Security Games with Information Leakage: Modeling and Computation

Haifeng Xu [*] Albert X. Jiang [†] Arunesh Sinha [*]

Zinovi Rabinovich [‡] Shaddin Dughmi [*] Milind Tambe [*]

Abstract

Most models of Stackelberg security games assume that the attacker only knows the defender's mixed strategy, but is not able to observe (even partially) the instantiated pure strategy. Such partial observation of the deployed pure strategy – an issue we refer to as *information leakage* – is a significant concern in practical applications. While previous research on patrolling games has considered the attacker's real-time surveillance, our settings, therefore models and techniques, are fundamentally different. More specifically, after describing the information leakage model, we start with an LP formulation to compute the defender's optimal strategy in the presence of leakage. Perhaps surprisingly, we show that a key subproblem to solve this LP (more precisely, the defender oracle) is NP-hard *even* for the simplest of security game models. We then approach the problem from three possible directions: efficient algorithms for restricted cases, approximation algorithms, and heuristic algorithms for sampling that improves upon the status quo. Our experiments confirm the necessity of handling information leakage and the advantage of our algorithms.

[*]University of Southern California, Los Angeles, USA; {haifengx,aruneshs,shaddin,tambe}@usc.edu. Shaddin Dughmi is supported in part by NSF CAREER Award CCF-1350900. Haifeng Xu is supported by NSF grant CCF-1350900, MURI grant W911NF-11-1-0332 and US-Naval Research grant Z14-12072.

[†]Trinity University, San Antonio, USA; xjiang@trinity.edu.

[‡]Independent Researcher, Israel; zr@zinovi.net.

1 Introduction

Stackelberg security games played between a defender (leader) and an attacker (follower) have been widely studied in the past few years [10, 12, 11, 16, 4]. Most models, in particular, including all the deployed security systems in [16], assume that the attacker is not able to observe (even partially) the defender's instantiated pure strategy (i.e., which targets are being protected), thus he makes decisions based only on his knowledge of the defender's mixed strategy. This fails to capture the attacker's real-time surveillance, by which he may *partially* observe the deployed pure strategy. For example, the attacker may observe the protection status of a certain target while approaching for an attack; or in some security domains information regarding the protection status of certain targets may leak to the attacker due to real-time surveillance or even an insider threat; further, well-prepared attackers may approach certain adversarially chosen target to collect information before committing an attack.

Unfortunately, this problem– an issue we refer to as *information leakage* – has not received much attention in Stackelberg security games. In the literature of patrolling games, attackers' real-time surveillance is indeed considered [2, 1, 3, 5, 6, 18]. However, all these papers study settings of patrols carried out over space and time, i.e., the defender follows a schedule of visits to multiple targets over time. In addition, they assume that it takes time for the attacker to execute an attack, during which the defender can interrupt the attacker by visiting the attacked target. Therefore, even if the attacker can fully observe the current position of the defender (in essence, status of *all* targets), he may not have enough time to complete an attack on a target before being interrupted by the defender. The main challenge there is to create patrolling schedules with the smallest possible time between any two target visits. In contrast, we consider information leakage in standard security game models, where the attack is *instantaneous* and cannot be interrupted by the defender's resource re-allocation. Furthermore, as may be more realistic in our settings, we assume that information is leaked from a limited number of targets. As a result, our setting necessitates novel models and techniques. We also provide efficient algorithms with complexity analysis.

This paper considers the design of optimal defender strategy in the presence of *partial* information leakage. Considering that real-time surveillance is costly in practice, we explicitly assume that information leaks from *only one* target, though our model and algorithms can be generalized. We start from the basic security game model where the defender allocates k resources to protect n targets without any scheduling constraint. Such models have applications in real security systems like ARMOR for LAX airport and GUARDS for airports in general [16]. We first show via a concrete example in Section 2 how ignoring information leakage can lead to significant utility loss. This motivates our design of optimal defending strategy given the possibility of information leakage. We start with a linear program formulation. However, perhaps surprisingly, we show that it is difficult to solve the LP *even* for this basic case, whereas the optimal mixed strategy without leakage can be computed easily. In particular, we show that the defender oracle, a key subproblem used in the column generation technique employed for most security games, is NP-hard. This shows the intrinsic difficulty of handling information leakage. We then approach the problem from three directions: efficient algorithms for special cases, approximation algorithms and heuristic algorithms for sampling that improves upon the status quo. Our experiments support our hypothesis that ignoring information leakage can result in significant loss of utility for the defender, and demonstrates the value of our algorithms.

2 Model of Information Leakage

Consider a standard zero-sum Stackelberg security game with a defender and an attacker. The defender allocates k security resources to protect n targets, which are denoted by the set $[n] = \{1, 2, ..., n\}$. In this paper we consider the case where the security resources do *not* have scheduling constraints. That is, the defender's pure strategy is to protect any subset of $[n]$ of size at most k. For any $i \in [n]$, let r_i be the reward [c_i be the cost] of the defender when the attacked target i is protected [unprotected]. We consider zero-sum games, therefore the attacker's utility is the negation of the defender's utility. Let s denote a pure strategy and S be the set of all possible pure strategies. With some abuse of notation, we sometimes regard s as a *subset* of $[n]$ denoting the protected targets; and sometimes view it as an n-dimensional $0-1$ *vector* with k 1's specifying the protected targets. The intended interpretation should be clear from context. The *support* of a mixed strategy is defined to be the set of pure strategies with non-zero probabilities. Without information leakage, the problem of computing the defender's optimal mixed strategy can be compactly formulated as linear program (1) with each variable x_i as the marginal probability of covering target i. The resulting marginal vector $\vec{x}$ is a convex combination of the indicator vectors of pure strategies, and a mixed strategy with small support can be efficiently sampled, e.g., by Comb Sampling [17].

$$\begin{array}{lll} \text{maximize} & u & \\ \text{subject to} & u \le r_i x_i + c_i(1 - x_i), & \text{for } i \in [n]. \\ & \sum_{i \in [n]} x_i \le k & \\ & 0 \le x_i \le 1, & \text{for } i \in [n]. \end{array} \tag{1}$$

Building on this basic security game, our model goes one step further and considers the possibility that the protection status of one target leaks to the attacker. Here, by "protection status" we mean whether this target is protected or not in an *instantiation* of the mixed strategy. We consider two related models of information leakage:

1. **PR**obabilistic **I**nformation **I**eakage (PRIL): with probability $p_i (\ge 0)$ a *single* target i leaks information; and with probability $p_0 = 1 - \sum_{i=1}^{n} p_i$ no targets leak information. So we have $\vec{p} = (p_0, p_1, ..., p_n) \in \Delta_{n+1}$ where Δ_{n+1} is the $(n+1)$-dimensional simplex. In practice, $\vec{p}$ is usually given by domain experts and may be determined by the nature or property of targets.

2. **AD**versarial **I**nformation **L**eakage (ADIL): with probability $1 - p_0$, one *adversarially* chosen target leaks information. This model captures the case where the attacker will strategically choose a target for surveillance and with certain probability he succeeds in observing the protection status of the surveyed target.

Given either model – PRIL with any $\vec{p} \in \Delta_{n+1}$ or ADIL – we are interested in computing the optimal defender patrolling strategy. The first question to ask is: why does the issue of information leakage matter and how does it affect the computation of the optimal defender strategy? To answer this question we employ a concrete example.

Consider a zero-sum security game with 4 targets and 2 resources. The profiles of reward r_i [cost c_i] is $\vec{r} = (1, 1, 2, 2)$ [$\vec{c} = (-2, -2, -1, -1)$], where the coordinates are indexed by target ids. If there is no information leakage, it is easy to see that the optimal marginal coverage is $\vec{x} = (\frac{2}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3})$. The attacker will attack an arbitrary target, resulting in a defender utility of 0. Now, let us consider a simple case of information leakage. Assume the attacker observes whether target 1 is protected or not in any instantiation of the mixed strategy, i.e., $p_1 = 1$. As we will argue, how the marginal probability $\vec{x}$ is implemented would matter now. One way to implement $\vec{x}$ is to protect target $\{1, 2\}$

with probability $\frac{2}{3}$ and protect $\{3,4\}$ with probability $\frac{1}{3}$. However, this implementation is "fragile" in the presence of the above information leakage. In particular, if the attacker observes that target 1 is protected (which occurs with probability $\frac{2}{3}$), he infers that the defender is protecting target $\{1,2\}$ and will attack 3 or 4, resulting in a defender utility of -1; if target 1 is not protected, the attacker will just attack, resulting in a defender utility of -2. Therefore, the defender gets expected utility $-\frac{4}{3}$.

Now consider another way to implement the *same* marginal $\vec{x}$ by the following mixed strategy:

$\{1,2\}$	$\{1,3\}$	$\{1,4\}$	$\{2,3\}$	$\{2,4\}$	$\{3,4\}$
10/27	4/27	4/27	4/27	4/27	1/27

If the attacker observes that target 1 is protected (which occurs with probability $\frac{2}{3}$), then he infers that target 2 is protected with probability $\frac{\frac{10}{27}}{\frac{10}{27}+\frac{4}{27}+\frac{4}{27}} = \frac{5}{9}$, and target $3,4$ are both protected with probability $\frac{2}{9}$. Some calculation shows that the attacker will have the same utility $\frac{1}{3}$ on target $2,3,4$ and thus will choose an arbitrary one to attack, resulting in a defender utility of $-\frac{1}{3}$. On the other hand, if target 1 is observed to be unprotected, the defender gets utility -2. In expectation, the defender gets utility $\frac{2}{3} \times (-\frac{1}{3}) + \frac{1}{3} \times (-2) = -\frac{8}{9}$.

As seen above, though implementing the same marginals, the latter mixed strategy achieves better defender utility than the former one in the presence of information leakage. However, is it optimal? It turns out that the following mixed strategy achieves an even better defender utility of $-\frac{1}{3}$, which can be proved to be optimal: protect $\{1,2\}$ with probability $\frac{5}{9}$, $\{1,3\}$ with probability $\frac{2}{9}$ and $\{1,4\}$ with probability $\frac{2}{9}$.

This example shows that compact representation by marginal coverage probabilities is not sufficient for computing the optimal defending strategy assuming information leakage. This naturally raises new computational challenges: how can we formulate the defender's optimization problem and compute the optimal solution? Is there still a compact formulation or is it necessary to enumerate all the exponentially many pure strategies? What is the computational complexity of this problem? We answer these questions in the next section.

3 Computing Optimal Defender Strategy

We will focus on the derivation of the PRIL model. The formulation for the ADIL model is provided at the end of this section since it admits a similar derivation. Fixing the defender's mixed strategy, let t_i ($\neg t_i$) denote the event that target i is *protected* (*unprotected*). For the PRIL model, the defender's utility equals

$$DefU = p_0 u + \sum_{i=1}^{n} p_i (u_i + v_i)$$

where $u = \min_j [r_j \mathbf{Pr}(t_j) + c_j \mathbf{Pr}(\neg t_j)]$ is the defender's utility when there is no information leakage; and

$$\begin{aligned} u_i &= \mathbf{Pr}(t_i) \times \min_j [r_j \mathbf{Pr}(t_j|t_i) + c_j \mathbf{Pr}(\neg t_j|t_i)] \\ &= \min_j [r_j \mathbf{Pr}(t_j, t_i) + c_j \mathbf{Pr}(\neg t_j, t_i)] \end{aligned}$$

is the defender's utility when target i leaks out its protection status as t_i (i.e., protected) multiplied by probability $\mathbf{Pr}(t_i)$. Similarly

$$v_i = \min_j [r_j \mathbf{Pr}(t_j, \neg t_i) + c_j \mathbf{Pr}(\neg t_j, \neg t_i)]$$

is the defender's expected utility multiplied by probability $\mathbf{Pr}(\neg t_i)$ when target i leaks status $\neg t_i$ (i.e., unprotected)

Define variables $x_{ij} = \mathbf{Pr}(t_i, t_j)$ (setting $x_{ii} = \mathbf{Pr}(t_i)$). Using the fact that $\mathbf{Pr}(t_i, \neg t_j) = x_{ii} - x_{ij}$ and $\mathbf{Pr}(\neg t_i, \neg t_j) = 1 - x_{ii} - x_{jj} + x_{ij}$, we obtain the following linear program which computes the defender's optimal patrolling strategy:

$$\begin{array}{lll} \text{maximize} & p_0 u + \sum_{i=1}^{n} p_i(u_i + v_i) & \\ \text{subject to} & u \le r_j x_{jj} + c_j(1 - x_{jj}), & \text{for } j \in [n]. \\ & u_i \le r_j x_{ij} + c_j(x_{ii} - x_{ij}), & \text{for } i, j \in [n]. \\ & v_i \le r_j(x_{jj} - x_{ij}) + c_j(1 - x_{ii} - x_{jj} + x_{ij}), & \text{for } i, j \in [n]. \\ & x_{ij} = \sum_{s: i,j \in s} \theta_s, & \text{for } i, j \in [n]. \\ & \sum_{s \in S} \theta_s = 1 & \\ & \theta_s \ge 0, & \text{for } s \in S. \end{array} \tag{2}$$

where $u, u_i, v_i, x_{ij}, \theta_s$ are variables; s denotes a pure strategy and the sum condition "$s : i, j \in s$" means summing over all the pure strategies that protect both targets i and j (or i if $i = j$); θ_s denotes the probability of choosing strategy s.

Unfortunately, LP (2) suffers from an exponential explosion of variables, specifically, θ_s. From the sake of computational efficiency, one natural idea is to find a compact representation of the defender's mixed strategy. As suggested by LP (2), the variables x_{ij}, indicating the probability that targets i, j are both protected, are sufficient to describe the defender's objective and the attacker's incentive constraints.

Let us call variables x_{ij} the *pair-wise marginals* and think of them as a matrix $X \in \mathbb{R}^{n \times n}$, i.e., the i'th row and j'th column of X is x_{ij} (not to be confused with the *marginals* $\vec{x}$). We say X is *feasible* if there exists a mixed strategy, i.e., a distribution over pure strategies, that achieves the pair-wise marginals X. Clearly, not all $X \in \mathbb{R}^{n \times n}$ are feasible. Let $\mathcal{P}(n, k) \in \mathbb{R}^{n \times n}$ be the set of all *feasible* X. The following lemma shows a structural property of $\mathcal{P}(n, k)$.

Lemma 1. *$\mathcal{P}(n, k)$ is a polytope and any $X \in \mathcal{P}(n, k)$ is a* symmetric positive semi-definite *(PSD) matrix.*

Proof. Notice that X is feasible if and only there exists θ_s for any pure strategy s such that the following linear constraints hold:

$$\begin{array}{ll} x_{ij} = \sum_{s: i,j \in s} \theta_s, & \text{for } i, j \in [n]. \\ \sum_{s \in S} \theta_s = 1 & \\ \theta_s \ge 0, & \text{for } s \in S. \end{array} \tag{3}$$

These constraints define a polytope for variables $(X, \vec{\theta})$, therefore its projection to the lower dimension X, which is precisely $\mathcal{P}(n, k)$, is also a polytope.

To prove $X \in \mathcal{P}(n, k)$ is PSD, we first show that any vertex of $\mathcal{P}(n, k)$, i.e., a pure strategy, is PSD. In fact, let $s \in \{0, 1\}^n$ be any pure strategy, then the pair-wise marginal w.r.t. s is $X_s = ss^T$, which is PSD. Therefore, any $X \in \mathcal{P}$, which is a convex combination of its vertexes, is also PSD. □

$$\begin{array}{lll} \text{maximize} & p_0 u + \sum_{i=1}^{n} p_i(u_i + v_i) & \\ \text{subject to} & u \le r_j x_{jj} + c_j(1 - x_{jj}), & \text{for } j \in [n]. \\ & u_i \le r_j x_{ij} + c_j(x_{ii} - x_{ij}), & \text{for } i, j \in [n]. \\ & v_i \le r_j(x_{jj} - x_{ij}) + c_j(1 - x_{ii} - x_{jj} + x_{ij}), & \text{for } i, j \in [n]. \\ & X \in \mathcal{P}(n, k) & \end{array} \tag{4}$$

With Lemma 1, we may re-write LP (2) compactly as LP (4) with variables u, u_i, v_i and X. Therefore, we would be able to compute the optimal strategy efficiently in polynomial time if the constraints determining the polytope $\mathcal{P}(n,k)$ were only polynomially many – recall that this is the approach we took with LP (1) in the case of no information leakage. However, perhaps surprisingly, the problem turns out to be much harder in the presence of leakage.

Lemma 2. *Optimizing over $\mathcal{P}(n,k)$ is NP-hard.*

Proof. We prove by reduction from the densest k-subgraph problem. Given any graph instance $G=(V,E)$, let A be the adjacency matrix of G. Consider the following linear program:

$$\begin{array}{ll} \text{maximize} & \sum_{i,j\in[n]} A_{ij}x_{ij} \\ \text{subject to} & X \in \mathcal{P}(n,k). \end{array} \tag{5}$$

This linear program must have a *vertex* optimal solution X^* which satisfies $X^* = ss^T$ for some pure strategy $s \in \{0,1\}^n$. Therefore, the linear objective satisfies

$$\sum_{i,j\in[n]} A_{ij}x_{ij} = tr(AX^*) = tr(A \times ss^T) = tr(s^TAs) = s^TAs.$$

Notice that $s^TAs/2k$ equals the density of a subgraph of G with k nodes indicated by s. Since X^* is the optimal solution to LP (5), it also maximizes the density $s^TAs/2k$ over all subgraphs with k nodes. In other words, the ability of optimizing LP (5) implies the ability of computing densest k-subgraph, which is NP-hard. Therefore, optimizing over $\mathcal{P}(n,k)$ is NP-hard. □

Lemma 2 suggests that there is no hope of finding polynomially many linear constraints which determine $\mathcal{P}(n,k)$ or, more generally, an efficient separation oracle for $\mathcal{P}(n,k)$, assuming $P \neq NP$. In fact, $\mathcal{P}(n,k)$ is closely related to a fundamental geometric object, known as the *correlation polytope*, which has applications in quantum mechanics, statistics, machine learning and combinatorial problems. We show a connection between $\mathcal{P}(n,k)$ and correlation polytope in Appendix B. For further information, we refer the reader to [14].

Another approach for computing the optimal defender strategy is to use the technique of column generation, which is a master/slave decomposition of an optimization problem. The essential part of this approach is the slave problem, which is also called the "defender best response oracle" or "defender oracle" for short [7]. We defer the derivation of the defender oracle to Appendix A, while only mention that a similar reduction as in Lemma 2 also implies the follows.

Lemma 3. *The defender oracle is NP-hard.*

By now, we have shown the evidence of the difficulty of solving LP (2) using either marginals or the technique of column generation. For the ADIL model, a similar argument yields that the following LP formulation computes the optimal defender strategy. It is easy to check that it shares the same marginals and defender oracle as the PRIL model.

$$\begin{array}{lll} \text{maximize} & p_0u + (1-p_0)w & \\ \text{subject to} & u \le r_jx_{jj} + c_j(1-x_{jj}), & \text{for } j \in [n]. \\ & u_i \le r_jx_{ij} + c_j(x_{ii} - x_{ij}), & \text{for } i,j \in [n]. \\ & v_i \le r_j(x_{jj} - x_{ij}) + c_j(1 - x_{ii} - x_{jj} + x_{ij}), & \text{for } i,j \in [n]. \\ & w \le u_i + v_i, & \text{for } i \in [n]. \\ & X \in \mathcal{P}(n,k) & \end{array} \tag{6}$$

where variable w is the defender's expected utility when an adversarially chosen target is observed by the attacker.

3.1 Leakage from Small Support of Targets

Despite the hardness results for the general case, we show that the defender oracle admits a polynomial time algorithm if information only leaks from a small subset of targets; we call this set the *leakage support*. By re-ordering the targets, we may assume without loss of generality that only the first m targets, denoted by set $[m]$, could possibly leak information in both the PRIL and ADIL model. For the PRIL model, this means $p_i = 0$ for any $i > m$ and for the ADIL model, this means the attacker only chooses a target in $[m]$ for surveillance.

Why does this make the problem tractable? Intuitively the reason is as follows: when information leaks from a small set of targets, we only need to consider the correlations between these leaking targets and others, which is a much smaller set of variables than in LP (2) or (6). Restricted to a leakage support of size m, the defender oracle is the following problem (See Appendix A for the derivation). Let A be a *symmetric* matrix of the following block form

$$A : \begin{bmatrix} A_{mm} & A_{mm'} \\ A_{m'm} & A_{m'm'} \end{bmatrix} \tag{7}$$

where $m' = n - m$; $A_{mm'} \in \mathbb{R}^{m \times m'}$ for any integers m, m' is a sub-matrix and, crucially, $A_{m'm'}$ is a *diagonal* matrix. Given A of form (7), find a pure strategy s such that $s^T A s$ is maximized. That is, the defender oracle identifies the size-k principle submatrix with maximum entry sum for any A of form (7). Note that $m = n$ in general case.

Before detailing the algorithm, we first describe some notation. Let $A[i,:]$ be the i'th row of matrix A and $diag(A)$ be the vector consisting of the diagonal entries of A. For any subset C_1, C_2 of $[n]$, let A_{C_1,C_2} be the submatrix of A consisting of rows in C_1 and columns in C_2, and $sum(A_{C_1,C_2}) = \sum_{i \in C_1, j \in C_2} A_{ij}$ be the entry sum of A_{C_1,C_2}. The following lemma shows that Algorithm 1 solves the defender oracle. Our main insight is that for a pure strategy s to be optimal, once the set $C = s \cap [m]$ is decided, its complement $\overline{C} = s \setminus C$ can be explicitly identified, therefore we can simply brute-force search to find the best $C \subseteq [m]$. Lemma 4 provides the algorithm guarantee, which then yields the polynomial solvability for the case of small m (Theorem 1).

Lemma 4. *Let m be the size of the leakage support. Algorithm 1 solves the defender oracle and runs in $poly(n, k, 2^m)$ time. In particular, the defender oracle admits a $poly(n, k)$ time algorithm if m is a constant.*

Proof. First, it is easy to see that Algorithm 1 runs in $poly(2^m, n, k)$ time since the for-loop is executed at most 2^m times. We show that it solves the defender oracle problem.

Let s, a set of k targets, be the pure strategy that indicates the size-k principle submatrix of A with maximum entry sum. Let $C = s \cap [m]$ and $\overline{C} = s \setminus C$. We claim that, given C, $\overline{C}$ must be the set of the indexes of the largest $k - |C|$ values from the set $\{v_{m+1}, ..., v_n\}$, where $\vec{v}$ is defined as $\vec{v} = 2\sum_{i \in C} A[i,:] + diag(A)$. In other words, if we know C, the set $\overline{C}$ can be easily identified. To prove the claim, we re-write the $sum(A_{s,s})$:

$$\begin{aligned} & sum(A_{s,s}) \\ = \; & sum(A_{C,C}) + 2sum(A_{C,\overline{C}}) + sum(A_{\overline{C},\overline{C}}) \\ = \; & sum(A_{C,C}) + 2sum(A_{C,\overline{C}}) + sum(diag(A_{\overline{C},\overline{C}})) \\ = \; & sum(A_{C,C}) + sum(2\sum_{i \in C} A_{i,\overline{C}} + diag(A_{\overline{C},\overline{C}})) \\ = \; & sum(A_{C,C}) + sum(v_{\overline{C}}) \\ = \; & val_C \end{aligned}$$

where $\vec{v} = 2\sum_{i\in C} A[i,:] + diag(A)$ and $v_{\overline{C}}$ is the sub-vector of v with indexes from $\overline{C}$. Given C, $sum(A_{C,C})$ is fixed, therefore $\overline{C}$ must be the set of indexes of the largest $k - |C|$ elements from $\{v_{m+1}, ..., v_n\}$. Algorithm 1 then loop over all the possible $C \subseteq [m]$ (2^m many) and identifies the best one, i.e., achieving the maximum val_C. □

Algorithm 1 Defender Oracle

Input: matrix A of form (7).

Output: a pure strategy s.

1: **for** all $C \subseteq [m]$ constrained by $|C| \leq k$ **do**
2: $\vec{v} = 2\sum_{i\in C} A[i,:] + diag(A)$;
3: Choose the largest $k - |C|$ values from the set $\{v_{m+1}, ..., v_n\}$, and denote the set of their indices as $\overline{C}$;
4: Set $val_C = sum(A_{C,C}) + sum(v_{\overline{C}})$;
5: **end for**
6: **return** the pure strategy $s = C \cup \overline{C}$ with maximum val_C.

Theorem 1. *(***Polynomial Solvability***) There is an efficient poly(n, k) time algorithm which computes the optimal defender strategy, if m is a constant.*

3.2 An Approximation Algorithm

We now consider approximation algorithms. Recall that information leakage is due to the correlation between targets, thus one natural way to minimize leakage is to allocate each resource *independently* with certain distributions. Naturally, the normalized marginal $\vec{x}^*/k$ becomes a choice, where $\vec{x}^*$ is the solution to LP (1). To avoid the waste of using multiple resources to protect the same target, we sample without replacement. Formally, the *independent sampling without replacement* algorithm proceeds as follows: 1. compute the optimal solution $\vec{x}^*$ of LP (1); 2. independently sample k elements from $[n]$ *without replacement* using distribution $\vec{x}^*/k$.

Zero-sum games exhibit negative utilities, therefore an approximation ratio in terms of utility is not meaningful. To analyze the performance of this algorithm we shift all the payoffs by a constant, $-\min_i c_i$, and get an equivalent constant-sum game with all non-negative payoffs. Theorem 2 shows that this algorithm is "almost" a $(1-\frac{1}{e})-$*approximation* to the optimal solution in the PRIL model, assuming information leaks out from any target i with equal probability $p_i = \frac{1-p_0}{n}$. We note that proving a general approximation ratio for any $\vec{p} \in \Delta_{n+1}$ turns out to be very challenging, intuitively because the optimal strategy adjusts according to different $\vec{p}$ while the sampling algorithm does not depend on $\vec{p}$. However, experiments empirically show that the ratio does not vary much for different $\vec{p}$ on average (see Section 5).

Theorem 2. *Assume each target leaks information with equal probability $p_i = \frac{1-p_0}{n}$. Let $\overline{c}_i \geq 0$ be the shifted cost and $U_{indepSample}$ be the defender utility achieved by* independent sampling without replacement*. Then we have:*

$$U_{indepSample} \geq (\tfrac{k-2}{k-1} - \tfrac{1}{e}) \left[Opt(LP\ 2) - (1-p_0)\tfrac{\sum_{i=1}^n \overline{c}_i}{n} \right].$$

Proof of Theorem 2

Let $Y = Y(\vec{x}) \in \mathbb{R}^{n\times n}$ be a function of any $\vec{x} \in \mathbb{R}^n$, where y_{ij} is the probability that target i, j are both protected using independent sampling *without* replacement. We first prove a lemma, which

provides a lower bound regarding how good the marginals and conditional marginals approximate the given marginal $\vec{x}$. Notice that Y does not have a compact close form in terms of $\vec{x}$ if we sample without replacement. Our proof is based on a coupling argument by relating the algorithm to independent sampling *with* replacement.

Lemma 5. *Given $\vec{x}$, $Y = Y(\vec{x})$ satisfies the following (in)equalities:*

$$\sum_{i\in[n]} y_{ii} = k; \tag{8}$$

$$y_{ii} \geq (1 - \frac{1}{e})x_i, \forall i \in [n]; \tag{9}$$

$$\frac{y_{ij}}{y_{ii}} \geq (\frac{k-2}{k-1} - \frac{1}{e})x_j, \forall i, j. \tag{10}$$

Proof. The first equation is easy to see, since each sampled pure strategy has k different targets due to sampling without replacement. To prove the other two inequalities, we instead consider independent sampling *with* replacement. Similarly, define function $Z = Z(\vec{x}^*) \in R^{n\times n}$ to be the matrix, where entry z_{ij} is the probability that target i, j are protected together when sampling with replacement. Contrast to Y, Z has succinct close form. In particular, we can first lower bound z_{ii}.

$$\begin{aligned} z_{ii} &= 1 - (1 - x_i/k)^k \\ &\geq 1 - e^{-x_i} \\ &\geq (1 - \frac{1}{e})x_i. \end{aligned}$$

where we used the fact $(1-\epsilon)^{\frac{1}{\epsilon}} \leq e^{-1}$ for any $\epsilon \in (0, 1)$. Now we lower bound z_{ij}/z_{ii} as follows.

$$\begin{aligned} \frac{z_{ij}}{z_{ii}} &= \frac{1 - (1 - \frac{x_i}{k})^k - (1 - \frac{x_j}{k})^k + (1 - \frac{x_i}{k} - \frac{x_j}{k})^k}{1 - (1 - x_i/k)^k} \\ &= 1 - (1 - \frac{x_j}{k})^k - \frac{(1 - \frac{x_i}{k})^k(1 - \frac{x_j}{k})^k - (1 - \frac{x_i}{k} - \frac{x_j}{k})^k}{1 - (1 - x_i/k)^k} \\ &\geq (1 - \frac{1}{e})x_j - \frac{(1 - \frac{x_i}{k})^k}{1 - (1 - x_i/k)^k}[(1 - \frac{x_j}{k})^k - (1 - \frac{x_j}{k - x_i})^k] \\ &\geq (1 - \frac{1}{e})x_j - \frac{e^{-x_i}}{1 - e^{-x_i}}[(1 - \frac{x_j}{k})^k - (1 - \frac{x_j}{k - x_i})^k] \end{aligned}$$

We now upper-bound $(1 - \frac{x_j}{k})^k - (1 - \frac{x_j}{k-1})^k$ using the formula $a^k - b^k = (a - b)\sum_{i=0}^{k-1} a^i b^{k-1-i}$, as follows

$$\begin{aligned} & (1 - \frac{x_j}{k})^k - (1 - \frac{x_j}{k - x_i})^k \\ = \; & (1 - \frac{x_j}{k} - 1 + \frac{x_j}{k - x_i})\sum_{t=0}^{k-1}(1 - \frac{x_j}{k})^t(1 - \frac{x_j}{k - x_i})^{k-1-t} \\ \leq \; & \frac{x_j x_i}{k(k - x_i)} \times k \\ \leq \; & \frac{x_i x_j}{k - 1} \end{aligned}$$

Plugging in the above upper bound, we thus have

$$\begin{aligned}\frac{z_{ij}}{z_{ii}} &\geq (1-\frac{1}{e})x_j - \frac{e^{-x_i}}{1-e^{-x_i}}\frac{x_i x_j}{k-1} \\ &\geq (1-\frac{1}{e})x_j - \frac{x_i}{e^{x_i}-1}\frac{x_j}{k-1} \\ &\geq (1-\frac{1}{e})x_j - \frac{x_j}{k-1} \\ &= (\frac{k-2}{k-1}-\frac{1}{e})x_j\end{aligned}$$

where the last inequality is due to the fact that $f(x) = \frac{x}{e^x-1}$ is a decreasing function for $x \in (0,1)$ and is upper bounded by $\lim_{x\to 0}\frac{x}{e^x-1} = 1$.

Therefore, we have $\frac{z_{ij}}{z_{ii}} \geq (\frac{k-2}{k-1}-\frac{1}{e})x_j$. We then conclude our proofs by claiming that $y_{ii} \geq z_{ii}$ and $y_{ij}/y_{ii} \geq z_{ij}/z_{ii}$.

To prove our claim, we use a coupling argument. Consider the following two stochastic process (StoP):

1. $StoP^1$: independently sample a random value i_t ($\in [n]$) with probability x_{i_t}/k for any $t = 1, 2, ...$ until precisely k *different* elements from $[n]$ show up.

2. $StoP^2$: independently sample a random value i_t ($\in [n]$) with probability x_{i_t}/k for $t = 1, 2, ...k$.

Let C^1 (C^2) denote all the possible random sequences generated by $StoP^1$ ($StoP^2$), and C_i^1 (C_i^2) denote the subset of C^1 (C^2), which consists of all the sequences including at least one i. For any $e \in C_i^2$, let C_e be the subset of sequences in C^1, whose first k element is precisely e. Notice that any sequence in C^1 has at least length of k while any sequence in C^2 has precisely k elements. Furthermore, $C_e \subseteq C_i^1$ and $C_e \cap C_{e'}$ for any $e, e' \in C_i^2$ and $e \neq e'$.

Now, think of each sequence as an event of the stochastic process. Notice that $P(e; StoP^2) = P(C_e; StoP^1)$ due to the independence of the sampling procedure, therefore, we have

$$\begin{aligned}P(C_i^2; StoP^2) &= \sum_{e\in C_i^2} P(e; StoP^2) \\ &= \sum_{e\in C_i^2} P(C_e; StoP^1) \\ &\leq P(C_i^1; StoP^1)\end{aligned}$$

However, $P(C_i^1|StoP^1) = y_{ii}$ and $P(C_i^2|StoP^2) = z_{ii}$. This proves $y_{ii} \geq z_{ii}$.

Notice that $y_{ij}/y_{ii} \geq z_{ij}/z_{ii}$ is equivalent to $P(e \in C_j^2 | e \in C_i^2; StoP^2) \geq P(e \in C_j^1 | e \in C_i^1; StoP^1)$. To prove this inequality, we claim that it is without loss of generality to assume the first sample is i in both processes. This is because, if the first i shows up at the t'th sample, moving i to the first position would not change the probability of the sequence due to independence between each sampling step. Conditioned on i is sampled first, a similar argument as above shows that the probability of Stochastic process $StoP^1$ generating j is at least the probability of stochastic process $StoP^2$ generating j. □

Let $\vec{x}^*$ be the optimal solution to LP (2) and U^* be the corresponding objective value – the defender optimal utility with no leakage. To prove Theorem 2, we start from comparing $OPT(LP\ 2)$ with U^*. From the objective of LP (2), we know that $u \leq U^*$, $u_i \leq U^*$ since U^* is the best possibly

utility using k resources, and $v_i \le \bar{c}_i$ since if target i is uncovered, the defender gets utility at most $\bar{c}_i$. Therefore, we have

$$\begin{aligned} OPT(LP\ 2) &\le p_0 U^* + \frac{1-p_0}{n}\sum_{i=1}^{n}(x^*_{ii}U^* + (1-x^*_{ii})\bar{c}_i) \\ &\le (p_0 + k\frac{1-p_0}{n})U^* + \frac{1-p_0}{n}\sum_{i=1}^{n}\bar{c}_i \end{aligned}$$

We now examine $U_{indepSample}$. A simple argument yields that, assuming $\bar{c}_i \ge 0$ for all i and each target i is covered by probability at least αx^*_i for any i, then the defender utility is at least αU^*. Therefore, by Lemma 5 we have

$$\begin{aligned} U_{indepSample} &\ge p_0(1-\frac{1}{e})U^* + \frac{1-p_0}{n}\sum_{i=1}^{n} y_{ii}(\frac{k-2}{k-1} - \frac{1}{e})U^* \\ &\ge (\frac{k-2}{k-1} - \frac{1}{e})(p_0 + k\frac{1-p_0}{n})U^* \end{aligned}$$

Comparing these two inequalities, we have $U_{indepSample} \ge (\frac{k-2}{k-1} - \frac{1}{e})[OPT(LP2) - \frac{1-p_0}{n}\sum_{i=1}^{n}\bar{c}_i]$. This concludes our proof of Theorem 2.

4 Sampling Algorithms

From Carathéodory's theorem we know that, given any marginal coverage $\vec{x}$, there are many different mixed strategies achieving the same marginal $\vec{x}$ (e.g., see examples in Section 2). Another way to handle information leakage is to generate the optimal marginal coverage $\vec{x}^*$, computed by LP (1), with low correlation between targets. Such a "good" mixed strategy, e.g., the mixed strategy with maximum entropy, is usually supported on a pure strategy set of exponential size. In this section, we propose two sampling algorithms, which efficiently generate a mixed strategy with exponentially large support and are guaranteed to achieve any given marginal $\vec{x}$.

4.1 Max-Entropy Sampling

Perhaps the most natural choice to achieve low correlation is the distribution with maximum entropy restricted to achieving the marginal $\vec{x}$, which can be formulated as the solution of Convex Program (CP) (11). However, naive approaches for CP (11) require exponential running time since there are $O(2^n)$ variables. Interestingly, it turns out that this be resolved.

$$\begin{array}{lll} \text{maximize} & \sum_{s\in S} -\theta_s \ln(\theta_s) & \\ \text{subject to} & \sum_{s:\, i\in s}\theta_s = x_i, & \text{for } i \in [n]. \\ & \sum_{s\in S}\theta_s = 1 & \\ & \theta_s \ge 0, & \text{for } s \in S. \end{array} \tag{11}$$

where variable θ_s is the probability of taking pure strategy s.

Theorem 3. *There is an efficient algorithm which runs in poly(n,k) time and outputs a pure strategy s with probability θ^*_s for any pure strategy $s \in S$, where $\vec{\theta}^*$ is the optimal solution to Convex Program* (11) *(within machine precision[1]).*

[1]Computers cannot solve general convex programs exactly due to possible irrational solutions. Therefore, our algorithm is optimal within machine precision, and we simply call it "solved".

The proof of Theorem 3 relies on Lemmas 6 and 7. Lemma 6 presents a compact representation of $\vec{\theta}^*$ based on the KKT conditions of CP (11) and its dual – the *unconstrained* Convex Program (12):

$$\text{minimize} \quad f(\vec{\beta}) = \sum_{i=1}^{n} \beta_i x_i + \ln(\sum_{s \in S} e^{-\beta_s}), \tag{12}$$

where variables $\vec{\beta} \in \mathbb{R}^n$ and $e^{-\beta_s} = \Pi_{i \in s} e^{-\beta_i}$. We notice that the dual program (12) as well as the characterization of θ_s^* in Lemma 6 are not new (e.g., see [15]), and we state it for completeness. Our contribution lies at proving that CP (12) can be computed efficiently in $poly(n, k)$ time in our security game setting despite the summation $\sum_{s \in S} e^{-\beta_s}$ of $O(2^k)$ terms.

Lemma 6. *Let $\vec{\beta}^* \in \mathbb{R}^n$ be the optimal solution to CP (12) and set $\alpha_i = e^{-\beta_i^*}$ for any $i \in [n]$, then the optimal solution of CP (11) satisfies*

$$\theta_s^* = \frac{\alpha_s}{\sum_{s \in S} \alpha_s}, \tag{13}$$

where $\alpha_s = \Pi_{i \in s} \alpha_i$ for any pure strategy $s \in S$.

Furthermore, $\vec{\beta}^*$ can be computed in $poly(n, k)$ time.

Proof. As proved in [15], the α^i above is precisely $e^{-\beta_i^*}$ where $\vec{\beta}^*$ is the optimal solution to CP (11). We show that $\vec{\beta}^*$ can be computed in $poly(n, k)$ time. Notice that CP (12) has n variables but an exponential-size expression. The essential difficulty of computing $f(\vec{\beta})$ (or $\nabla f(\vec{\beta})$) is the sum $\sum_{s \in S} e^{-\beta_s}$, which fortunately exhibits combinatorial structures. Therefore, combinatorial algorithms could be employed for computing this sum. In particular, we show that a dynamic program computes the sum $\sum_{s \in S} e^{-\beta_s}$, therefore the value of $f(\vec{\beta})$ as well, in $poly(n, k)$ time. The algorithm for $\nabla f(\vec{\beta})$ can be designed in a similar fashion, and hence left to the reader. Since a convex program can be solved efficiently in machine precision given the access to its function value and derivatives, we then conclude our proof by describing the following dynamic program to compute $\sum_{s \in S} e^{-\beta_s}$, given any $\vec{\beta}$.

Notice that the set of all pure strategies consists of all the subsets of $[n]$ of cardinality k. Let $\alpha_i = e^{-\beta_i}$ and $\alpha_s = \Pi_{i \in s} \alpha_i$. We then build the following DP table $T(i, j) = \sum_{s : s \subseteq [j], |s| = i} \alpha_s$, which sums over all the subsets of $[j]$ of cardinality i. Our goal is to compute $T(k, n) = \sum_{s \in S} e^{-\beta_s}$. We first initialize $T(1, j) = \sum_{i=1}^{j} \alpha_i$ and $T(j, j) = \Pi_{i=1}^{j} \alpha_i$ for any j. Then using the following update rule, we can build the DP table and compute $T(k, n)$ in $poly(k, n)$ time.

$$T(i, j) \quad = \quad T(i, j-1) + \alpha_j T(i-1, j-1).$$

□

Our next lemma considers how to efficiently sample a pure strategy s from an exponentially large support with probability θ_s^* represented by Equation (13). The algorithm, as detailed in Algorithm 2, simply goes through each target and includes the target with a specifically designed probability until exactly k targets are sampled.

Lemma 7. *Given any input $\vec{\alpha} \in [0, \infty)^n$, Algorithm 2 runs in $poly(k, n)$ time and correctly samples a pure strategy s with probability $\theta_s = \frac{\alpha_s}{\sum_{s \in S} \alpha_s}$, where $\alpha_s = \Pi_{i \in s} \alpha_i$.*

Proof. It is easy to see that Table $T(i, j)$ can be computed in $poly(n, k)$. We first show that the "while" loop in Algorithm 2 terminates within at most n steps. In fact, j decreases by 1 each step and furthermore $j \geq i \geq 0$ always holds. This is because when j decreases until $j = i$, j will be

Algorithm 2 Max-Entropy Sampling

Input: : $\vec{\alpha} \in [0, \infty)^n$, k.

Output: : a pure strategy s with $|s| = k$.

1: Initialize: $s = \emptyset$; the DP table $T(1, j) = \sum_{i=1}^{j} \alpha_i$ and $T(j, j) = \Pi_{i=1}^{j} \alpha_i$ for any $j \in [n]$.

2: Compute $T(i, j) = \sum_{s:s\subseteq[j],|s|=i} \alpha_s$ for any i, j satisfying $i \leq k, j \leq n$ and $i \leq j$, using the following update rule

$$T(i, j) \quad = \quad T(i, j-1) + \alpha_j T(i-1, j-1).$$

3: Set $i = k$, $j = n$;

4: **while** $i > 0$ **do**

5: Independently add j to s with probability

$$p = \frac{\alpha_j T(i-1, j-1)}{T(i, j)};$$

6: **if** j added to s **then**

7: $i = i - 1$;

8: **end if**

9: $j = j - 1$;

10: **end while**

11: **return** s.

sampled with probability $\frac{\alpha_j T(i-1,j-1)}{T(i,j)} = \frac{\alpha_i T(i-1,i-1)}{T(i,i)} = 1$, and at the next iteration, $j-1$ will also be sampled with probability 1 and so on so forth until $i = 0$. Furthermore, the algorithm terminates with $|s| = k$ because $|s| = k - i$ always holds until the termination at $i = 0$. Therefore, Algorithm 2 runs in $poly(n, k)$ time.

Now we show that Algorithm 2 outputs s with probability θ_s. Let the output $s = \{i_1, ..., i_k\}$ be sorted in decreasing order, i.e., $i_1 > i_2 > ... > i_k$. Notice that

$$T(i, j) = \alpha_j T(i-1, j-1) + T(i, j-1).$$

Therefore, in the *Sampling* step in Algorithm 2, j is not included to s with probability $T(i, j-1)/T(i, j)$. Therefore, to sample $s = \{i_1, ..., i_k\}$, it must be the case that $n, n-1, ..., i_1 + 1$ are not included, while i_1 is included; $i_1 - 1, ..., i_2 + 1$ are not included, while i_2 is included; and so on so forth. In addition, each of these steps are sampled independently and the probability of each step is known. Therefore, by multiplying these probabilities together, we have

$$\begin{aligned} P(s) \quad &= \quad \frac{T(k, n-1)}{T(k, n)} \times \frac{T(k, n-2)}{T(k, n-1)} ... \times \frac{\alpha_{i_1} T(k-1, i_1 - 1)}{T(k, i_1)} \\ &\quad \times \frac{T(k-1, i_1 - 2)}{T(k-1, i_1 - 1)} ... \frac{\alpha_{i_n} T(0, i_n - 1)}{T(1, i_n)} \\ &= \quad \frac{\Pi_{t \leq k} \alpha_{i_t}}{T(k, n)} \\ &= \quad \theta_s \end{aligned}$$

where we set $T(0, j) = 1$ for any j. This precisely gives the probability we want. □

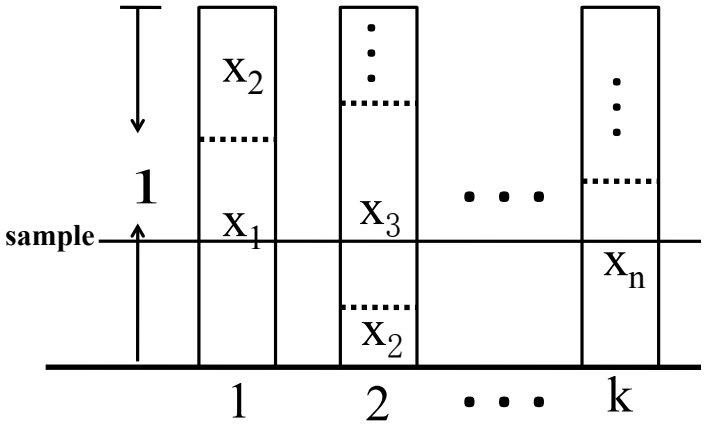

Figure 1: Comb Sampling

Remark: we notice that *approximately uniform* sampling from combinatorial structures has been studied in theoretical computer science [8]. Algorithm 2 uses a variant of the algorithm in [8], and extends their results to the *weighted* (by θ_s^*) and *exact* case.

4.2 Uniform Comb Sampling

[17] presented the Comb Sampling algorithm, which randomly samples a pure strategy and achieves a given marginal in expectation. The algorithm can be elegantly described as follows (also see Figure 1): thinking of k resources as k buckets with height 1 each, we then put each target, the height of which equals precisely its marginal probability, one by one into the buckets. If one bucket gets full when filling in a certain target, we move the "rest" of that target to a new empty bucket. Continue this until all the targets are filled in, at which time we know that k buckets are also full. The algorithm then takes a horizontal line with a uniformly randomly chosen height from the interval $[0, 1]$, and the k targets intersecting the horizontal line constitute the sampled pure strategy. As easily observed, Comb Sampling achieves the marginal coverage in expectation [17].

However, is Comb Sampling robust against information leakage? We first observe that Comb Sampling generates a mixed strategy with support size at most $n + 1$, which precisely matches the upper bound of Carathéodory's theorem.

Proposition 1. *Comb Sampling generates a mixed strategy which mixes over at most $n + 1$ pure strategies.*

Proposition 1 suggests that the mixed strategy sampled by Comb Sampling might be very easy to explore. Therefore we propose a variant of the Comb Sampling algorithm. Our key observation is that Comb Sampling achieves the marginal coverage regardless of the order of the targets. That is, the marginal is still obtained if we randomly shuffle the order of the targets *each time* before sampling, and then fill in them one by one. Therefore, we propose the following Uniform Comb Sampling (UniCS) algorithm:

1. Order the n targets uniformly at random;

2. fill the targets into the buckets based on the random order, and then apply Comb Sampling.

Since the order is chosen randomly each time, the mixed strategy implemented by UniCS mixes over exponentially many pure strategies, and achieves the marginal.

Proposition 2. *Uniform Comb Sampling (UniCS) achieves the marginal coverage probability.*

5 Experiments

Traditional algorithms for computing Strong Stackelberg Equilibrium (SSE) only optimize the coverage probability at each target, without considering their correlations. In this section, we experimentally study how traditional algorithms and our new algorithms perform in presence of *probabilistic* or *adversarial* information leakage. In particular, we compare the following five algorithms.

- *Traditional*: optimal marginal + comb sampling, the traditional way to solve security games with no scheduling constraints [9, 17];
- *OPT*: the optimal algorithm for PRIL or ADIL model (Section 3.1) using column generation with the defender oracle in Algorithm 1;
- *indepSample*: independent sampling without replacement (Section 3.2);
- *MaxEntro*: max entropy sampling (Algorithm 2);
- *UniCS*: uniform comb sampling (Section 4.2).

All algorithms are tested on the following two sets of data:

Los Angeles International Airport (LAX) Checkpoint Data from [13]. This problem was modeled as a Bayesian Stackelberg game with multiple adversary types in [13]. To be consistent with our model, we instead only consider the game against one particular type of adversary – the terrorist-type adversary, which is the main concern of the airport. The defender's rewards and costs are obtained from [13] and the game is assumed to be zero-sum in our experiments.

Simulated Game Payoffs. A systematic examination is conducted with simulated payoffs. All generated games have 20 targets and 10 resources. The reward r_i (cost c_i) of each target i is chosen uniformly at random from the interval $[0, 10]$ ($[-10, 0]$).

In terms of running time, all the algorithms run efficiently as expected (terminate within seconds using MATLAB) except the optimal algorithm *OPT*, which takes about 3 minutes per simulated game on average. Therefore we mainly compare defender utilities. All the comparisons are listed in Figure 2 (for LAX data) and Figure 3 (for simulated data). The line "Basis" is the utility with no leakage and is listed as a basis for utility comparisons. Y-axis is the defender's utility – the higher, the better. We examine the effect of the *total probability of leakage* (i.e., the x-axis $1 - p_0$) on the defender's utility and consider $1 - p_0 = 0, 0.1, ..., 1$. For probabilistic information leakage, we randomly generate the probabilities that each target leaks information with the constraint $\sum_{i=1}^{n} p_i = 1 - p_0$. For the case of leakage from small support (for simulated payoffs only), we randomly choose a support of size 5. All the utilities are *averaged* over 50 random games except the ADIL model for LAX data. For the simulated payoffs, we also consider a special case of uniform leakage probability of each target (see Theorem 2). The following observations follow from the figures.

Observation 1. The gap between the line "*Basis*" and "*OPT*" shows that information leakage from even one target does cause dramatic utility decrease to the defender. Moreover, adversarial

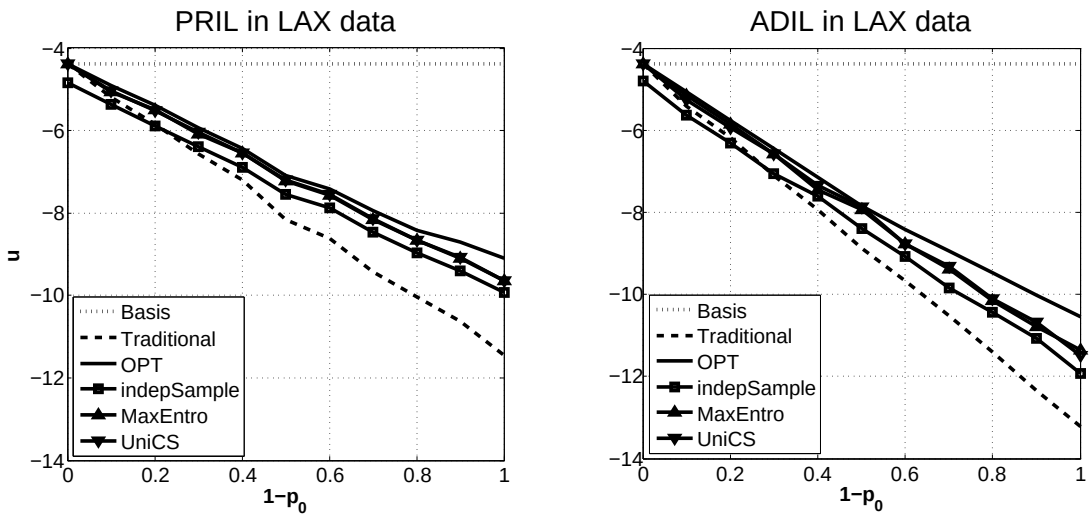

Figure 2: Comparisons on real LAX airport data.

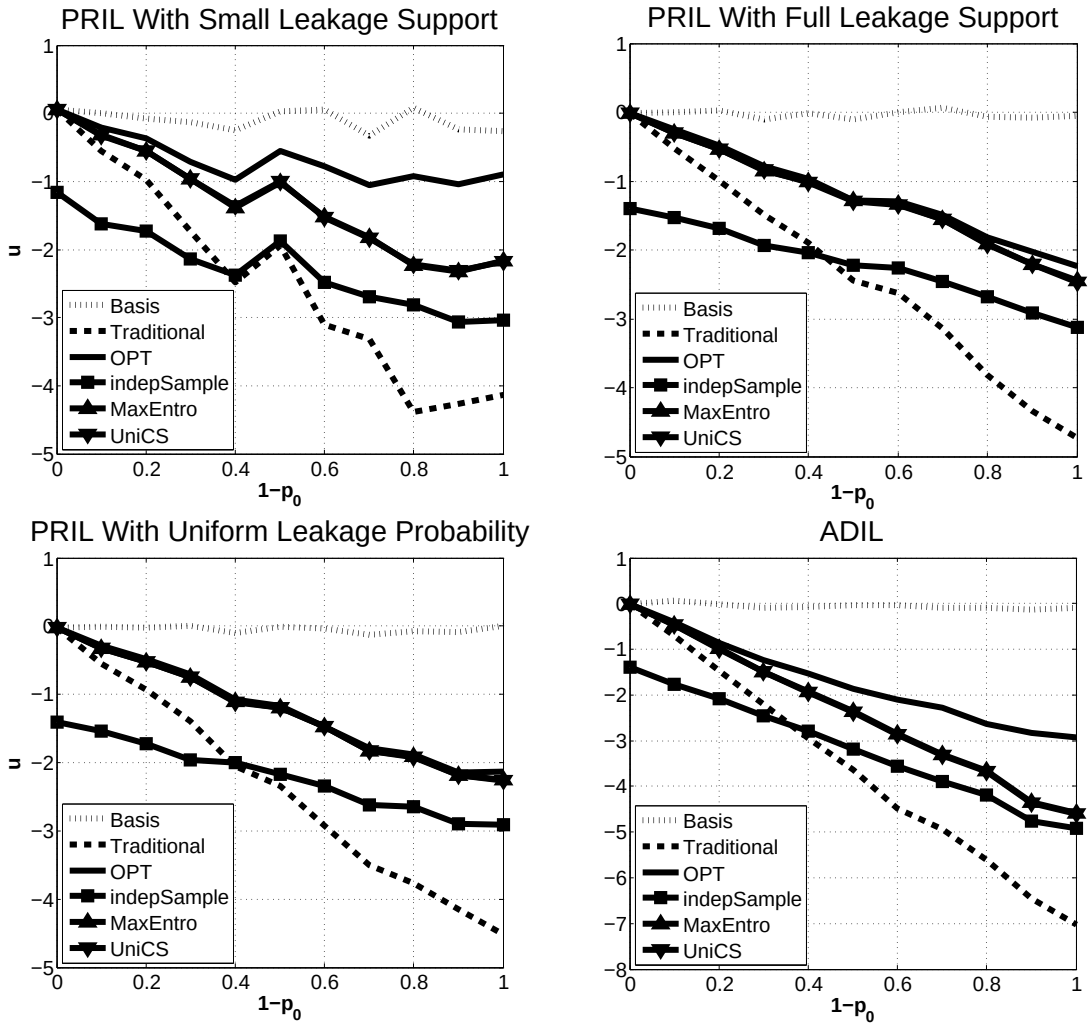

Figure 3: Comparisons in Simulated Games.

leakage causes more utility loss than probabilistic leakage; leakage from a restricted small support of targets causes less utility decrease than from full support.

Observation 2. The gap between the line "*OPT*" and "*Traditional*" demonstrates the necessity of handling information leakage. In particular, the relative loss $u(OPT)-u(Basis)$ is approximately half of the relative loss $u(Traditional)-u(Basis)$ in Figure 3 (and 65% in Figure 2). Furthermore, if leakage is from a small support (left-up panel in Figure 3), *OPT* is close to *Basis*.

Observation 3. *MaxEntro* and *UniCS* have almost the same performance (overlapping in all these figures). Both algorithms are almost optimal when the leakage support is the full set $[n]$ (they almost overlap with *OPT* in the right-up and left-down panels in Figure 3).

Observation 4. An interesting observation is that *IndepSample* outperforms *Traditional* at $1-p_0 = 0.3$ or 0.4 in all of these figures, which is around $\frac{1}{e} \approx 0.37$. Furthermore, the gap between *IndepSample* and *OPT* does not change much at different $1-p_0$.

Observation 5. From a practical view, if the leakage is from a small support, *OPT* is preferred as it admits efficient algorithms (Section 3.1); if the leakage is from a large support, *MaxEntropy* and *UniCS* are preferred as they can be computed efficiently and are close to optimality. From a theoretical perspective, we note that the intriguing performance of *IndepSample*, *MaxEntropy* and *UniCS* raises questions for future work.

6 Conclusions and Discussions

In this paper, we considered partial information leakage in Stackelberg security games. We focused on the one-target leakage case, but do emphasize that our models, hardness results and algorithms can be easily generalized. Our results raise several new research questions, e.g., is it possible to derive a theoretical approximation guarantee for *MaxEntro* and *UniCS*, and can we develop efficient algorithms to handle information leakage in other security game settings? More generally, it is an interesting problem to study analogous issues of information leakage in other settings beyond security, e.g., auctions or general games.

References

[1] Noa Agmon, Sarit Kraus, and Gal A. Kaminka. Multi-robot perimeter patrol in adversarial settings. 2008.

[2] Noa Agmon, Vladimir Sadov, Gal A. Kaminka, and Sarit Kraus. The impact of adversarial knowledge on adversarial planning in perimeter patrol. volume 1, pages 55–62, 2008.

[3] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, 2009.

[4] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artif. Intell.*, 184-185:78–123, June 2012.

[5] Nicola Basilico, Nicola Gatti, Thomas Rossi, Sofia Ceppi, and Francesco Amigoni. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '09. IEEE Computer Society, 2009.

[6] Branislav Bošanský, Viliam Lisý, Michal Jakob, and Michal Pěchouček. Computing time-dependent policies for patrolling games with mobile targets. In *AAMAS*. IFAAMAS, 2011.

[7] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordonez, and Milind Tambe. Security games with arbitrary schedules: A branch and price approach. In *AAAI*. AAAI Press, 2010.

[8] Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[9] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordonez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[10] D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *AAAI*, 2010.

[11] J. Letchford and Y. Vorobeychik. Computing randomized security strategies in networked domains. In *AARM*, 2011.

[12] Joshua Letchford and Yevgeniy Vorobeychik. Computing randomized security strategies in networked domains. In *Applied Adversarial Reasoning and Risk Modeling*, volume WS-11-06 of *AAAI Workshops*. AAAI, 2011.

[13] James Pita, Manish Jain, Janusz Marecki, Fernando Ordonez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection: The application of a game theoretic model for security at the los angeles international airport. In *AAMAS*, 2008.

[14] Itamar Pitowsky. Correlation polytopes: Their geometry and complexity. *Math. Program.*, pages 395–414, 1991.

[15] Mohit Singh and Nisheeth K. Vishnoi. Entropy, optimization and counting. *CoRR*, 2013.

[16] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, New York, NY, USA, 2011.

[17] Jason Tsai, Zhengyu Yin, Jun young Kwak, David Kempe, Christopher Kiekintveld, and Milind Tambe. Urban security: Game-theoretic resource allocation in networked physical domains. In *National Conference on Artificial Intelligence (AAAI)*, 2010.

[18] Y. Vorobeychik, B. An, M. Tambe, and S. Singh. Computing solutions in infinite-horizon discounted adversarial patrolling game. In *International Conference on Automated Planning and Scheduling (ICAPS), June 2014*, 2014.

7 Appendix A: Derivation of Defender Oracle

A defender oracle is a subroutine used to solve security games with large number of pure strategies by the *Column Generation* technique. In this section, we first describe the technique of column generation and then derive the formulation for the defender oracle.

Recall that LP (2) has a large number of variables because the number of pure strategies is exponential. However, in the optimal mixed strategy, only polynomially many of these pure strategies will have strictly positive probabilities while all the other pure strategies have probability 0. Column generation is based on this observation. Basically, instead of solving LP (2) on the set S of all pure strategies, it starts from a small subset of pure strategies, denoted as A, and solve the following "restricted" LP.

$$\begin{array}{lll}
\text{maximize} & p_0 u + \sum_{i=1}^{n} p_i(u_i + v_i) & \\
\text{subject to} & u \le r_j x_{jj} + c_j(1 - x_{jj}), & \text{for } j \in [n]. \\
& u_i \le r_j x_{ij} + c_j(x_{ii} - x_{ij}), & \text{for } i, j \in [n]. \\
& v_i \le r_j(x_{jj} - x_{ij}) + c_j(1 - x_{ii} - x_{jj} + x_{ij}), & \text{for } i, j \in [n]. \\
& x_{ij} = \sum_{s: i,j \in s} \theta_s, & \text{for } i, j \in [n]. \\
& \sum_{s \in A} \theta_s = 1 & \\
& \theta_s \ge 0, & \text{for } s \in A.
\end{array} \tag{14}$$

Notice that the only difference between LP (2) and LP (14) is that the set S of all pure strategies is substituted by a small subset A. In practice, A is usually initialized with constantly many pure strategies that are arbitrarily chosen. Column generation proceeds as follows: 1. it solves LP (14); 2. by checking the dual of LP (2) the defender oracle asserts whether the computed solution is also optimal to LP (2); if not, the oracle finds a new pure strategy to be added to the set A and updates A. This procedure continues until the defender oracle asserts that the computed optimal solution w.r.t. current A is also optimal to LP (2).

We first derive the dual of LP (14). In fact, to emphasize the key aspects and avoid messy derivations, we abstractly re-write LP (14) in the following form:

$$\begin{array}{lll}
\text{maximize} & d^T \alpha & \\
\text{subject to} & Mx + N\alpha \le c & \\
& x_{ij} - \sum_{s: i,j \in s} \theta_s = 0, & \text{for } i, j \in [n]. \\
& \sum_{s \in A} \theta_s = 1 & \\
& \theta_s \ge 0, & \text{for } s \in A.
\end{array} \tag{15}$$

where variable α represents the vector consisting of u, v_i, u_i while variable x is the vector representation of x_{ij} (putting i, j in a certain order); d is a vector summarizing the original objective; the constraints $Mx + N\alpha \le c$ summarizes the first three constraints in LP (14) where importantly M, N are of polynomial size. Let $M_{index(i,j)}$ be the column of M corresponding to x_{ij} and N_k be the column of N corresponding to the k'th component of α. We can now simply derive the dual of LP (15) as follows:

$$\begin{array}{lll}
\text{minimize} & c^T \rho + \omega & \\
\text{subject to} & \rho^T N_k \ge d_k, & \text{for all } k. \\
& \rho^T M_{index(ij)} + \beta_{ij \ge 0}, & \text{for } i, j \in [n]. \\
& -\sum_{i,j \in s} \beta_{ij} + \omega \ge 0, & \text{for } s \in A. \\
& \rho \ge 0 &
\end{array} \tag{16}$$

where ρ are the dual variables w.r.t. first constraints in LP (15) and β_{ij}, ω are the dual variables w.r.t. the second and third constraints.

First notice that both the optimal solution to LP (15) (denoted as $OptSol_A$) and the optimal solution to LP (16) (denoted as $OptSolDual_A$) can both be computed efficiently when A is small. A key observation here is that, if $OptSolDual_A$, in particular, the ω, β_{ij} inside, happens to make the constraints $-\sum_{i,j \in s} \beta_{ij} + \omega \ge 0$, $\forall s \in A$ hold more generally as $-\sum_{i,j \in s} \beta_{ij} + \omega \ge 0$, $\forall s \in S$, then we claim that the $OptSol_A$ is also an optimal solution to LP (2) (by picking pure strategies in $S \setminus A$

with probability 0). This is because, if we substitute A by S in both LP (15) and LP (16), $OptSol_A$ is still feasible to LP (15) because all the strategies in $S \setminus A$ has probability 0; $OptSolDual_A$ is still feasible to LP (16) because our ω, β_{ij} make constraints $-\sum_{i,j\in s}\beta_{ij}+\omega \geq 0$ hold for all $s \in S$ by assumption. Furthermore, *complementary slackness* still holds since the added new variables in LP (15) all take value 0. By linear program basics, we know that $OptSol_A$ is still optimal if we substitute A in LP (15) by S, which is precisely LP (2).

As a result, our key task is to judge whether $-\sum_{i,j\in s}\beta_{ij}+\omega \geq 0$ holds for all $s \in S$ for a given dual solution. This is equivalent to decide whether $\omega \geq \max_{s\in S}\left[\sum_{i,j\in s}\beta_{ij}\right]$. The defender oracle is precisely the following problem:

$$\max_{s\in S}\left[\sum_{i,j\in s}\beta_{ij}\right] = \max_{s\in S} s^T(\frac{M+M^T}{2})s \tag{17}$$

where M is the matrix satisfying $M_{ij}=\beta_{ij}$.

With this oracle, column generation proceeds in more details as follows: 1. compute LP (15) and LP (16); 2. use the defender oracle to solve Problem (17), if the optimal value is less than or equal to the dual variable ω, asserts optimality; otherwise, add the s^* – the optimal solution to Problem (17) – to A; 3. repeat until optimality is reached. Notice that the newly added s^* does not belong to the original A because all $s \in A$ satisfies $\sum_{i,j\in s}\beta_{ij} \leq \omega$. Column generation does not guarantee polynomial convergence, but usually converges very fast in practice.

Notice that, one reason that we use the abstract representation in LP (15) is also to convey the message that the incentive constraints and objective are not essential in column generation as long as there are polynomially many. In other words, if other objectives or constraints are used, the same technique also applies. When information leaks from a small subset of targets (Section 3.1), the variables x_{ij} become less since only correlation between leaking targets and other targets are considered. By modifying LP (15) and LP (16) a bit, we can get the defender oracle form as in Section 3.1.

8 Appendix B: Relation Between $\mathcal{P}(n,k)$ and Correlation Polytope

In this section, we show a connection between $\mathcal{P}(n,k)$ and the correlation polytope, defined as follows:

Definition 1. *[14] Given an integer n, the* Correlation Polytope $\mathcal{P}(n)$ *is defined as follows*

$$\mathcal{P}(n) = Conv\left(\{vv^T : v \in \{0,1\}^n\}\right).$$

where $Conv(S)$ denotes the convex hull of set S. Notice that $vv^T \in \mathbb{R}^{n\times n}$.

The following lemma captures the relation between $\mathcal{P}(n)$ and $\mathcal{P}(n,k)$.

Lemma 8. *$X \in \mathcal{P}(n,k)$ if and only if the following three constraints hold: 1. $X \in \mathcal{P}(n)$; 2. $tr(X)=\sum_{i=1}^n x_{ii}=k$; 3. $sum(X)=\sum_{i,j=1}^n x_{ij}=k^2$.*

Proof. We show that, given $X \in \mathcal{P}(n)$, if X satisfies the following two linear constraints: $tr(X)=k$, $sum(X)=k^2$, then $X \in \mathcal{P}(n,k)$.

Since $X \in \mathcal{P}(n)$, there exits $X_i \in \mathcal{P}(n,i)$ and $p_i \geq 0$, such that $X=\sum_{i=1}^n p_i X_i$ and $\sum_{i=1}^n p_i = 1$. That is, X is a convex combination of elements from each $\mathcal{P}(n,i)$. Notice that $\forall X_i \in \mathcal{P}(n,i)$, we

have $tr(X_i) = i$ and $sum(X_i) = i^2$, because any vertex of $\mathcal{P}(n, i)$ satisfies these constraints. Let $X \in \mathcal{P}(n, k)$, then we have:

$$(i) : 1 = \sum_{i=1}^{n} p_i$$

$$(ii) : k = tr(X) = \sum_{i=1}^{n} p_i \times tr(X_i) = \sum_{i=1}^{n} p_i \times i$$

$$(iii) : k^2 = sum(X) = \sum_{i=1}^{n} p_i \times sum(X_i) = \sum_{i=1}^{n} p_i \times i^2$$

By Cauchy–Schwarz inequality, we have $(\sum_{i=1}^{n} p_i)(\sum_{i=1}^{n} p_i i^2) \geq (\sum_{i=1}^{n} p_i i)^2$. Therefore, equality holds according to the above three equations. The condition of equality for Cauchy–Schwarz inequality is that $p_i i^2 / p_i$ is a constant for all i, such that $p_i \neq 0$. This only yields that there is only one non-zero p_i, i.e., $p_k = 1$. Thereofore, $X \in \mathcal{P}(n, k)$. □

Remark: we notice that [14] defines correlation polytope in a more general fashion and our definition of $\mathcal{P}(n)$ is an important special case of the definition of correlation polytope in [14], which is called full correlation polytope. Nevertheless, this definition is sufficient for our model. [14] proved that membership check for polytope $\mathcal{P}(n)$ is NP-complete. Lemma 8 basically conveys that optimizing over polytope $\mathcal{P}(n, k)$ is "easier" than optimizing over $\mathcal{P}(n)$. Nevertheless, Lemma 4 shows that optimizing over $\mathcal{P}(n, k)$ is still NP-hard.