{b}^{st} - b^{st}$. Then, for each $i = 1, \ldots, m$, we have

$$-2\epsilon' \le 2\ln\frac{1}{1+\epsilon'} \le e_i = \ln\frac{\hat{D}^{x^i}(s)D^{x^i}(t)}{\hat{D}^{x^i}(t)D^{x^i}(s)} \le 2\ln(1+\epsilon') \le 2\epsilon'$$

Therefore, we have $|e| \le 2\epsilon' m$. Let $\hat{w}$ be such that $A^{st}\hat{w} = \hat{b}^{st}$, i.e. $\hat{w} - w = (A^{st})^{-1}e$. Observe that

$$\begin{aligned}
\frac{|(A^{st})^{-1}e|/|(A^{st})^{-1}b^{st}|}{|e|/|b^{st}|} &\le \max_{\tilde{e},\tilde{b}^{st}\neq 0} \frac{|(A^{st})^{-1}\tilde{e}|/|(A^{st})^{-1}\tilde{b}^{st}|}{|\tilde{e}|/|\tilde{b}^{st}|} \\
&= \max_{\tilde{e}\neq 0}\frac{|(A^{st})^{-1}\tilde{e}|}{|\tilde{e}|}\max_{\tilde{b}^{st}\neq 0}\frac{|\tilde{b}^{st}|}{|(A^{st})^{-1}\tilde{b}^{st}|} \\
&= \max_{\tilde{e}\neq 0}\frac{|(A^{st})^{-1}\tilde{e}|}{|\tilde{e}|}\max_{y\neq 0}\frac{|A^{st}y|}{|y|} \\
&= ||(A^{st})^{-1}|| \cdot ||A^{st}||
\end{aligned}$$

This leads to

$$\begin{aligned}
|(A^{st})^{-1}e| &\le ||(A^{st})^{-1}|| \cdot ||A^{st}|| \cdot |e| \cdot \frac{|(A^{st})^{-1}b^{st}|}{|b^{st}|} \\
&\le ||(A^{st})^{-1}|| \cdot ||A^{st}|| \cdot |e| \cdot \max_{\tilde{b}^{st}\neq 0}\frac{|(A^{st})^{-1}\tilde{b}^{st}|}{|\tilde{b}^{st}|} \\
&= ||(A^{st})^{-1}||^2 \cdot ||A^{st}|| \cdot |e| \\
&\le \alpha^2 m(2\epsilon' m)
\end{aligned}$$

For any observable feature configuration x,

$$\begin{aligned}
&\left|\left(\sum_{j=1}^m w_j x_{ij}\right) - \left(\sum_{j=1}^m \hat{w}_j x_{ij}\right)\right| \le \sum_{j=1}^m |\hat{w}_j - w_j| \\
&= |(A^{st})^{-1}e| \le \alpha^2 m(2\epsilon' m) = \frac{\epsilon}{2}
\end{aligned}$$

Therefore,

$$\frac{1}{1+\epsilon} \le \frac{f(x_i)}{\hat{f}(x_i)} \le 1+\epsilon$$

□

B.2 Proof of Theorem 3.2

Proof. Let $\hat{f}(x_i) = \exp(\sum_k \hat{w}_k x_{ik})$ and $f(x_i) = \exp(\sum_k w_k x_{ik})$. Since

$$\frac{1}{1+\epsilon} < \frac{\hat{f}(x_i)}{f(x_i)} < 1+\epsilon,$$

we get

$$-\epsilon \le -\ln(1+\epsilon) < \sum_k (\hat{w}_k - w_k)x_{ik} = \ln\frac{\hat{f}(x_i)}{f(x_i)} < \ln(1+\epsilon) \le \epsilon.$$

That is, $|\sum_k(\hat{w}_k - w_k)x_{ik}| < \epsilon$. The proof of Theorem 3.7 in [Haghtalab *et al.*, 2016] now follows if we redefine their $u_i(p_i)$ as $\sum_{k\in M} w_k x_{ik}$ and $\hat{u}_i(p_i)$ as $\sum_{k\in M}\hat{w}_k x_{ik}$. For completeness, we adapt their proof below using our notations.

Let $\bar{D}^x(t) = \frac{\hat{f}(x_t)}{\sum_i \hat{f}(x_i)}$. Then, we have

$$\begin{aligned}
\left|\ln\frac{\bar{D}^x(t)}{D^x(t)}\right| &= \left|\left(\sum_k(\hat{w}_k - w_k)x_{tk}\right) - \ln\frac{\sum_i\exp\{\sum_k \hat{w}_k x_{ik}\}}{\sum_i\exp\{\sum_k w_k x_{ik}\}}\right| \\
&\le \left|\sum_k(\hat{w}_k - w_k)x_{tk}\right| + \\
&\left|\ln\frac{\sum_i\exp\{\sum_k w_k x_{ik}\}\exp\{\sum_k(\hat{w}_k - w_k)x_{ik}\}}{\sum_i\exp\{\sum_k w_k x_{ik}\}}\right| \\
&< \epsilon + \max_i\left|\ln\exp\{\sum_k(\hat{w}_k - w_k)x_{ik}\}\right| \\
&< 2\epsilon
\end{aligned}$$

Using a few inequalities we can bound $\left|\frac{\bar{D}^x(t)}{D^x(t)} - 1\right| \le 4\epsilon$. Finally,

$$\begin{aligned}
|\hat{U}(x) - U(x)| &= \left|\sum_{i\in N}(\bar{D}^x(i) - D^x(i))u_i\right| \\
&\le \sum_{i\in N}\left|\bar{D}^x(i) - D^x(i)\right||u_i| \\
&= \sum_{i\in N}\left|\frac{\bar{D}^x(i)}{D^x(i)} - 1\right||u_i|\,D^x(i) \\
&\le 4\epsilon\sum_{i\in N}|u_i|\,D^x(i) \\
&\le 4\epsilon\max_{i\in N}|u_i| \\
&\le 4\epsilon
\end{aligned}$$

Let $x^* = \arg\min_x U(x)$ be the true optimal feature configuration and $x' = \arg\min_x \hat{U}(x)$ be the optimal configuration using the learned score function $\hat{f}$. Thus, we have $U(x') \le \hat{U}(x') + 4\epsilon \le \hat{U}(x^*) + 4\epsilon \le U(x^*) + 8\epsilon$. □

B.3 Proof of Theorem 4.2

To analyze the approximation bound of this MILP, we first need to analyze the tightness of the linear approximation.

Consider two points s_1, s_2 where $s_2 - s_1 = \epsilon$. The line segment is $t(s) = \frac{1}{\epsilon}(e^{s_2} - e^{s_1})s - \frac{1}{\epsilon}(e^{s_2} - e^{s_1})s_1 + e^{s_1}$. Let $\Delta(s)$ be the ratio between the line and e^s on the interval $[s_1, s_2]$. It is easy to find that $\Delta(s)$ is maximized at

$$s^* = 1 + s_1 - \frac{\epsilon}{e^\epsilon - 1},$$

with

$$\Delta(s^*) = \frac{\frac{e^\epsilon - 1}{\epsilon}}{\exp\{1 - \frac{\epsilon}{e^\epsilon - 1}\}}.$$

Now, let $v = \frac{e^\epsilon - 1}{\epsilon}$. It is known that $v \in [1, 1+\epsilon]$ when $\epsilon < 1.7$. Note that $\delta(x^*) = v \exp\{\frac{1}{v} - 1\} \le 1 + (v-1)^2/2$, which holds for all $v \ge 1$. Let $\hat{f}(\cdot)$ be the piecewise linear approximation. For any target i and observable feature configuration x_i, we have

$$\frac{\hat{f}(x_i)}{f(x_i)} \le v \le 1 + \frac{\epsilon^2}{2}.$$

Let x^* be the optimal observable features against the true score function f, and let x' be the optimal observable features to the above MILP. Let $U(\cdot)$ be the defender's expected loss, and $\hat{U}(\cdot)$ be the approximate defender's expected loss. For any observable feature configuration x, we have

$$\begin{aligned}
|\hat{U}(x) - U(x)| &= \left| \frac{\sum_i \hat{f}(x_i) u_i}{\sum_i \hat{f}(x_i)} - \frac{\sum_i f(x_i) u_i}{\sum_i f(x_i)} \right| \\
&= \left| \frac{\sum_i \hat{f}(x_i) u_i}{\sum_i \hat{f}(x_i)} - \frac{\sum_i \hat{f}(x_i) u_i}{\sum_i f(x_i)} + \frac{\sum_i \hat{f}(x_i) u_i}{\sum_i f(x_i)} - \frac{\sum_i f(x_i) u_i}{\sum_i f(x_i)} \right| \\
&\le \frac{2}{\sum_i f(x_i)} \left| \sum_i f(x_i) - \sum_i \hat{f}(x_i) \right| = 2 \left(\frac{\sum_i \hat{f}(x_i)}{\sum_i f(x_i)} - 1 \right) \\
&\le \epsilon^2
\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}
U(x') - U(x^*) &= U(x') - \hat{U}(x') + \hat{U}(x') - U(x^*) \\
&\le U(x') - \hat{U}(x') + \hat{U}(x^*) - U(x^*) \\
&\le 2\epsilon^2
\end{aligned}$$

□

B.4 Proof of Theorem 4.3

Suppose binary search terminates with interval of length $U - L \le \epsilon_{bs}$, and observable features x^{bs}. Both x^{bs} and the optimal observable features x' to the MILP lie in this interval. This means $\hat{U}(x^{bs}) - \hat{U}(\hat{x}) \le \epsilon_{bs}$. Recall that x^* is the optimal observable features against the true score function f. Therefore, we have

$$\begin{aligned}
U(x^{bs}) - U(x^*) &= U(x^{bs}) - \hat{U}(x^{bs}) + \hat{U}(x^{bs}) - U(x^*) \\
&\le U(x^{bs}) - \hat{U}(x^{bs}) + \hat{U}(\hat{x}) + \epsilon_{bs} - U(x^*) \\
&\le U(x^{bs}) - \hat{U}(x^{bs}) + \hat{U}(x^*) + \epsilon_{bs} - U(x^*) \\
&\le 2\epsilon^2 + \epsilon_{bs}
\end{aligned}$$

□

C Exact Algorithms for Special Cases

C.1 Deception cost on discrete features

In our first attempt at exact algorithms, we assume the cost of deception is only associated with discrete features, i.e. $\eta_{ik} = 0$ if k is a continuous feature.

Recall that we use $M_c \subseteq M$ to denote the set of continuous features, and $M_d = M - M_c$ denotes the set of discrete features. Suppose $x_i^d = (x_{ik})_{k \in M_d}$ and $x_i^c = (x_{ik})_{k \in M_c}$, and let $x_i = (x_i^d, x_i^c)$ be the observable features decomposed into discrete features and continuous features. Our score function $f(x_i) = \exp\{\sum_{k \in M} w_k x_{ik}\}$ can be factorized into $f(x_i) = f_d(x_i^d) f_c(x_i^c)$, where f_d, f_c are scores considering discrete and continuous features only, respectively.

Let $A_i^d = \{x_i^{d_j} : j = 1, \dots, k\}$ be the finite set of possible discrete observable feature combinations at target i. Each $x_i^{d_j} \in \{0,1\}^{m_d}$ is a m_d-dimensional vector, where $m_d = |M_d|$ is the number of discrete features in FDG. Based on the hidden discrete features $\hat{x}_i^d$ and the feasible regions $A(\hat{x}_{ik})$, we can compute both A_i^d and all possible scores $f_{ij}^d = f_d(x_i^{d_j})$. Similarly, we can compute the interval $[\alpha_i, \beta_i]$ in which the continuous score f_i^c could possibly lie, since each continuous feature x_{ik} can take value in an interval $A(\hat{x}_{ik})$.

Subsequently, we formulate the following mathematical program. The binary variable $y_{ij} = 1$ if target i's discrete observable features are the j-th combination in A_i^d, that is, $x_i^d = x_i^{d_j} \in A_i^d$. The cost c_{ij} for setting the discrete observable features to $x_i^{d_j}$ could be computed accordingly.

$$\begin{aligned}
\min_{y, f^c} \quad & \frac{\sum_{i \in N} \sum_{j=1}^k u_i f_{ij}^d y_{ij} f_i^c}{\sum_{i \in N} \sum_{j=1}^k f_{ij}^d y_{ij} f_i^c} & \\
s.t. \quad & \sum_{j=1}^k y_{ij} = 1 & \forall i \in N \\
& \sum_{i \in N} \sum_{j=1}^k c_{ij} y_{ij} \le B & \\
& f_i^c \in [\alpha_i, \beta_i] & \forall i \in N \\
& y_{ij} \in \{0,1\} & \forall i \in N, j \in [k]
\end{aligned}$$

We may apply the same linearization method as before to obtain a MILP.

Solving this MILP yields the optimal discrete feature configuration, as well as the optimal scores f_i^c's of the continuous features. One may then solve the system of linear equations $\ln f_i^c = \sum_{k \in M} w_k x_{ik}$ for $i \in N$ for the optimal continuous features. Since the feasible regions of the continuous features are connected, there exists at least one solution to the system of equations. We remark that when all features are continuous, the above approach finds the optimal feature configuration in polynomial time.

C.2 No budget and feasibility constraints

We present an efficient algorithm when the defender has no budget and feasibility constraints. [Schlenker *et al.*, 2018] proposed a greedy algorithm in a similar setting (with feasibility constraints), whose complexity is polynomial in the size of "type space", which is still exponential in the representation of FDG, not to mention that with a single continuous

feature the size of our "type space" becomes infinite. Furthermore, the probabilistic behavior of the attacker in FDG makes their key strategy inapplicable.

Since the defender aims at minimizing her expected loss, one intuitive idea is to give the lowest score to the target with the highest loss u_i. In fact, we show below that the defender should configure the features at each target in only two possible ways: the ones which maximizes or minimizes the score. First, we assume that the defender's losses have been sorted in ascending order $u_1 \leq u_2 \leq \cdots \leq u_n$.

Lemma C.1. *Let $x = (x_1, \ldots, x_n)$ be an optimal observable feature configuration. There exists a permutation σ on N where $x^\sigma = (x_{\sigma(1)}, \ldots, x_{\sigma(n)})$ is also an optimal observable feature configuration, and*

$$f(x_{\sigma(1)}) \geq f(x_{\sigma(2)}) \geq \cdots \geq f(x_{\sigma(n)}).$$

In particular, if $u_1 < u_2 < \cdots < u_n$, σ can be the identity permutation.

Proof. We prove by contradiction. Suppose $i < j$ and $f(x_i) < f(x_j)$. We have

$$\begin{aligned}&(f(x_j)u_i + f(x_i)u_j) - (f(x_i)u_i + f(x_j)u_j)\\ &= (f(x_j) - f(x_i))(u_i - u_j) \leq 0\end{aligned}$$

and the inequality is strict if $u_i < u_j$. Thus, when $u_i < u_j$, if we swap the features on target i and j, we would strictly improve the solution, which contradicts x being optimal. When $u_i = u_j$, we could swap the observed features on node i and j, and strictly decrease the number of score inversions. □

Lemma C.2. *There exists an optimal observable feature configuration $x = (x_1, \ldots, x_n)$ such that, for some $j \in N - \{n\}$, if $i \leq j$, $f(x_i) = \max_{x_i'} f(x_i')$; otherwise, $f(x_i) = \min_{x_i'} f(x_i')$.*

Proof. Let x be an optimal observable feature configuration. Fix a target $i \in N$. Consider an alternative configuration $\tilde{x}_i$ for target i, while keeping features of other targets unchanged. We have

$$\begin{aligned}&\frac{f(x_i)u_i + \sum_{j\neq i} f(x_j)u_j}{f(x_i) + \sum_{j\neq i} f(x_j)} - \frac{f(\tilde{x}_i)u_i + \sum_{j\neq i} f(x_j)u_j}{f(\tilde{x}_i) + \sum_{j\neq i} f(x_j)}\\ &\propto \left((f(x_i) - f(\tilde{x}_i))\left(\sum_{j\neq i} f(x_j)(u_i - u_j)\right)\right)\end{aligned}$$

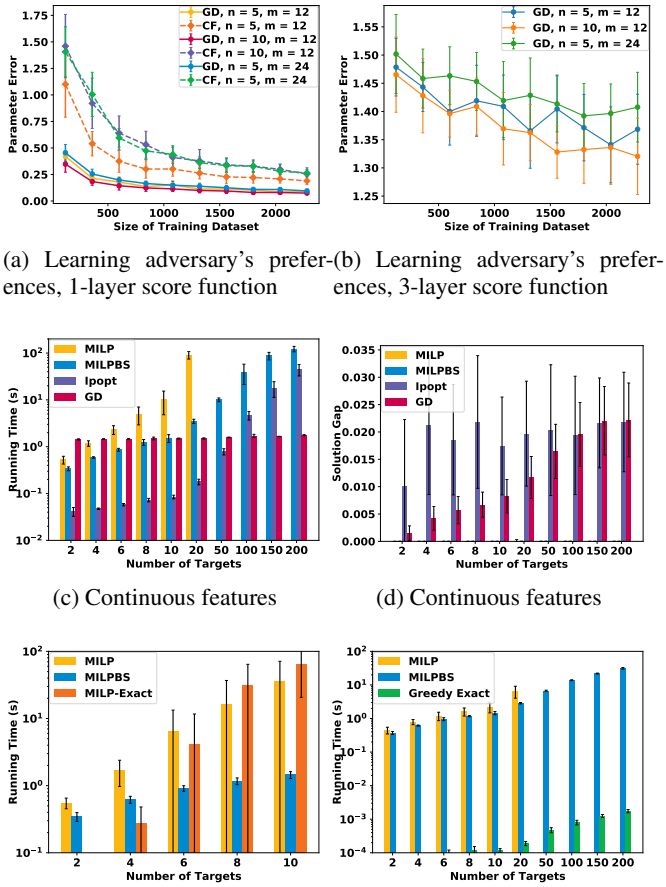
Depending on its sign, we could improve the solution x by making $f(\tilde{x}_i)$ greater or smaller than $f(x_i)$, and obviously we should take it to extreme by setting $f(\tilde{x}_i) = \max_{x_i'} f(x_i')$ or $f(\tilde{x}_i) = \min_{x_i'} f(x_i')$. Since we assumed x is optimal, then we know $\tilde{x} = (\tilde{x}_i, x_{-i})$ is also optimal, with $f(\hat{x}_i)$ at an extreme value. By Lemma C.1, we could permute the features in $\tilde{x}$ so that the scores are in decreasing order. After applying the above argument repeatedly, the score of each target achieves either the maximum or minimum score possible. Therefore, there exists some $j \in N - \{n\}$, such that

$$\begin{aligned}&\max_{x_i'} f(x_i') = f(x_1) = \cdots = f(x_j)\\ &\geq f(x_{j+1}) = \cdots = f(x_n) = \min_{x_i'} f(x_i')\end{aligned}$$

□

Theorem C.3. *The optimal feature configuration can be found in $O(n \log n + m)$ time.*

Proof. We can do an exhaustive search on the "cut-off" node j in Lemma C.2. With bookkeeping, the search can be done in $O(n)$ time. Since f is monotone in each observable feature, the maximum and minimum score can be found in $O(m)$ time. Sorting the targets' losses takes $O(n \log n)$ time. □



(a) Learning adversary's preferences, 1-layer score function

(b) Learning adversary's preferences, 3-layer score function

(c) Continuous features

(d) Continuous features

(e) Cost on discrete features, 1-layer score function, $m = 12$

(f) Cost on discrete features, 1-layer score function, $m = 12$

Figure 2: Experimental results

D Additional Experiments

D.1 Learning

In addition to the mean total variation distance reported in the main text, we present another metric to measure the performance of learning. We consider $|\hat{\theta} - \theta|$, the L_1 error in the score function parameter θ, which directly relates to the sample complexity bound in Theorem 3.1. Since the dimension of θ depends on the number of features k and other factors, we consider the L_1 error divided by the number of parameters and report this metric in Fig. 2a and Fig. 2b.

For a single layer score function, the log-likelihood is concave. Thus GD is expected to find the global maximizer. Indeed, we see that in Fig 2a, the learning error is close to zero, which corroborates this claim. The L_1 error for CF also decreases as the sample size increases, though not as small as GD. According to Theorem 3.1 we would need much more samples than 2000 to achieve an error of 0.25.

For complex score function, the learning error is larger as shown in Fig. 2b, even though Fig. 1b in the main text shows the total variation distance is small. This suggests that the loss surface for complex score function is, true to its name, more complex. Comparing Fig. 2a- 2b with Fig. 1i- 1k, we can obtain more intuition why the solution gap in Fig. 1k is much larger than that in Fig. 1i.

D.2 Planning

We present the performance of our algorithms on some special cases of FDG.

When all features are continuous, in addition to our MILP, we may use non-convex solver or GD as a heuristic to find optimal feature configurations. Fig. 2c shows that these two heuristics scale better than the approximation algorithms. In Fig. 1f and Fig. 1h, we showed that GD demonstrates the best solution quality among the heuristics on complex score functions. A natural question to ask is if GD is in practice close to exact. In Fig. 2d, we show that at least when we have a single-layer score function, GD solution is not far from optimal, though its solution deteriorates as the problem size grows. Non-convex solver yields relatively constant, and small, solution gap as well.

When deception cost is only associated with discrete features, we presented an exact MILP formulation in Appendix C.1. Fig. 2e shows that it is especially efficient on smaller problems. Yet as the problem size grows its efficiency decreases quickly.

Finally, in Appendix C.2, we proposed a $O(n \log n + m)$ time algorithm for FDG without budget and feasibility constraints. Indeed, as shown in Fig. 2f, the algorithm's running time is several magnitudes less than the MILP-based approaches.

E Experiment Parameter Distributions

Complex score function architecture The 3-layer neural network score function has input layer of size $m \times 24$, second layer 24×12, and third layer 12×1. The first and second layers are followed by a tanh activation, and the last layer is followed by an exponential function. The neural network parameters are initialized uniformly at random in $[-0.5, 0.5]$. We use this network architecture for all of our experiments.

FDG parameters for 1-layer score function We detail in Table 2 the parameter distributions used in the planning and combined learning and planning experiments, when the adversary assumes the single-layer score function. These distributions apply to the results shown in Fig. 1c, 1d, 1i, 1j.

FDG parameters for 3-layer score function We detail in Table 3 the parameter distributions used in the planning and combined learning and planning experiments, when the adversary assumes the 3-layer score function. These distributions apply to the results shown in Fig. 1e,1f, 1g, 1h,1k, 1l.

Discrete feature $k \in M_d$		Continuous feature $k \in M_c$	
Variable	Distribution	Variable	Distribution
$\lvert M_d \rvert$	$2m/3$	$\lvert M_c \rvert$	$m/3$
η_{ik}	$U(-3, 3)$	η_{ik}	$U(0, 3)$
τ_{ik}		τ_{ik}	$U(0, 0.25)$
$\hat{x}_{ik}$	$U\{0, 1\}$	$\hat{x}_{ik}$	$U(0, 1)$
u_i	$U(0, 1)$		
B	$U(0, 0.2C_{\max}$		
$C_{\max}$	$\sum_{i \in N} \sum_{k \in M_c} \eta_{ik} \min(\hat{x}_{ik}, 1 - \hat{x}_{ik}, \tau_{ik}) + \sum_{k \in M_d} \eta_{ik}$		

Table 2: FDG parameter distributions for experiments on 1-layer attacker score function. Used in Fig. 1c, 1d, 1i, 1j

Variable	Distribution
η_{ik}	$U(0, 1)$
τ_{ik}	1
$\hat{x}_{ik}$	$U(0, 1)$
u_i	$U(0, 1)$
B	$U(0, 0.2nm)$

Table 3: FDG parameter distributions for experiments on 3-layer attacker score function. Used in Fig. 1e,1f, 1g, 1h,1k, 1l