# Security Games with Multiple Attacker Resources

**Dmytro Korzhyk, Vincent Conitzer, Ronald Parr**
Department of Computer Science, Duke University
Durham, NC 27708, USA
{dima, conitzer, parr}@cs.duke.edu

## Abstract

Algorithms for finding game-theoretic solutions are now used in several real-world security applications. This work has generally assumed a Stackelberg model where the defender commits to a mixed strategy first. In general two-player normal-form games, Stackelberg strategies are easier to compute than Nash equilibria, though it has recently been shown that in many security games, Stackelberg strategies are also Nash strategies for the defender. However, the work on security games so far assumes that the attacker attacks only a single target. In this paper, we generalize to the case where the attacker attacks multiple targets simultaneously. Here, Stackelberg and Nash strategies for the defender can be truly different. We provide a polynomial-time algorithm for finding a Nash equilibrium. The algorithm gradually increases the number of defender resources and maintains an equilibrium throughout this process. Moreover, we prove that Nash equilibria in security games with multiple attackers satisfy the interchange property, which resolves the problem of equilibrium selection in such games. On the other hand, we show that Stackelberg strategies are actually NP-hard to compute in this context. Finally, we provide experimental results.

## 1 Introduction

Algorithms for computing game-theoretic solutions have recently found several real-world applications to security, including the randomized allocation of checkpoints and canine units at Los Angeles International Airport [5; 6] and of federal air marshals to flights [7]. These *security games* are generally two-player games between a defender, who allocates defensive resources to targets (or subsets of targets), and an attacker, who chooses which target to attack. The attacked target determines both players' utilities; holding the attacked target fixed, if the target is defended by one of the defender's resources, this increases the defender's utility and decreases the attacker's utility. In spite of their apparently adversarial nature, security games are generally *not* modeled as zero-sum games.

This leads to a question of which game-theoretic solution concept should be used. One possibility is to solve for a Nash equilibrium of the game. Instead, however, most of the work on these security games assumes that the defender has a *Stackelberg leader advantage*, where she can commit to a mixed strategy before the attacker moves. One can support the Stackelberg model by arguing that an attacker would spend some time observing the defender's day-to-day actions and thereby learn the defender's mixed strategy before engaging in an attack. The validity of this argument can be debated. Nevertheless, there are advantages to the Stackelberg model, including that being a Stackelberg leader cannot hurt the defender [8], as well as that at least in some cases, such as general two-player normal-form games, computing a Nash equilibrium is PPAD-complete [1] whereas a Stackelberg strategy for the defender can be computed in polynomial time [2].

Security games are a restricted class of games. This additional structure can remove the distinction between Stackelberg and Nash strategies to some extent: if we make certain assumptions about the security game, it turns out that any defender Stackelberg strategy is also a defender Nash strategy [9], though this does not hold in the most general case. The additional structure in security games can also allow for more efficient algorithms. On the other hand, the size of the normal-form representation of the game is generally exponential in the natural parameters of the problem, so that it becomes impractical to apply algorithms based on the normal form.

Kiekintveld et al. [3] define a general class of security games based on allocating defensive resources to targets or subsets of targets. This model was further extended by Yin et al. [9] to allow multiple attacker resources, that is, the attacker can simultaneously attack up to $n_a$ different targets. This extension is motivated by the fact that terrorist attacks are often performed at multiple locations simultaneously (for example, the 9/11 attacks or the 2008 Mumbai attacks).

In the case of a single attacker resource, Kiekintveld et al. give a simple algorithm called ORIGAMI for the single attacker resource case. It computes a defender strategy that is both a Stackelberg and a Nash strategy (the latter follows from the work of Yin et al. [9]). The main observation used in ORIGAMI is that any Stackelberg strategy for the defender minimizes the attacker's best-response utility. Using this observation, ORIGAMI computes the defender's Nash/Stackelberg strategy by gradually decreasing the attacker's best-response utility, keeping track of the number of defender resources required to bring the attacker's best-response utility down to this level, until the number of required defender resources reaches the limit. However, in games with multiple attacker resources, the defender's minimax strategy is not necessarily a Nash or Stackelberg strategy. Consider the following example (from an extended version of Yin et al. [9]). Suppose there are two targets, and the attacker has two resources, so that both targets will be attacked no matter what strategy the defender chooses. If the defender has only one resource, then the defender is better off allocating that resource in such a way that the defender's utility increases the most. However, in the defender's minimax strategy, the defender would allocate the resource so that the attacker's utility is reduced the most. Thus the defender's Nash and Stackelberg strategies can differ from the minimax strategy in this example. In the appendix, we present an example from the extended version of Yin et al. [9] in which the defender's Nash equilibrium strategy differs from the Stackelberg strategy.

In this paper, we give a polynomial-time algorithm for finding a Nash equilibrium in the case of multiple attacker resources. This algorithm can be thought of as a generalization of ORIGAMI in the sense that it also keeps track of the smallest utility that the attacker is going to get from any of his targets, and this utility gradually decreases over the course of the algorithm. However, our algorithm is far more complicated compared to ORIGAMI. Furthermore, we show that Nash equilibria in security games with multiple attacker resources possess the interchange property, which states that as long as each player plays *some* equilibrium strategy, the resulting strategy profile must be a Nash equilibrium, thus resolving the problem of equilibrium selection for both players. On the other hand, we show that, with multiple attacker resources, computing a defender Stackelberg strategy is actually NP-hard.

## 2 Security Games, Strategies, and Notation

In the security games that we study, there is a set of *targets*, $T$. The defender has $n_d$ resources and the attacker has $n_a$ resources. A pure strategy for the defender (attacker) consists of a subset $S_d \subseteq T$ with $|S_d| = n_d$ ($S_a \subseteq T$ with $|S_a| = n_a$), corresponding to the targets she defends (he attacks). Targets that are not attacked do not affect either player's utility. Each player's utility is additive over attacked targets. For a target $t$, the defender receives utility $u_d^u(t)$ if the target is attacked but not defended (uncovered), and $u_d^c(t)$ if the target is attacked and defended (covered). Similarly, the attacker's utility for an attacked target $t$ is $u_a^u(t)$ in the uncovered case and $u_a^c(t)$ in the covered case. (We require $\Delta u_d(t) = u_d^c(t) - u_d^u(t) > 0$ and $\Delta u_a(t) = u_a^u(t) - u_a^c(t) > 0$.) Hence, the defender's overall utility is $\sum_{t \in S_a \cap S_d} u_d^c(t) + \sum_{t \in S_a \setminus S_d} u_d^u(t)$, and the attacker's overall utility is $\sum_{t \in S_a \cap S_d} u_a^c(t) + \sum_{t \in S_a \setminus S_d} u_a^u(t)$.

Because of the additive nature of the utility function, the players' expected utilities depend only on the marginal probability that each target is attacked/defended. Hence, a defender mixed strategy can be represented as a vector $\mathbf{d} = (d_1, \ldots, d_{|T|})$ with $\sum_{t \in T} d_t = n_d$, where $d_t \in [0,1]$ is the probability that target $t$ obtains a defensive resource, and similarly for the attacker $\mathbf{a} = (a_1, \ldots, a_{|T|})$ with $\sum_{t \in T} a_t = n_a, a_t \in [0,1]$. (Note that it does not help to have more than one resource on one target. This assumption was introduced in the security game model by Kiekintveld et al. [3].)

We will use the following shorthand: $u_a(t, d_t) = d_t u_a^c(t) + (1 - d_t) u_a^u(t)$ is the attacker's expected utility for attacking target $t$, and $v_d(t, a_t) = a_t \Delta u_d(t)$ is the marginal expected utility that the defender gets from defending $t$.

## 3 Nash Equilibrium

We now consider how to compute a Nash equilibrium, that is, a pair of mixed strategies, one for each player, such that each is a best response to the other. Because of the additive nature of the utility functions, best-responding simply means defending (attacking) the $n_d$ ($n_a$) targets with the highest utility for the defender (attacker). If there is a tie for the $n_d$th ($n_a$th) place, then it is possible to randomize over the corresponding targets. Therefore, $\mathbf{d}$ is a best response to $\mathbf{a}$ iff there exists some threshold marginal utility $\theta_d$ such that $v_d(t, a_t) < \theta_d \Rightarrow d_t = 0$ and $v_d(t, a_t) > \theta_d \Rightarrow d_t = 1$. Similarly, $\mathbf{a}$ is a best response to $\mathbf{d}$ iff there exists some threshold utility $\theta_a$ such that $u_a(t, d_t) < \theta_a \Rightarrow a_t = 0$ and $u_a(t, d_t) > \theta_a \Rightarrow a_t = 1$. Hence, we have a Nash equilibrium iff both these conditions are satisfied. Note that the defender's (and, similarly, the attacker's) strategy can be mixed (randomized), because any target $t$ such that $v_d(t, a_t) = \theta_d$ can be defended with a fractional probability $d_t \in [0,1]$. Similarly, the attacker's strategy can also be mixed.

The high-level idea of our algorithm for computing a Nash equilibrium is as follows. We start with a modified game where the defender has no resources at all, for which it is straightforward to compute an equilibrium, and then we gradually increase the number of defender resources (not necessarily to integral amounts), while maintaining an equilibrium of the game as it is changing—until we arrive at the actual number of defender resources. The algorithm transitions among phases that correspond to phases of qualitatively different behavior in the process of increasing the number of defender's resources. The change resulting from a single phase can be computed through a simple calculation.

### 3.1 Detailed Example Run of the Algorithm

Before we present the algorithm for computing a Nash equilibrium, we first give a detailed example of how it solves a particular game. This example demonstrates the different phases of the algorithm. During these phases, each target will be considered to be in one of 6 possible states: Untouched (U), Newly Attacked (NA), Pending (P), Active (A), Defender Saturated (DS), and Done (D). We will informally introduce every state in this example (precise definitions can be found in the algorithm in Figure 2). The target states depend on the values of the thresholds $\theta_a, \theta_d$, which are computed at the beginning and gradually decrease during the course of the algorithm.

|  | $t_1$[P] | $t_2$[P] | $t_3$[NA] | $t_4$[U] |
|---|---|---|---|---|
| $d_{ti}$ | 0 | 0 | 0 | 0 |
| $a_{ti}$ | 1 | 1 | 0 | 0 |
| $u_a(t_i,\mathbf{c})$ | 5 | 4 | 3 | 2 |
| $\upsilon_d(t_i,\mathbf{a})$ | 1 | 2 | 0 | 0 |

(a) Initialize. $\theta_d$=2, $\theta_a$=3

|  | $t_1$[P] | $t_2$[A] | $t_3$[NA] | $t_4$[U] |
|---|---|---|---|---|
| $d_{ti}$ | 0 | **1/3** | 0 | 0 |
| $a_{ti}$ | 1 | 1 | 0 | 0 |
| $u_a(t_i,\mathbf{c})$ | 5 | **3** | 3 | 2 |
| $\upsilon_d(t_i,\mathbf{a})$ | 1 | 2 | 0 | 0 |

(b) IMOP. $\theta_d$=2, $\theta_a$=3

|  | $t_1$[P] | $t_2$[A] | $t_3$[A] | $t_4$[U] |
|---|---|---|---|---|
| $d_{ti}$ | 0 | 1/3 | 0 | 0 |
| $a_{ti}$ | 1 | **3/5** | **2/5** | 0 |
| $u_a(t_i,\mathbf{c})$ | 5 | 3 | 3 | 2 |
| $\upsilon_d(t_i,\mathbf{a})$ | 1 | **6/5** | **6/5** | 0 |

(c) MMNA. $\theta_d$=6/5, $\theta_a$=3

|  | $t_1$[P] | $t_2$[A] | $t_3$[A] | $t_4$[NA] |
|---|---|---|---|---|
| $d_{ti}$ | 0 | **2/3** | **1/3** | 0 |
| $a_{ti}$ | 1 | 3/5 | 2/5 | 0 |
| $u_a(t_i,\mathbf{c})$ | 5 | **2** | **2** | 2 |
| $\upsilon_d(t_i,\mathbf{a})$ | 1 | 6/5 | 6/5 | 0 |

(d) IMOA. $\theta_d$=6/5, $\theta_a$=2

|  | $t_1$[P] | $t_2$[A] | $t_3$[A] | $t_4$[NA] |
|---|---|---|---|---|
| $d_{ti}$ | 0 | 2/3 | 1/3 | 0 |
| $a_{ti}$ | 1 | **1/2** | **1/3** | **1/6** |
| $u_a(t_i,\mathbf{c})$ | 5 | 2 | 2 | 2 |
| $\upsilon_d(t_i,\mathbf{a})$ | 1 | **1** | **1** | **1/12** |

(e) MMNA. $\theta_d$=1, $\theta_a$=2

|  | $t_1$[A] | $t_2$[A] | $t_3$[A] | $t_4$[NA] |
|---|---|---|---|---|
| $d_{ti}$ | **3/5** | 2/3 | 1/3 | 0 |
| $a_{ti}$ | 1 | 1/2 | 1/3 | 1/6 |
| $u_a(t_i,\mathbf{c})$ | **2** | 2 | 2 | 2 |
| $\upsilon_d(t_i,\mathbf{a})$ | 1 | 1 | 1 | 1/12 |

(f) IMOP. $\theta_d$=1, $\theta_a$=2

|  | $t_1$[A] | $t_2$[A] | $t_3$[A] | $t_4$[P] |
|---|---|---|---|---|
| $d_{ti}$ | 3/5 | 2/3 | 1/3 | 0 |
| $a_{ti}$ | **6/11** | **3/11** | **2/11** | **1** |
| $u_a(t_i,\mathbf{c})$ | 2 | 2 | 2 | 2 |
| $\upsilon_d(t_i,\mathbf{a})$ | **6/11** | **6/11** | **6/11** | **1/2** |

(g) MMNA. $\theta_d$=6/11, $\theta_a$=2

|  | $t_1$[A] | $t_2$[DS] | $t_3$[A] | $t_4$[P] |
|---|---|---|---|---|
| $d_{ti}$ | **4/5** | **1** | **2/3** | 0 |
| $a_{ti}$ | 6/11 | 3/11 | 2/11 | 1 |
| $u_a(t_i,\mathbf{c})$ | 1 | 1 | 1 | 2 |
| $\upsilon_d(t_i,\mathbf{a})$ | **6/11** | **6/11** | **6/11** | 1/2 |

(h) IMOA. $\theta_d$=6/11, $\theta_a$=1

|  | $t_1$[A] | $t_2$[DS] | $t_3$[A] | $t_4$[P] |
|---|---|---|---|---|
| $d_{ti}$ | 4/5 | 1 | 2/3 | 0 |
| $a_{ti}$ | **1/2** | **1/3** | **1/6** | 1 |
| $u_a(t_i,\mathbf{c})$ | 1 | 1 | 1 | 2 |
| $\upsilon_d(t_i,\mathbf{a})$ | **1/2** | **2/3** | **1/2** | 1/2 |

(i) MMDS. $\theta_d$=1/2, $\theta_a$=1

|  | $t_1$[A] | $t_2$[DS] | $t_3$[A] | $t_4$[A] |
|---|---|---|---|---|
| $d_{ti}$ | 4/5 | 1 | 2/3 | **1/2** |
| $a_{ti}$ | 1/2 | 1/3 | 1/6 | 1 |
| $u_a(t_i,\mathbf{c})$ | 1 | 1 | 1 | **1** |
| $\upsilon_d(t_i,\mathbf{a})$ | 1/2 | 2/3 | 1/2 | 1/2 |

(j) IMOP. $\theta_d$=1/2, $\theta_a$=1

|  | $t_1$[A] | $t_2$[D] | $t_3$[A] | $t_4$[A] |
|---|---|---|---|---|
| $d_{ti}$ | 4/5 | 1 | 2/3 | 1/2 |
| $a_{ti}$ | **3/10** | **1** | **1/10** | **3/5** |
| $u_a(t_i,\mathbf{c})$ | 1 | 1 | 1 | 1 |
| $\upsilon_d(t_i,\mathbf{a})$ | **3/10** | **2** | **3/10** | **3/10** |

(k) MMDS. $\theta_d$=3/10, $\theta_a$=1

|  | $t_1$ | $t_2$ | $t_3$ | $t_4$ |
|---|---|---|---|---|
| $d_{ti}$ | **≈.806** | 1 | **≈.677** | **≈.516** |
| $a_{ti}$ | 3/10 | 1 | 1/10 | 3/5 |
| $u_a(t_i,\mathbf{c})$ | **≈.968** | 1 | **≈.968** | **≈.968** |
| $\upsilon_d(t_i,\mathbf{a})$ | 3/10 | 2 | 3/10 | 3/10 |

(l) IMOA. $\theta_d$=3/10, $\theta_a$≈.968

Figure 1: The example run of the algorithm. Each subfigure shows the current equilibrium, threshold values, and target states *at the end* of the corresponding phase.

**Example 1.** *Consider a game with* $|T| = 4$ *targets,* $n_d = 3$ *defender resources, and* $n_a = 2$ *attacker resources. The utilities are as follows:*

|  | $t_1$ | $t_2$ | $t_3$ | $t_4$ |
|---|---|---|---|---|
| $u_a^u$ | 5 | 4 | 3 | 2 |
| $u_a^c$ | 0 | 1 | 0 | 0 |
| $\Delta u_d$ | 1 | 2 | 3 | .5 |

*Figure 1 shows the sequence of equilibria for different amounts of defender resources computed by the algorithm. We start with the equilibrium for* 0 *defender resources. The attacker attacks the two targets that give him the highest utility, namely,* $t_1$ *and* $t_2$ *(Fig. 1(a)). Since these two targets are attacked with probability* 1*, they are likely to get defended as the number of defender resources increases. Thus, these targets are both in the Pending (P) state. Target* $t_4$ *is in the Untouched (U) state, because* $u_a(t_4) < \theta_a$*, and thus it is neither attacked nor defended.*

*Increase Defender Mass on Pending (IMOP) phase: We increase the number of defender resources and allocate them to the Pending target* $t_2$*, which is currently the most appealing to the defender. We cannot put more than* $1/3$ *defender probability on* $t_2$ *without breaking the attacker's equilibrium condition, so the phase ends at that point. The new equilibrium strategies and updated target states are shown in Figure 1(b).*

*Move Attacker Mass to Newly Attacked (MMNA) phase: At the beginning of this phase, target* $t_2$ *is in the Active (A) state, because both players' utilities for defending/attacking this targets are at the corresponding threshold values. Target* $t_3$ *is in the Newly Attacked (NA) state, because it is currently at the attacker threshold and would start to be attacked if the defender put more probability on* $t_2$*. In this phase, we move* $2/5$ *of the attacker's probability mass from* $t_2$ *to* $t_3$*. As a result, both* $t_2$ *and* $t_3$ *now have the highest marginal defender utility (Fig. 1(c)).*

*Increase Defender Mass on Active (IMOA) phase: We increase the defender probability on Active targets* $t_2$ *and* $t_3$*. Since* $\Delta u_a(t_2) = \Delta u_a(t_3)$*, we have to add the same amount of probability to each of these targets; otherwise, the attacker's best-response condition would be broken. We can add up to* $1/3$ *defender probability to these targets, until the attacker's utility for attacking* $t_4$ *becomes equal to the utility for attacking* $t_2$ *and* $t_3$ *(Fig. 1(d)).*

*MMNA phase: We now move attacker mass from Active targets* $t_2$ *and* $t_3$ *to the Newly Attacked* $t_4$*. To maintain the defender's best-response condition, we need to take mass from* $t_2$ *and* $t_3$ *proportionally to* $1/\Delta u_d(t)$*. We stop when the defender becomes indifferent between* $t_2$*,* $t_3$*, and the Pending target* $t_1$ *(Fig. 1(e)).*

*IMOP phase: We add* $3/5$ *defender mass to* $t_1$*, after which the attacker becomes indifferent between all targets (Fig. 1(f)).*

*MMNA phase: The defender cannot add any mass to her optimal targets* $t_1$*,* $t_2$*, and* $t_3$*, because that would make* $t_4$ *strictly preferred for the attacker, and the attacker's best-response condition would be broken. Therefore, we move attacker mass from* $t_1, t_2, t_3$ *to* $t_4$ *in the right proportions, until the probability on* $t_4$ *reaches* 1 *(Fig. 1(g)).*

*IMOA phase: We can now add defender mass to* $t_1, t_2, t_3$*. That will make* $t_4$ *strictly preferred for the attacker. However, as long as we add mass in the right proportions, the attacker will still be best-responding, because* $t_4$ *is attacked with probability* 1*. We stop when the defender probability on* $t_2$ *becomes* 1 *and target* $t_2$ *transitions to the Defender Saturated (DS) phase (Fig. 1(h)).*

*Move Attacker Mass to Defender Saturated (MMDS) phase: We move attacker mass from* $t_1$ *and* $t_3$ *to* $t_2$*. This does not violate the defender's best-response condition because* $t_2$ *is already fully defended. We stop when the defender becomes* *indifferent between $t_1, t_3$, and $t_4$ (Fig. 1(i)).*

*IMOP phase: We increase the defender mass on $t_4$, until the attacker becomes indifferent between all targets (Fig. 1(j)).*

*MMDS phase: We move attacker mass from Active targets $t_1, t_3, t_4$ to DS target $t_2$ in the proportions that keep the defender indifferent between the three Active targets, until $t_2$ becomes attacked with probability $1$ (Fig. 1(k)).*

*IMOA phase: We add defender mass to $t_1, t_3, t_4$, until all defender mass is allocated. After the end of this phase, the algorithm terminates, and the resulting equilibrium profile is an equilibrium of the game with $3$ defender resources and $2$ attacker resources (Fig. 1(l)).*

## 3.2 Algorithm, Correctness, Runtime

We present the pseudocode for the algorithm in Figure 2. The pseudocode contains the exact definitions of the target states. For proof convenience, we will split the Pending target state into two states: $P_1 = \{t \in P : \upsilon_d(t, a_t) < \theta_d\}$, $P_2 = \{t \in P : \upsilon_d(t, a_t) = \theta_d\}$.

**Theorem 1.** *Throughout the algorithm, the following holds.*

- *UpdateTargetStates always assigns each target to exactly one of the states $U, A, P_1, P_2, NA, DS, D$.*
- *At the end of each phase, if the algorithm does not terminate, then at least one target changes its state.*
- *Each phase terminates.*

*Proof sketch.* It follows from the state definitions that no target can be in two states at the same time. To prove the theorem, we will first show that each target is assigned a state after the initialization phase, and then we will show which targets change states at the end of each phase.

At the beginning of the algorithm, each target is assigned to exactly one state. The $n_a$ targets for which $a_t = 1$ are assigned to the P state, except that targets with $a_t = 1, u_a^u(t) = \theta_a$, if any, are assigned to the A state. The other $|T| - n_a$ targets are assigned to the U state, except that targets with $a_t = 0, u_a^u(t) = \theta_a$, if any, are assigned to the NA state.

Next, we specify all target state changes for each phase and for each termination criterion within each phase.

Next, for each phase, we list, for every termination criterion of that phase, which targets change states (always a nonempty set, unless the algorithm terminates). It is straightforward to check that one of these criteria will always apply.

IMOP phase: (1) $t^*$ becomes A. (2) $t^*$ becomes D. (3) The algorithm terminates.

IMOA: (1) Every $t \in U$ s.t. $u_a^u(t) = \theta_a$ becomes A. (2) Every $t \in A$ s.t. $d_t = 1$ becomes either DS (if $a_t < 1$) or D (if $a_t = 1$). (3) The algorithm terminates.

MMNA: (1) Every $t \in P$ s.t. $\upsilon_d(t, a_t) = \theta_d$ transitions from $P_1$ to $P_2$. (2) Every $t \in NA$ s.t. $\upsilon_d(t, a_t) = \theta_d$ becomes A. (3) $t^*$ transitions from NA to $P_1$.

MMDS: (1) Every $t \in P$ s.t. $\upsilon_d(t, a_t) = \theta_d$ transitions from $P_1$ to $P_2$. (2) If this condition happens, then $\theta_d = 0$, thus $a_t = 0$ for all $t \in P$. Also, $a_t = 0$ for all $t \in A$, and NA is empty. The algorithm will terminate after this phase because $a_t > 0$ implies $d_t = 1$. (3) $t^*$ transitions to D.

DDT: Every $t \in NA$ s.t. $\upsilon_d(t, a_t) = \theta_d$ transitions to A. Every $t \in P$ s.t. $\upsilon_d(t, a_t) = \theta_d$ transitions from $P_1$ to $P_2$.

To complete the proof, we also need to show that no target can change state before the phase is over. This can be done in a straightforward way by carefully checking all state definitions. □

**Theorem 2.** *Throughout the algorithm, the current strategies $\langle \mathbf{d}, \mathbf{a} \rangle$ constitute a Nash equilibrium for the current number of defender resources.*

*Proof.* This follows immediately from the following facts: (1) each target is always in one of the states (Theorem 1), and (2) each state definition implies that the equilibrium condition with respect to the thresholds (beginning of Section 3) is satisfied for such a state. □

**Theorem 3.** *The algorithm terminates after at most $6|T|$ phases, and each phase requires $O(|T|)$ time.*

*Proof.* We can order the 7 possible states as follows: $U < NA < P_1 < P_2 < A < DS < D$. As we can see from the proof of Theorem 1, after each phase (except the last one), some target changes its state to a later state. Thus the algorithm terminates after at most $6|T|$ phases. In each phase, we can calculate directly at what point the phase will terminate, though this in general requires examining all $|T|$ targets. □

## 3.3 Interchangeability

While we have shown how to compute a Nash equilibrium efficiently, a defender may still be unconvinced about whether she actually wants to play her corresponding strategy. For example, if she has a commitment advantage where the attacker observes her distribution before acting, she would prefer to play a Stackelberg strategy; we will return to this in Section 4. However, even if the attacker cannot observe her distribution, she may worry that she is playing her strategy from the "wrong" equilibrium: in general games, if one player plays according to one equilibrium and the other according to another, the result may be disastrous for both (see the game of chicken). In this section, we alleviate this latter concern, by showing that the security games in this paper satisfy the *interchange* property: any combination of equilibrium strategies is, in fact, itself an equilibrium. (This was previously shown for a large class of security games with a single attacker resource [9].)

Suppose $\sigma = \langle \mathbf{d}, \mathbf{a} \rangle$ and $\sigma' = \langle \mathbf{d}', \mathbf{a}' \rangle$ are two NE profiles in a security game with multiple attacker resources. We need to show that $\langle \mathbf{d}', \mathbf{a} \rangle$ and $\langle \mathbf{d}, \mathbf{a}' \rangle$ are also NE profiles of the same game. We first prove that for any target, either the defender probability on that target is the same in all equilibrium profiles, or the attacker probability is the same in all equilibrium profiles, or both.

**Lemma 4.** *If $\sigma = \langle \mathbf{d}, \mathbf{a} \rangle$ and $\sigma' = \langle \mathbf{d}', \mathbf{a}' \rangle$ are two NE profiles, then there is no target $t$ for which both (1) the defender probabilities are different in the two profiles and (2) the attacker probabilities are different in the two profiles. In other words, for any target $t$, at least one of equalities $d_t = d'_t$, $a_t = a'_t$ must hold.*

```
MainRoutine
Initialize
Repeat:
  If (no defender mass is left), return
  UpdateTargetStates
  If (for all t, $a_t > 0$ implies $d_t = 1$),
    distribute the remaining defender
    resources arbitrarily and return
Else if (exists t in P s.t. $\upsilon_d(t, a_t) = \theta_d$), IMOP
Else if ($A \neq \emptyset$, $NA = \emptyset$, $DS = \emptyset$), IMOA
Else if ($A \neq \emptyset$, $NA \neq \emptyset$), MMNA
Else if ($A \neq \emptyset$, $DS \neq \emptyset$), MMDS
Else DDT

Initialize
Order the targets by decreasing $u_a^u(t)$,
    breaking ties arbitrarily
$a_t \leftarrow 1$ for the first $n_a$ targets; $a_t \leftarrow 0$ for the rest
$\theta_a \leftarrow u_a^u(t^{n_a+1})$,
    where $t^{n_a+1}$ is the $(n_a+1)^{st}$ target
$\theta_d \leftarrow \max_t a_t \Delta u_d(t, a_t)$
```

```
UpdateTargetStates
U ← $\{t: u_a(t,d_t) < \theta_a, a_t = 0, \upsilon_d(t,a_t) \le \theta_d, d_t = 0\}$
A ← $\{t: u_a(t,d_t) = \theta_a, \upsilon_d(t,a_t) = \theta_d, d_t < 1\}$
P ← $\{t: u_a(t,d_t) > \theta_a, a_t = 1, \upsilon_d(t,a_t) = \theta_d, d_t < 1\}$
    $\cup \{t: u_a(t,d_t) \ge \theta_a, a_t = 1, \upsilon_d(t,a_t) < \theta_d, d_t = 0\}$
NA ← $\{t: u_a(t,d_t) = \theta_a, a_t < 1, \upsilon_d(t,a_t) < \theta_d, d_t = 0\}$
DS ← $\{t: \upsilon_d(t,a_t) \ge \theta_d, d_t = 1, u_a(t,d_t) = \theta_a, a_t < 1\}$
D ← $\{t: \upsilon_d(t,a_t) \ge \theta_d, d_t = 1, u_a(t,d_t) \ge \theta_a, a_t = 1\}$

IMOP (Increase Defender Mass on Pending)
Choose $t^* \in P$ s.t. $\upsilon_d(t^*, a_t) = \theta_d$
Increase $d_{t^*}$ until one of the following:
  (1) $u_a(t^*, d_{t^*}) = \theta_a$
  (2) $d_{t^*} = 1$
  (3) No defender mass is left

IMOA (Increase Defender Mass on Active)
Simultaneously for all $t \in A$, increase $d_t$, and decrease
$\theta_a$, at relative rates that maintain
$u_a(t, d_t) = \theta_a$ for all t in A, until one of the following:
  (1) For some $t \in U$, $u_a^u(t) = \theta_a$
  (2) For some $t \in A$, $d_t = 1$
  (3) No defender mass is left
```

```
MMNA (Move Attacker Mass to NA)
Choose $t^* \in \arg\max_{t \in NA} \upsilon_d(t, a_t)$
Simultaneously move attacker mass from
all $t \in A$ to $t^*$, and decrease $\theta_d$, at relative
rates that maintain $\upsilon_d(t, a_t) = \theta_d$ for all $t \in A$,
until one of the following:
  (1) For some $t \in P$, $\upsilon_d(t, a_t) = \theta_d$
  (2) $\upsilon_d(t^*, a_{t^*}) = \theta_d$
  (3) $a_{t^*} = 1$

MMDS (Move Attacker Mass to DS)
Choose $t^* \in DS$
Simultaneously move attacker mass from
all $t \in A$ to $t^*$, and decrease $\theta_d$, at relative
rates that maintain $\upsilon_d(t, a_t) = \theta_d$ for all $t \in A$,
until one of the following:
  (1) For some $t \in P$, $\upsilon_d(t, a_t) = \theta_d$
  (2) For some (in fact, all) $t \in A$, $a_t = 0$
  (3) $a_{t^*} = 1$

DDT (Decrease the Defender's Threshold)
Decrease $\theta_d$ until
    $\exists t \in NA \cup P$ s.t. $\upsilon_d(t, a_t) = \theta_d$
```

Figure 2: The algorithm for computing a Nash equilibrium.

*Proof.* The proof is by contradiction. Suppose that there is a target $t$ for which $d_t \neq d'_t$ and $a_t \neq a'_t$. We show that the following four cases must all hold even though two are contradictory.

**Case "$--$":** There exists a target $t_1$ such that $d'_{t_1} < d_{t_1}$ and $a'_{t_1} < a_{t_1}$. In profile $\sigma$, the attacker's utility $u_a(t_1, d_{t_1})$ must be greater than or equal to the threshold value $\theta_a$, because $a_{t_1} > 0$. Similarly, in profile $\sigma'$, the attacker's utility $u_a(t_1, d'_{t_1})$ must be less than or equal to the threshold $\theta'_a$, because $a'_{t_1} < 1$. At the same time, the attacker's utility for attacking $t_1$ is higher in profile $\sigma'$ than in profile $\sigma$, because $d'_{t_1} < d_{t_1}$. Thus, the following three inequalities must hold.

$$\theta_a \le u_a(t_1, d_{t_1})$$
$$u_a(t_1, d_{t_1}) < u_a(t_1, d'_{t_1})$$
$$u_a(t_1, d'_{t_1}) \le \theta'_a$$

It follows from these three inequalities that $\theta'_a > \theta_a$. Because $\sum_t a_t = \sum_t a'_t$ and $a'_{t_1} < a_{t_1}$, there must exist a target $t_2$ such that $a'_{t_2} > a_{t_2}$. Since $a'_{t_2} > 0$, it must be the case that $u_a(t_2, d'_{t_2}) \ge \theta'_a$. Similarly, because $a_{t_2} < 1$, it must be the case that $u_a(t_2, d_{t_2}) \le \theta_a$. Using the last two inequalities and the fact that $\theta'_a > \theta_a$, it follows that $u_a(t_2, d_{t_2}) < u_a(t_2, d'_{t_2})$, which implies $d'_{t_2} < d_{t_2}$. By considering the target $t_2$, it follows that the case "$-+$" must also hold.

**Case "$-+$":** There is a target $t_1$ such that $d'_{t_1} < d_{t_1}$ and $a'_{t_1} > a_{t_1}$. The defender's marginal utility for defending target $t_1$ must be at or above the threshold $\theta_d$ in profile $\sigma$ (because $d_{t_1} > 0$) and at or below the threshold $\theta'_d$ in profile $\sigma'$ (because $d'_{t_1} < 1$). At the same time, since $t_1$ is attacked with a higher probability in $\sigma'$ than in $\sigma$, it must be that $\upsilon_d(t_1, a'_{t_1}) > \upsilon_d(t_1, a_{t_1})$. Thus we have $\theta'_d \ge \upsilon_d(t_1, a'_{t_1}) > \upsilon_d(t_1, a_{t_1}) \ge \theta_d$. Because $\sum_t d_t = \sum_t d'_t = n_d$ and $d'_{t_1} < d_{t_1}$, there must be a target $t_2$ such that $d'_{t_2} > d_{t_2}$. The defender's marginal utility for defending $t_2$ must be at or above the threshold $\theta'_d$ in profile $\sigma'$ (because $d'_{t_2} > 0$) and at or below the threshold $\theta_d$ in profile $\sigma$ (because $d_{t_2} < 1$). Since $\theta'_d > \theta_d$, it follows that $\upsilon(t_2, a'_{t_2}) \ge \theta'_d > \theta_d \ge \upsilon(t_2, a_{t_2})$, which implies $a'_{t_2} > a_{t_2}$. By considering the target $t_2$, it follows that the case "$++$" must also hold.

We can also prove the following two implications similarly to the two cases described above, by reversing the roles of equilibria $\sigma$ and $\sigma'$:

**Case "$++$":** There is a target $t_1$ such that $d'_{t_1} > d_{t_1}$ and $a'_{t_1} > a_{t_1}$. If this case holds, then the case "$+-$" must also hold. This can be proven similarly to the implication "$--$" ⇒ "$-+$", by reversing the roles of equilibria $\sigma$ and $\sigma'$.

**Case "$+-$":** There is a target $t_1$ such that $d'_{t_1} > d_{t_1}$ and $a'_{t_1} < a_{t_1}$. If this case holds, then the case "$--$" must also hold. This can be proven similarly to the implication "$-+$" ⇒ "$++$", by reversing the roles of equilibria $\sigma$ and $\sigma'$.

It follows that if at least one of the cases "$--$", "$-+$", "$++$", "$+-$" holds, then all of them must hold. But if both "$--$" and "$++$" hold, then both inequalities $\theta'_a > \theta_a$ and $\theta_a > \theta'_a$ must hold, which is impossible. Hence, none of the four cases can hold. □

We now show that in an equilibrium the defender obtains the same marginal utility from all targets that have different defender probabilities in a different equilibrium.

**Lemma 5.** *Suppose that $\sigma$ and $\sigma'$ are two NE profiles, and $t_1, t_2$ are two targets such that $d_{t_1} \neq d'_{t_1}$ and $d_{t_2} \neq d'_{t_2}$. Then $\upsilon_d(t_1, a_{t_1}) = \upsilon_d(t_2, a_{t_2}) = \upsilon_d(t_1, a'_{t_1}) = \upsilon_d(t_2, a'_{t_2})$.*

*Proof.* Because $\sum_t d_t = \sum_t d'_t = n_d$, it is enough to show that $\upsilon_d(t_1, a_{t_1}) = \upsilon_d(t_2, a_{t_2})$ holds for any pair of targets $t_1, t_2$ such that $d_{t_1} < d'_{t_1}$ and $d_{t_2} > d'_{t_2}$. (This is because if (say) $d_{t_1} < d'_{t_1}$ and $d_{t_2} < d'_{t_2}$, there must exist a third target $t_3$ with $d_{t_3} > d'_{t_3}$, so that we can then conclude $\upsilon_d(t_1, a_{t_1}) = \upsilon_d(t_3, a_{t_3}) = \upsilon_d(t_2, a_{t_2})$.)

In profile $\sigma$, the defender can shift her probability from $t_2$ to $t_1$, because $d_{t_2} > 0$ and $d_{t_1} < 1$. Since $\sigma$ is an equilibrium profile, the defender must not benefit from such a shift of probability. Thus $v_d(t_1, a_{t_1}) \leq v_d(t_2, a_{t_2})$. Using a similar argument for profile $\sigma'$, we get $v_d(t_1, a'_{t_1}) \geq v_d(t_2, a'_{t_2})$. It also follows from Lemma 4 that $a_{t_1} = a'_{t_1}$ and $a_{t_2} = a'_{t_2}$. Hence, we have $v_d(t_1, a_{t_1}) \leq v_d(t_2, a_{t_2}) = v_d(t_2, a'_{t_2}) \leq v_d(t_1, a'_{t_1}) = v_d(t_1, a_{t_1})$, so it follows that these four quantities are all the same. □

In the following lemma, we show that any defender's NE strategy is a best-response to any attacker's NE strategy.

**Lemma 6.** *If $\sigma = \langle \mathbf{d}, \mathbf{a} \rangle$ and $\sigma' = \langle \mathbf{d}', \mathbf{a}' \rangle$ are two NE profiles, then $\mathbf{d}'$ is a best-response to $\mathbf{a}$.*

*Proof.* We will show that the defender's utility for playing strategy $\mathbf{d}'$ against $\mathbf{a}$ is the same as the defender's utility for playing $\mathbf{d}$ against $\mathbf{a}$. First, note that $u_d(\mathbf{d}', \mathbf{a}) - u_d(\mathbf{d}, \mathbf{a}) = \sum_{t:d'_t \neq d_t} [d'_t - d_t]\, v_d(t, a_t)$. Consider any target $t^*$ such that $d_{t^*} \neq d'_{t^*}$. Using Lemma 5, we can rewrite the difference in the utilities as follows: $u_d(\mathbf{d}', \mathbf{a}) - u_d(\mathbf{d}, \mathbf{a}) = v_d(t_1, a_{t_1}) \sum_{t:d'_t \neq d_t} [d'_t - d_t] = 0$. The last summation is equal to zero because $\sum_t d_t = \sum_t d'_t$. □

The following lemma can be proven similarly to Lemma 6, by switching from the defender's to the attacker's perspective.

**Lemma 7.** *If $\sigma = \langle \mathbf{d}, \mathbf{a} \rangle$ and $\sigma' = \langle \mathbf{d}', \mathbf{a}' \rangle$ are two NE profiles, then $\mathbf{a}'$ is a best-response to $\mathbf{d}$.*

The interchange property follows from Lemmas 6 and 7.

**Theorem 8.** *If $\sigma = \langle \mathbf{d}, \mathbf{a} \rangle$ and $\sigma' = \langle \mathbf{d}', \mathbf{a}' \rangle$ are two NE profiles in a security game with multiple attacker resources, then $\langle \mathbf{d}', \mathbf{a} \rangle$ and $\langle \mathbf{d}, \mathbf{a}' \rangle$ are also NE profiles in that game.*

## 3.4 Experimental Results

We now show experimental results for our implementation of the algorithm (Figure 3). For given $|T|, n_d, n_a$, we randomly draw $u_a^u$ and $u_d^c$ from $\{1, \ldots, 100\}$, and then we randomly draw $u_a^c$ from $\{0, \ldots, u_a^u - 1\}$ and $u_d^u$ from $\{0, \ldots, u_d^c - 1\}$. Each data point averages over 20 games. As a sanity check, our implementation verified that the strategies computed for each game did constitute a Nash equilibrium. The quadratic runtime of the algorithm is reflected in the experimental results. We note that the numbers of pure strategies for the players are $\binom{|T|}{n_d}$ and $\binom{|T|}{n_a}$, so *any* alternative algorithm that is based on writing out the normal form is doomed to exponential space (and hence, time) requirements. The time to compute the $\binom{|T|}{n_d} \times \binom{|T|}{n_a}$ utility matrix of the normal-form game for $n_d = n_a = 10$ is shown in Figure 3 with a dashed line. We can see that our algorithm scales well in the number of targets and attacker resources.

## 4 NP-Hardness of Computing Stackelberg Strategies

We now turn to the problem of computing a defender Stackelberg strategy. $\mathbf{d}$ is an *optimal mixed strategy to commit to* or *Stackelberg* strategy for the defender if there exists some $\mathbf{a}$ that is a best response to $\mathbf{d}$ such that for any alternative strategies $\langle \mathbf{d}', \mathbf{a}' \rangle$, where $\mathbf{a}'$ is a best response to $\mathbf{d}'$, we have $u_d(\mathbf{d}, \mathbf{a}) \geq u_d(\mathbf{d}', \mathbf{a}')$.

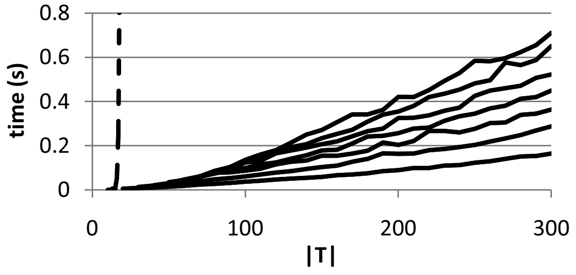

Figure 3: Solid lines: time to compute NE as a function of $|T|$, for $n_a = n_d = 10, \ldots, 70$. Dashed line: time to compute the normal-form of the game with $n_a = n_d = 10$.

**Theorem 9.** *In security games with multiple attacker resources, finding an optimal defender Stackelberg strategy is weakly NP-hard. This holds even when the defender has only one resource, and the defender's utility for a target does not depend on whether she has a resource there (that is, $u_d^c(t) = u_d^u(t)$ for all $t$).*[1][2]

*Proof sketch.* We reduce an arbitrary knapsack problem instance—given by $k$ items, where each item $j$ is defined by a pair $(w_j, v_j)$, and we are asked if there is a subset $S$ of the items with $\sum_{j \in S} w_j \leq 1$ and $\sum_{j \in S} v_j \geq V$—to the following game. We construct a game with $2k$ targets, in which the defender has one resource and the attacker has $k$ resources. Targets $t_1, \ldots, t_k$ correspond to the items in the knapsack, and the utilities are set up as follows for $1 \leq i \leq k$.

$$u_a^u(t_i) = w_i$$
$$u_a^c(t_i) = w_i - 1$$
$$u_d^c(t_i) = u_d^u(t_i) = -v_i$$

Targets $t_{k+1}, \ldots, t_{2k}$ are "dummy" targets, so that for $k+1 \leq i \leq 2k$: $u_a^u(t_i) = u_a^c(t_i) = u_d^c(t_i) = u_d^u(t_i) = 0$.

Let the vector $\mathbf{d}$ represent the defender's strategy, so that $d_i$ is the probability of target $t_i$ being covered. If $d_{t_i} \geq w_i$ for $1 \leq i \leq k$, the attacker attacks a dummy target instead of $t_i$, thus increasing the defender's utility by $v_i$.

There exists attacker's pure-strategy best-response to the defender's Stackelberg strategy (with ties broken in the defender's favor) such that the optimal subset $S$ of the items in the knapsack corresponds to the targets $t_i$ with $1 \leq i \leq k$ and $a_{t_i} = 0$. It can be shown that the defender can get a utility of at least $V - \sum_{i=1}^{k} v_i$ if and only if the knapsack instance has a solution (with value at least $V$). □

## 5 Future Research

Future research can take a number of directions. Can the algorithm that we presented in this paper be generalized to richer settings? For example, is it possible to compute Nash

[1]We have also found a pseudopolynomial-time algorithm (not presented here) for the special case where $u_d^c(t) = u_d^u(t)$ for all $t$.

[2]Note that this violates the assumption that $u_d^c(t) > u_d^u(t)$. It is easy to modify the utilities by $\epsilon$ so that this property holds again and the reduction still works.

equilibria efficiently in cases where either defender resources or attacker resources (or both) are heterogeneous? (With heterogeneous resources, not every resource can defend/attack every target. Our work in this paper assumes homogeneous resources.) Can we efficiently compute them in (restricted) settings with schedules? (A schedule is a subset of targets that can be simultaneously defended/attacked by a single resource.)

In the extended version of the paper by Yin et al. [9], a model with uncertainty about whether the attacker can observe the defender's commitment has been proposed; setting this probability of observability to 0 corresponds to the Nash case, whereas setting it to 1 corresponds to the Stackelberg case. In follow-up work [4], we have designed an algorithm that computes a solution for this richer game model, using Nash and Stackelberg solvers as subroutines. For the case of multiple attacker resources, the algorithm for computing a Nash equilibrium that we have given in this paper can be used as the Nash subroutine; in future research, perhaps an algorithm to compute a Stackelberg strategy can be given that, in spite of the NP-hardness of the problem proven here, nevertheless runs fast in practice.

Beyond security games, it is interesting to investigate further the relationship between Stackelberg strategies and Nash equilibrium strategies, especially regarding whether they can be efficiently computed. We now know of settings where Stackelberg strategies are easier to compute (general two-player normal-form games) as well as settings where Nash equilibria are easier to compute (this paper). Is there a general principle that informs us when each one is easy to compute?

## 6 Acknowledgements

We would like to acknowledge NSF IIS-0812113, CAREER-0953756, ARO 56698-CI, and an Alfred P. Sloan Research Fellowship for support. Any opinions, findings, and conclusions or recommendations in this document are those of the authors and do not necessarily reflect views of the funding agencies. We also thank the anonymous reviewers for valuable suggestions.

## A An Example in Which the Defender's Nash Equilibrium Strategy is Different from the Stackelberg Strategy

Consider the following example game (from the extended version of Yin et al. [9]) in which the defender's Nash equilibrium strategy is different from the defender's Stackelberg (SSE) strategy. In this example, the defender has one resource, the attacker has two resources, and the utilities are set up as shown in the following table.

| | $\mathbf{t_1}$ | | $\mathbf{t_2}$ | | $\mathbf{t_3}$ | |
|---|---|---|---|---|---|---|
| | Cov. | Uncov. | Cov. | Uncov. | Cov. | Uncov. |
| **Def** | 0 | −2 | −9 | −10 | 0 | −1 |
| **Att** | 5 | 6 | 2 | 4 | 1 | 3 |

In this game, the defender has a unique Nash equilibrium strategy and a unique Stackelberg strategy, which can be computed as follows (quoting the extended version of Yin et al. [9]):

> $t_1$ must be attacked with probability 1. Because $\Delta u_d(t_1) = 2 > 1 = \Delta u_d(t_2) = \Delta u_d(t_3)$, in NE, this implies that the defender must put her full probability 1 on $t_1$. Hence, the attacker will attack $t_2$ with his other resource. So, the unique NE profile is $\langle(1,0,0),(1,1,0)\rangle$.
>
> In contrast, in SSE, the defender's primary goal is to avoid an attack on $t_2$, which requires putting probability at least .5 on $t_2$ (so that the attacker prefers $t_3$ over $t_2$). This will result in $t_1$ and $t_3$ being attacked; the defender prefers to defend $t_1$ with her remaining probability because $\Delta u_d(t_1) = 2 > 1 = \Delta u_d(t_3)$. Hence, the unique SSE profile is $\langle(.5,.5,0),(1,0,1)\rangle$.

## References

[1] X. Chen and X. Deng. Settling the complexity of two-player Nash equilibrium. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 261–272, 2006.

[2] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 82–90, Ann Arbor, MI, USA, 2006.

[3] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 689–696, Budapest, Hungary, 2009.

[4] D. Korzhyk, V. Conitzer, and R. Parr. Solving Stackelberg games with uncertain observability. In *AAMAS*, Taipei, Taiwan, 2011.

[5] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordóñez, and S. Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *AAMAS*, pages 895–902, Estoril, Portugal, 2008.

[6] J. Pita, M. Jain, F. Ordóñez, C. Portway, M. Tambe, and C. Western. Using game theory for Los Angeles airport security. *AI Magazine*, 30(1):43–57, 2009.

[7] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordonez, and M. Tambe. IRIS - a tool for strategic security allocation in transportation networks. In *AAMAS - Industry Track*, pages 37–44, 2009.

[8] B. von Stengel and S. Zamir. Leadership games with convex strategy sets. *Games and Economic Behavior*, 69:446–457, 2010.

[9] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS*, pages 1139–1146, Toronto, Canada, 2010.