INTERFACES
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0092-2102 | EISSN 1526-551X | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000

Software Assistants for Randomized Patrol Planning for The LAX Airport Police and The Federal Air Marshals Service

Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Fernando Ordóñez, Milind Tambe
{manish.jain,jasontts,jpita,kiekintv,srathi,fordon,tambe}@usc.edu
University of Southern California
Los Angeles, CA - 90089

Security at major locations of economic or political importance is a key concern around the world, particularly given the increasing threat of terrorism. Limited security resources prevent full security coverage at all times, which allows adversaries to observe and exploit patterns in patrolling or monitoring, e.g. they can plan an attack that avoids existing patrols. An important method of countering the surveillance capabilities of an adversary is to use randomized security policies that are more difficult to predict and exploit. We describe two deployed applications that assist security forces in randomizing their operations based on fast algorithms for solving large instances of Bayesian Stackelberg games. The first is the ARMOR system (Assistant for Randomized Monitoring over Routes), which has been successfully deployed since August 2007 at the Los Angeles International Airport (LAX). This system is used by airport police to randomize the placement of checkpoints on roads entering the airport, and the routes of canine unit patrols in the airport terminals. The IRIS system (Intelligent Randomization in Scheduling) is designed to randomize flight schedules for the Federal Air Marshals Service (FAMS). IRIS has been deployed in a pilot program by FAMS since October 2009 to randomize schedules of air marshals on international flights. These assistants share several key features: (i) they are based on Stackelberg game models to intelligently weight the randomized schedules, (ii) they use efficient mixed-integer programming formulations of the game models to enable fast solutions for large games, and (iii) they allow for interactive manipulation of the domain constraints and parameters by the users. This paper examines the design choices, information, and evaluation that went into building these effective applications.

1. Introduction

Protecting critical infrastructure and targets such as airports, historical landmarks, power generation facilities, and political figures is a challenging task for police and security agencies worldwide. The growing threat of international terrorism has exacerbated this challenge in recent years. Transportation networks such as buses, trains, and airplanes carry millions of people per day to their destinations, also making them a prime target for terrorists and extremely difficult to protect for law enforcement agencies. In 2001, the 9/11 attack on the World Trade Center in New York City via commercial airliners resulted in $27.2 billion of direct short term costs (26) as well as a government-reported 2,974 lives lost. The 2004 Madrid commuter train bombings resulted in 191 lives lost, 1755 wounded, and an estimated cost of 212 million Euros (12). Finally, in the 2005 London subway and bus bombings, 52 lives were lost, 700 were wounded, and there was an estimated economic cost of 2 billion pounds (42).

Measures for protecting potential target areas include monitoring entrances or inbound roads, checking inbound traffic and patrols aboard transportation vehicles. However, limited resources imply that it is typically impossible to provide full security coverage at all times. Furthermore, adversaries can observe security arrangements over time and exploit any predictable patterns to their advantage. One way to mitigate the ability of adversaries to exploit patterns is the judicious use of randomization in scheduling the actions of security forces. For example, police patrols, baggage screenings, vehicle checkpoints, and other security procedures are often randomized. However, security forces face many difficulties in effectively randomizing their operations. In our work on randomized patrol planning, we have developed two software assistants,

ARMOR and IRIS, that address many of the key difficulties of randomization and provide an easy-to-use solution for security forces.

One important question in randomizing security operations is how the different actions should be weighted. One obvious approach is to "roll dice" effectively using a uniform random policy where all possible targets or entry points are treated the same. However, this is severely limited in that it fails to take into account that some targets are more attractive or vulnerable than others. As a result, valuable security resources are under-utilized by protecting relatively unimportant targets. Instead, the defense strategy should place greater emphasis on protecting high-valued targets. For example, a more sophisticated security policy could weight the protection provided for each target based on the ratio of the value of the targets. However, this weighted randomization still fails to account for the possibility that the attacker is intelligent, and will update his/her strategy based on knowledge of the security strategy. Asking a human to generate a random security policy has additional drawbacks. Humans are not good at generating truly random behavior (47, 43), and can easily fall into predictable patterns. Furthermore, in transportation networks and many other security domains, the problem of scheduling security forces is prohibitively large, even without considering randomization. Creating a schedule by hand is a costly and labor-intensive process.

We address these difficulties by developing software assistants for scheduling security forces. These assistants use game-theoretic models and solution algorithms to determine good randomization strategies that take into account target values and assume intelligent adversary responses to security measures. Game theory is a well-established paradigm for reasoning about situations with multiple self-interested decision makers (18). We model security games as Stackelberg games (45) between the defender (i.e., the security forces), and the attacker (i.e., a terrorist adversary). Stackelberg games are a bilevel model (9) that account for the ability of an attacker to gather information about the defense strategy before planning an attack. These games specify different payoff values for both players in the event of an attack on every potential target. Extending these games to Bayesian Stackelberg games (17) allows us to capture uncertainty about these payoffs in the game model. Solutions to these games provide a randomized policy for the defense strategy, which can be used to generate specific schedules for security patrols.

We apply this game-theoretic approach in two different software solutions that provide assistance in scheduling security operations. The ARMOR program (35) was developed for the Los Angeles airport police, and randomizes checkpoints on the roadways entering the airport and canine patrol routes within the airport terminals. The IRIS program (44) was developed for the Federal Air Marshal Service (FAMS) to assist with randomly scheduling the air marshals on flights. These software assistants are interactive and domain experts can change domain parameters when necessary. Underlying each of these tools is a model of the domain as a Bayesian Stackelberg game, along with fast solution algorithms for computing an optimal solution to the game model. These algorithms use various techniques for exploiting structure in the security domains to speed up the computation and enable large real-world problem instances to be solved in reasonable amounts of time (33, 31, 22).

ARMOR has been in regular use by the airport police since its deployment in August 2007, and IRIS has been deployed in limited use since October 2009. As of January 2009, 285,589 vehicles have been inspected at checkpoints scheduled by ARMOR since it was deployed. In the month of January 2009 alone, seven stops discovered one or more firearms, with five of them leading to arrests. In one instance, the person arrested was carrying 22 guns, including a loaded assault rifle.

Our approach of using Stackelberg games to model real world security problems is applicable in a wide range of domains with similar attributes, including (i) intelligent players, (ii) one player's strategy is observable by the other player, (iii) varying preferences among targets, (iv) and full coverage of all targets is not feasible. Some examples of similar security situations include security in computer networks, checkpoints at subway stations, security inspections at ports and monitoring of other mediums of public transport. A deterministic strategy is a weakness for the defender when there is incomplete coverage. Furthermore, variations in target values are ubiquitous in these domains, so intelligent weighted randomization is able to provide a greater degree of protection.

The rest of the article is organized as follows. Related work is discussed in Section 2. The Bayesian Stackelberg game methodology, its formal definition and the technical formulation of the security game is discussed in Section 3. The LAX and FAMS domains are described in Section 4. In Section 5, we present the system architecture of the two software assistants. Our evaluations are presented in Section 6 and are followed by a summary in Section 7.

2. Related Work

There exists related work both in game-theoretic as well as non-game theoretic literature studying the security domain. Much of this work is theoretical analysis of hypothetical scenarios, while our work focuses on developing tools for use in real-world security operations. This required us to address many practical aspects of the problem that only arise in fielded applications.

There are three main areas of related work. The first apply optimization techniques to model the security domain, but do not address the strategic aspects of the problem. These methods provide a randomization strategy for the defender, but they do not take into account the fact that the adversaries can observe the defender's actions and then adjust their behavior. Examples of such approaches include (36, 33) which are based on learning, Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs). As part of this work, the authors model the patrolling problem with locations and varying incident rates in each of the locations and solve for optimal routes using a MDP framework. Another example is the "Hypercube Queueing Model" (24) which is based on queueing theory and depicts the detailed spatial operation of urban police departments and emergency medical services. It has found application in police beat design, in allocation of patrolling time, etc. Such frameworks can address many of the problems we raise, including different target values and increasing uncertainty by using many possible patrol routes. However, they fail to account for the possibility that an intelligent attacker will observe and exploit patterns in the security policy. If a policy is based on the historical frequency of attacks, it is essentially a reactive policy and an intelligent attacker will always be one step ahead.

A second set of work uses Stackelberg games to model a variety of security domains. Bier et al. (11) give a strong endorsement of this type of modeling for security problems. Game-theoretic models have been applied in a variety of homeland security settings, such as protecting critical infrastructure (16, 35, 28). Lawrence et al (49) apply Stackelberg games in the context of screening visitors entering the US. In their work, they model the U.S. Government as the leader who specifies the biometric identification strategy to maximize the detection probability using finger print matches, and the follower is the terrorist who can manipulate the image quality of the finger print. They have also been used for studying missile defense systems (14) and for studying the development of an adversary's weapon systems (15). A family of Stackelberg games known as inspection games is closely related to the security games we are interested in and includes models of arms inspections and border patrols (7). Another recent work is on randomized security patrolling using Stackelberg games for generic "police and robbers" scenario (19) and perimeter patrols (6). Our work differs from the previous work in two main aspects. First, we use a new, more efficient game representation and MILP for modeling and solving the Stackelberg games to enable systems to scale to complex real-world situations. Second, we model the game with defender actions that incorporate the domain constraints (e.g. scheduling constraints) to more accurately model the specific games we are interested in.

The third area of related work is the application of game theoretic techniques that are not based on Stackelberg games to security applications. Security problems are increasingly studied using game-theoretic analysis, ranging from computer network security (48, 38) to terrorism (37). Babu et al (8) have worked on modeling passenger security system at US airports using linear programming approaches, however, their objective is to classify the passengers in various groups and then screen them based on the group they belong to. Thus, although game theory has been used in security domains in the past, our work focuses on overcoming the challenges that arise from its application in the real-world.

3. Methodology

ARMOR and IRIS build on the game-theoretic foundations to reason about multiple types of players – the police force and the adversary – to provide a randomized security policy. The algorithms we use in our applications build on several years of research reported in the Autonomous Agents and Multiagent Systems (AAMAS) conference main track and workshops (32, 33, 31). Although, the major developments in this line of research are the new algorithms that are at the heart of the ARMOR and IRIS systems, we first explain how a security domain can be modeled as a Bayesian Stackelberg game.

3.1. Stackelberg Equilibrium

We begin by defining a normal-form Stackelberg game. A generic Stackelberg game has two players, a *leader*, Θ, and a *follower*, Ψ. These players need not represent individuals, but could also be groups that cooperate to execute a joint strategy, such as a police force or terrorist organization. Each player has a set of possible *pure strategies*, denoted $\sigma_\Theta \in \Sigma_\Theta$ and $\sigma_\Psi \in \Sigma_\Psi$. A *mixed strategy* allows a player to play a probability distribution over pure strategies, denoted $\delta_\Theta \in \Delta_\Theta$ and $\delta_\Psi \in \Delta_\Psi$. Payoffs for each player are defined over all possible joint pure-strategy outcomes: $\Omega_\Theta : \Sigma_\Psi \times \Sigma_\Theta \rightarrow \mathcal{R}$ for the defender and similarly for each attacker. The payoff functions are extended to mixed strategies in the standard way by taking the expectation over pure-strategy outcomes. The follower can observe the leader's strategy, and then act in a way to optimize its own payoffs. Formally, the attacker's strategy in a Stackelberg security game becomes a function that selects a strategy for each possible leader strategy: $F_\Psi : \Delta_\Theta \rightarrow \Delta_\Psi$.

The most common solution concept in game theory is a *Nash equilibrium*, which is a profile of strategies for each player in which no player can gain by unilaterally changing to another strategy (29). Stackelberg equilibrium is a refinement of Nash equilibrium specific to Stackelberg games. It is a form of sub-game perfect equilibrium in that it requires that each player select the best-response in any subgame of the original game (where subgames correspond to partial sequences of actions). The effect is to eliminate equilibrium profiles that are supported by non-credible threats off the equilibrium path. Subgame perfection is a natural requirement, but it does not guarantee a unique solution in cases where the follower is indifferent among a set of strategies. The literature contains two form of Stackelberg equilibria that identify unique outcomes, first proposed by Leitmann (25), and typically called "strong" and "weak" after Breton et. al. (13). The strong form assumes that the follower will always choose the optimal strategy for the leader in cases of indifference, while the weak form assumes that the follower will choose the worst strategy for the leader. Unlike the weak form, strong Stackelberg equilibria are known to exist in all Stackelberg games (10). A standard argument suggests that the leader is often able to induce the favorable strong form by selecting a strategy arbitrarily close to the equilibrium which causes the follower to strictly prefer the desired strategy (46). We adopt strong Stackelberg equilibrium as our solution concept in part for these reasons, but also because it is the most commonly used in related literature (29, 17, 30).

Definition 1 *A set of strategies* (δ_Θ, F_Ψ) *form a* Strong Stackelberg Equilibrium *(SSE) if they satisfy the following:*

1. *The leader plays a best-response:*
$\Omega_\Theta(\delta_\Theta, F_\Psi(\delta_\Theta)) \geq \Omega_\Theta(\delta'_\Theta, F_\Psi(\delta'_\Theta)) \, \forall \, \delta'_\Theta \in \Delta_\Theta$.
2. *The follower plays a best-response:*
$\Omega_\Psi(\delta_\Theta, F_\Psi(\delta_\Theta)) \geq \Omega_\Psi(\delta_\Theta, \delta_\Psi) \, \forall \, \delta_\Theta \in \Delta_\Theta, \delta_\Psi \in \Delta_\Psi$.
3. *The follower breaks ties optimally for the leader:*
$\Omega_\Theta(\delta_\Theta, F_\Theta(\delta_\Theta)) \geq \Omega_\Theta(\delta_\Theta, \delta_\Psi) \; \forall \; \delta_\Theta \in \Delta_\Theta, \delta_\Psi \in \Delta^*_\Psi(\delta_\Theta)$*, where* $\Delta^*_\Psi(\delta_\Theta)$ *is the set of follower best-responses, as above.*

Whether or not the Stackelberg leader benefits from the ability to commit depends on whether commitment to mixed strategies is allowed. Committing to a pure strategy can be either good or bad for the leader; for example, in the "Rock, Paper, and Scissors" game, forcing commitment to a pure strategy would guarantee a loss. However, it has been shown that the ability to commit to a mixed strategy always weakly increases

the leader's payoffs in equilibrium profiles of the game (46). In the context of a Stackelberg security game, a deterministic policy is a liability for the defender (the leader), but a credible randomized security policy is an advantage. Our model allows commitment to mixed strategies by the defender.

The Bayesian extension to the Stackelberg game allows for multiple types of players, with each type associated with its own payoff values. For the security games of interest in this paper, we assume that there is only one leader type (e.g. only one police force), although there are multiple follower types (e.g. multiple adversary types trying to infiltrate security). The set of follower types is denoted by Γ. Each type γ is represented by a different payoff matrix. The leader does not know the follower's type. The goal is to *find the optimal mixed strategy* for the leader to commit to, given that each follower type will know the mixed strategy of the leader when choosing its own strategy. Payoffs for each type are defined over all possible joint pure-strategy outcomes: $\Omega_\Theta : \Sigma_\Psi^\Gamma \times \Sigma_\Theta \rightarrow \mathcal{R}$ for the defender and similarly for each attacker type. The leader's best response is now a weighted best response to the followers' responses, where the weights are based on the probability of occurrence of each type. The strategy of each attacker type γ becomes: $F_\Psi^\gamma : \Delta_\Theta \rightarrow \Delta_\Psi^\gamma$, which still satisfies constraints 2 and 3 in Definition 1.

3.2. Security Game Representation

In a security game, a defender must perpetually defend the site in question, whereas the attacker is able to observe the defender's strategy and attack when success seems most likely. This fits neatly into the description of a Stackelberg game if we map the attackers to the follower's role and the defender to the leader's role (7, 16). The actions for the security forces represent the action of scheduling a patrol or checkpoint, e.g. a checkpoint at the LAX airport or a federal air marshal scheduled to a flight. The actions for an adversary represent an attack at the corresponding infrastructural entity. The strategy for the leader is a mixed strategy spanning the various possible actions.

There are two major problems with using conventional methods to represent security games in normal form. First, many solution methods require the use of a Harsanyi transformation when dealing with Bayesian games (20). The Harsanyi transformation converts a Bayesian game into a normal-form game, but the new game may be exponentially larger than the original Bayesian game. Our compact representation avoids this Harsanyi transformation, and instead we directly operate on the Bayesian game. Operating directly on the Bayesian representation is possible in our model because the evaluation of the leader strategy against a Harsanyi-transformed game matrix is equivalent to its evaluation against each of the game matrices for the individual follower types. (For more details, see the Appendix; a further detailed explanation appears in (30)). The second problem arises that the defender has many possible resources to schedule in the security policy. This can also lead to a combinatorial explosion in a standard normal-form representation. For example, if the leader has m resources to defend n entities, then normal-form representations model this problem as a single leader with $\binom{n}{m}$ rows, each row corresponding to a leader action of covering m targets with security resources. However, in our compact representation, the game representation would only include n rows, each row corresponding to whether the corresponding target was covered or not. Such a representation is equivalent to the normal form representation for the class of problems we address in this work (see (22) for additional details). This compactness in our representation is possible because the payoffs for the leader in these games simply depend on whether the attacked target was covered or not, and not on what other targets were covered (or not covered). The representation we use here avoids both of these potential problems, using methods similar to other compact representations for games (23, 21).

We now introduce our compact representation for security games. Let $T = \{t_1, \ldots, t_n\}$ be a set of *targets* that may be attacked, corresponding to pure strategies for the attacker. The defender has a set of resources available to *cover* these targets, $R = \{r_1, \ldots, r_m\}$ (for example, in the FAMS domain, targets could be flights and resources could be federal air marshals). Associated with each target are four payoffs defining the possible outcomes for an attack on the target, as shown in Table 1. There are two cases, depending on whether or not the target is covered by the defender. The defender's payoff for an uncovered attack when facing an adversary of type γ is denoted $U_\Theta^{\gamma,u}(t)$, and for a covered attack $U_\Theta^{\gamma,c}(t)$. Similarly, $U_\Psi^{\gamma,u}(t)$ and $U_\Psi^{\gamma,c}(t)$ are the payoffs of the attacker.

	Covered	Uncovered
Defender	5	–20
Attacker	–10	30

Table 1 Example payoffs for an attack on a target.

A crucial feature of the model is that payoffs depend only on the target attacked, and whether or not it is covered by the defender. The payoffs do *not* depend on the remaining aspects of the schedule, such as whether any unattacked target is covered or which specific defense resource provides coverage. For example, if an adversary succeeds in attacking Terminal 1, the penalty for the defender is the same whether the defender was guarding Terminal 2 or 3. Therefore, from a payoff perspective, many resource allocations by the defender are identical. We exploit this by summarizing the payoff-relevant aspects of the defender's strategy in a *coverage* vector, C, that gives the probability that each target is covered, c_t. The analogous attack vector A^γ gives the probability of attacking a target by a follower of type γ. We restrict the attack vector for each follower type to attack a single target with probability 1. This is without loss of generality because a strong Stackelberg equilibrium (SSE) solution still exists under this restriction (30). Thus, the follower of type γ can choose any pure strategy $\sigma_\Psi^\gamma \in \Sigma_\Psi^\gamma$, that is, attack any one target from the set of targets.

The payoff for a defender when a specific target t is attacked by an adversary of type γ is given by $U_\Theta^\gamma(t, C)$ and is defined in Equation 1. Thus, the expectation of $U_\Theta^\gamma(t, C)$ over t gives U_Θ^γ, which is the defender's expected payoff given coverage vector C when facing an adversary of type γ whose attack vector is A^γ. U_Θ^γ is defined in Equation 2. The same notation applies for each follower type, replacing Θ with Ψ. Thus, $U_\Psi^\gamma(t, C)$ gives the payoff to the attacker when a target t is attacked by an adversary of type γ. We will see $U_\Psi^\gamma(t, C)$ and $U_\Theta^\gamma(t, C)$ used in the MILP discussed later and provided in detail in the Appendix (Section 9). We also define the useful notion of the *attack set* in Equation 3, $\Lambda^\gamma(C)$, which contains all targets that yield the maximum expected payoff for the attacker type γ given coverage C. This *attack set* is used by the adversary to break ties when calculating a strong Stackelberg equilibrium. Moreover, in these security games, exactly one adversary is attacking in one instance of the game; however, the adversary could be of any type and the defender does not know the type of the adversary faced.

$$U_\Theta^\gamma(t, C) = c_t U_\Theta^{\gamma,c}(t) + (1 - c_t) U_\Theta^{\gamma,u}(t) \tag{1}$$

$$U_\Theta^\gamma(C, A^\gamma) = \sum_{t \in T} a_t^\gamma \cdot (c_t \cdot U_\Theta^{\gamma,c}(t) + (1 - c_t) U_\Theta^{\gamma,u}(t)) \tag{2}$$

$$\Lambda^\gamma(C) = \{t : U_\Psi^\gamma(t, C) \geq U_\Psi^\gamma(t', C) \, \forall \, t' \in T\}. \tag{3}$$

In a strong Stackelberg equilibrium, the attacker selects the target in the attack set with maximum payoff for the defender. Let t^* denote this optimal target. Then the expected SSE payoff for the defender when facing this adversary of type γ with probability p^γ is $\hat{U}_\Theta^\gamma(C) = U_\Theta^\gamma(t^*, C) \times p^\gamma$, and for the attacker $\hat{U}_\Psi^\gamma(C) = U_\Psi^\gamma(t^*, C)$.

4. Modeling the LAX and FAMS Domains

In this section, we describe both security domains (LAX and FAMS) in detail, and discuss how we represent these domains using the framework of Stackelberg security games.

4.1. Domain Description

Both LAX and FAMS are security scenarios which share important characteristics. There is a leader/follower dynamic between the security forces and terrorist adversaries in both cases, since LAX and FAMS are both concerned about surveillance and insider threats. In both instances there are limited resources available to protect a very large space of possible targets, so it is not possible to provide complete

(a) LAX Checkpoint (b) Canine Patrol

Figure 1 Security Checkpoints and Canine Patrols at LAX

coverage. Finally, the targets in question clearly have different values and vulnerabilities in each domain. For all of these reasons, Stackelberg games are an ideal model for accurately capturing the interaction between security forces and adversaries in both the LAX and FAMS domains.

There also exist some important differences between these domains that must be considered in the solutions we provide. One is the scale of the problem; at LAX, there are eight terminals that must be protected, but the air marshals are responsible for protecting tens of thousands of commercial flights each day. This is also an issue for specifying the model. Domain experts for LAX need to specify a fairly small number of payoffs, while FAMS needs to evaluate the different values of thousands of potential targets. This leads us to consider different methods and interfaces for constructing game models in each case. Finally, the FAMS domain requires more complex reasoning about spatial and temporal constraints in how resources are scheduled (for example, a given marshal cannot be assigned to two flights with overlapping time schedules). We now describe each domain in more detail.

4.1.1. LAX Domain Description: LAX is the fifth busiest airport in the United States, the largest destination airport in the United States, and serves 60-70 million passengers per year (1, 39). LAX is known to be a prime terrorist target on the west coast of the United States, with multiple arrests of plotters attempting to attack LAX (39). To protect LAX, the airport police have designed a security system that utilizes multiple rings of protection. As is evident to anyone traveling through the airport, these rings include such things as vehicular checkpoints, police units patrolling the roads to the terminals, patrolling inside the terminals (with canines), and security screening and bag checks for passengers. Airport police use intelligent randomization within two of these rings: (1) placing vehicle checkpoints on inbound roads that service the LAX terminals, including both location and timing (a checkpoint is shown in Figure 1(a)) (2) scheduling patrols for bomb-sniffing canine units at the different LAX terminals (as shown in Figure 1(b)). The numbers of available vehicle checkpoints and canine units are limited by resource constraints, so randomization is used as a method to increase the effectiveness of these resources while avoiding patterns in the scheduled deployments.

The eight different terminals at LAX have very different characteristics, leading to different assessments of the value/risk for each terminal. For example, some terminals have international flights, while others serve only domestic flights. The terminals have varying physical size, passenger loads, and levels of foot traffic. All of these factors may contribute to the assessments used in determining a security policy. Uncertainty about the adversary was identified by airport police as a key problem to address. For example, there may be both hard-line, well-funded international terrorists planning attacks as well as amateur individuals. The payoff values for different attack scenarios may depend on the type of attacker and their capabilities.

4.1.2. FAMS Domain Description: The Federal Air Marshals Service (FAMS) places undercover law enforcement personnel aboard flights originating in and departing from the United States to dissuade potential aggressors and prevent an attack should one occur (4). The exact methods used to evaluate the risks posed by individual flights is not made public by the service, but we can identify many factors that might influence such an evaluation. For example, flights have different numbers of passengers, and some fly over densely populated areas while others do not. International flights also serve different countries, which may pose different risks. Special events can also change the risks for particular flights at certain times (2).

The scale of the domain is massive. There are currently tens of thousands of commercial flights scheduled each day, and public estimates state that there are thousands of air marshals. Air marshals must be scheduled on tours of flights that obey various constraints (e.g., the time required to board, fly, and disembark). Simply finding schedules for the marshals that meet all of these constraints is a computational challenge. Our task is made more difficult by the need to find a randomized policy that meets these scheduling constraints, while also accounting for the different values of each flight.

4.2. Game Models

We now describe the instantiation of each of these domains using a specific Stackelberg game model. This involves specifying the possible targets that could be attacked, the defense resources and constraints on how they may be scheduled, and the payoffs that describe the outcomes of attacks on each target for both the defender and the attacker. We rely on domain experts to provide the values necessary to specify these game models. In some cases these values change over time, so we provide interfaces for the users to input the key parameters, which are then used to populate the game model at runtime. Once we have a game model, we compute a solution using the ERASER-C method described in the Appendix (Section 9). This model returns optimal coverage probabilities for each target. By sampling according to these probabilities, we obtain an explicit schedule for the security forces (e.g., checkpoints or air marshals).

4.2.1. LAX Game Model: We now describe how the problem of scheduling vehicle checkpoints at the LAX airport is modeled as a Bayesian Stackelberg game. The canine model is omitted because the model is very similar. At LAX, there are several inbound roads approaching the terminals where police can set up checkpoints. The adversary chooses to attack through one of these roads or "none", and the police place up to $m < n$ checkpoints on these roads. Thus, we define the set of actions for the police Σ_Θ to be this set of n roads.

We model Γ different types of attackers with different payoff functions, representing different capabilities and preferences for the attacker. If an adversary of type $\gamma \in \Gamma$ attacks road i and the LAX police have not placed a checkpoint on road i, then the police receive a payoff of $U_\Theta^{u,\gamma}(i)$ and the adversary receives a payoff of $U_\Psi^{u,\gamma}(i)$. If there is a checkpoint on road i, then the police receive payoff $U_\Theta^{c,\gamma}(i)$ and the adversary gets a payoff $U_\Theta^{c,\gamma}(i)$. The payoff values used in the model depend on several factors, including (i) the likelihood of a LAX police checkpoint intercepting an adversary that is crossing that checkpoint (which may depend on traffic volume); (ii) the damage the adversary can cause if it attacks via a particular inbound road; (iii) type of adversary, i.e., adversary capability. These factors are provided as inputs to the system by domain experts, and used to calculate the payoff values for each road. The game is not necessarily zero-sum. Even if the adversary is caught, there may be some benefits due to publicity or other considerations. Ideally, we would be able to obtain estimates of the adversaries' payoffs directly from these players. However, this is impossible in practice so we rely on the domain experts to provide the most informed estimates possible based on intelligence information.

The different attacker types in the Bayesian model are used to represent very distinct groups of adversaries. For example, a hard-core, well-financed adversary could inflict significant damage on LAX, so the payoff values for an attack by this type of adversary have a greater magnitude than the payoffs for an amateur attacker. In addition to specifying the payoff functions for each attacker type, we also require a probability distribution over the possible types. For example, if these are the only two types of adversaries faced, then a 20-80 split of probability implies that while there is a 20% chance that the LAX police face

a hard-core adversary, and an 80% chance that they face an amateur attacker. These probabilities are also provided by the domain experts.

4.2.2. FAMS Game Model: We can also model the FAMS domain as a Stackelberg game. The targets in this domain are n flights, and the attacker chooses to attack one of these flights. The FAMS have $m < n$ air marshals that may be assigned to protect these flights. However, the schedules that the air marshals can actually fly are subject to certain logistical constraints. For example, an air marshal that is current in Los Angeles can only leave on flights that are outbound from this area. Similarly, there are timing constraints that must be modeled. A single air marshal cannot be scheduled to fly on two flights with overlapping times (plus some window before and after the actual flight time).

We model the constraints on how air marshals can be assigned to cover flights by introducing the notions of *schedules* and *resource types* into the game model. Each air marshal can cover one schedule, consisting of a legal tour of flights that returns to the origin city. Air marshals may be of different types, and each type is able to cover a different set of schedules. Additional details about schedules and resource types are given in the Appendix (Section 9). The user of the system is able to specify as input various parameters that define which schedules are legal, how many marshals are available, and what schedules they are able to fly. These parameters are translated into the specific constraints in the game model during a pre-processing phase.

The payoffs for the FAMS game are defined similarly to those for the ARMOR game, based on whether or not there is a marshal on a flight i that is attacked by the adversary. As discussed in Section 4.1.2, the exact methods used to arrive at these values are sensitive information. However, there are many intuitive factors that could be used to estimate payoff values including information about the flight itself as well as the capabilities of the air marshals and potential adversaries. The relevant parameters, discussed in Section 5.1.2, were defined in discussion with domain experts, and the specific values are provided as input by the user as part of the scheduling process.

5. Software Assistants

We now describe in detail the system architecture for each of the two software assistants, focusing primarily on the ARMOR system but providing some discussion of IRIS as a point of comparison. We paid particular attention to organization acceptance during the development process. The end users of both ARMOR and IRIS are security officers, and the system must be simple enough for them to be comfortable using it on a regular basis. In particular, the systems are designed to hide as much of the complexity of the game-theoretic models as possible, while still allowing enough flexibility for the users to input important parameters that change regularly. This required considerable effort in both user interface design and identifying ways to simplify and reduce the inputs required by the system to specify a game model. In the case of IRIS, it was also very important to build in functionality to import data from other systems to ease the burden of data entry (e.g., importing flight information from existing databases). Finally, the schedules that the system produces must be presented in a format that is easy to understand, with tools that allow final modifications if necessary.

Both ARMOR and IRIS are stand-alone desktop applications. ARMOR was developed in the Microsoft .NET framework, while IRIS is a standalone Java application. Due to security concerns, both systems are run on machines that are not connected to any network. The underlying solution methods use the open source glpk[1] toolkit to solve the necessary mixed-integer programs. The general structure of the two applications is shown in Figure 5. The core architecture can be divided into three modules, which we describe in detail in the subsequent sections:

1. **Input:** Interface for the user to enter parameters and domain knowledge.
2. **Back-end:** Inputs are translated into a game model, which is passed to the Bayesian Stackelberg game solver and then to a final process that generates a specific sample schedule based on the computed probabilities.

[1] http://www.gnu.org/software/glpk/

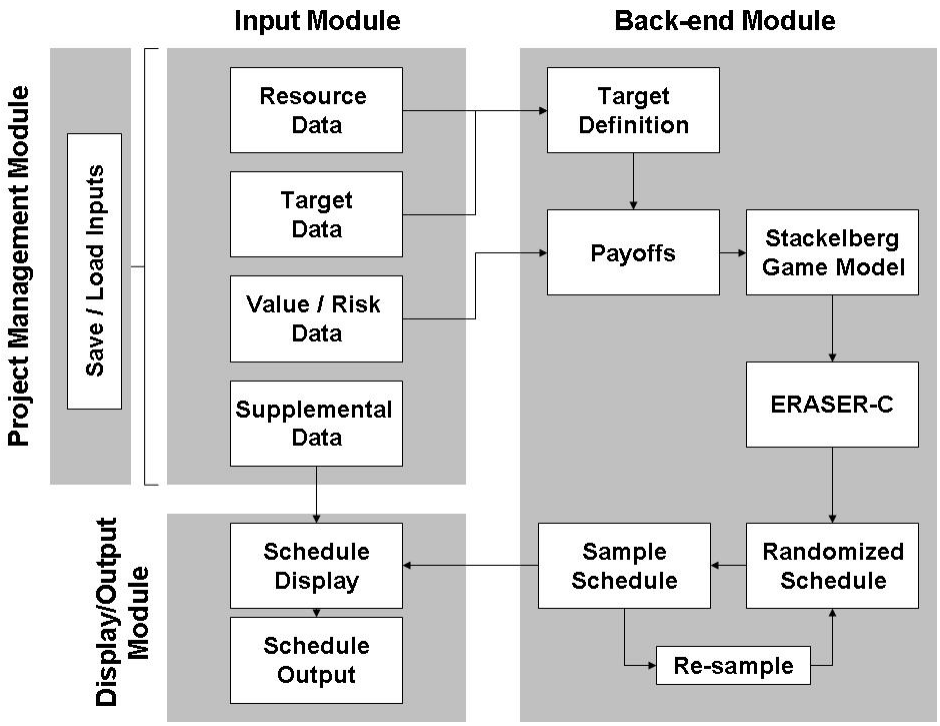


Figure 2 General Structure for the Security Assistants

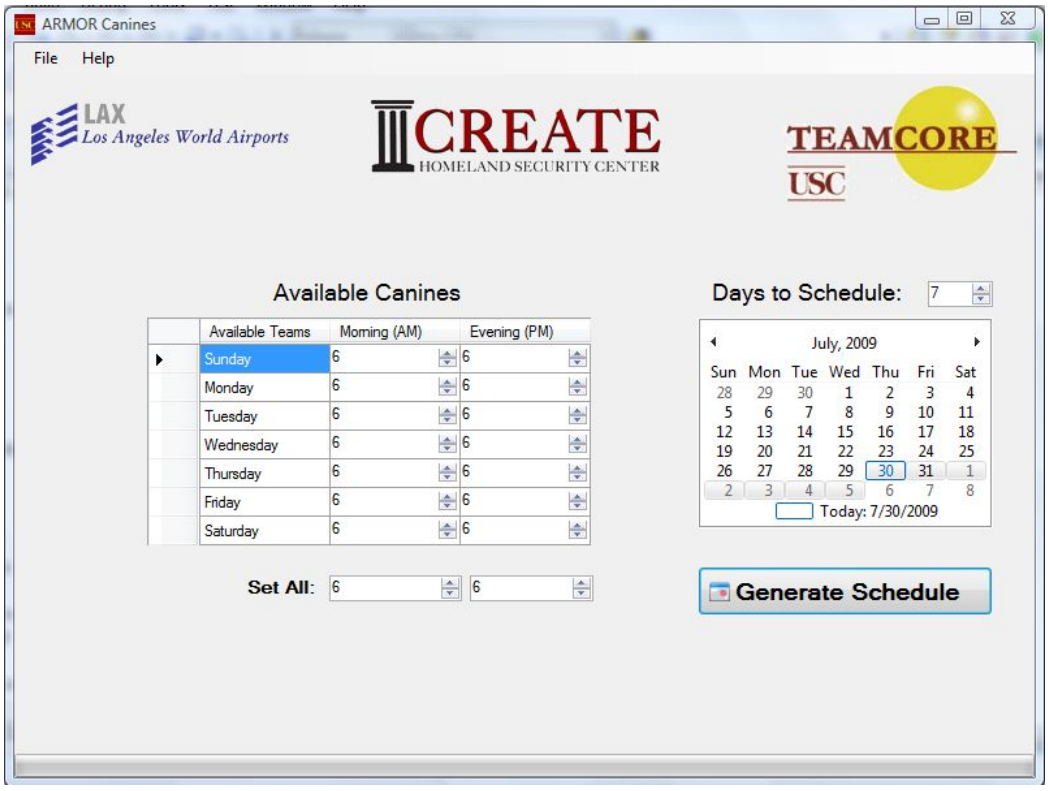
3. **Display Module:** The final schedule is presented to the user, with options to modify the output if necessary.

5.1. User Input

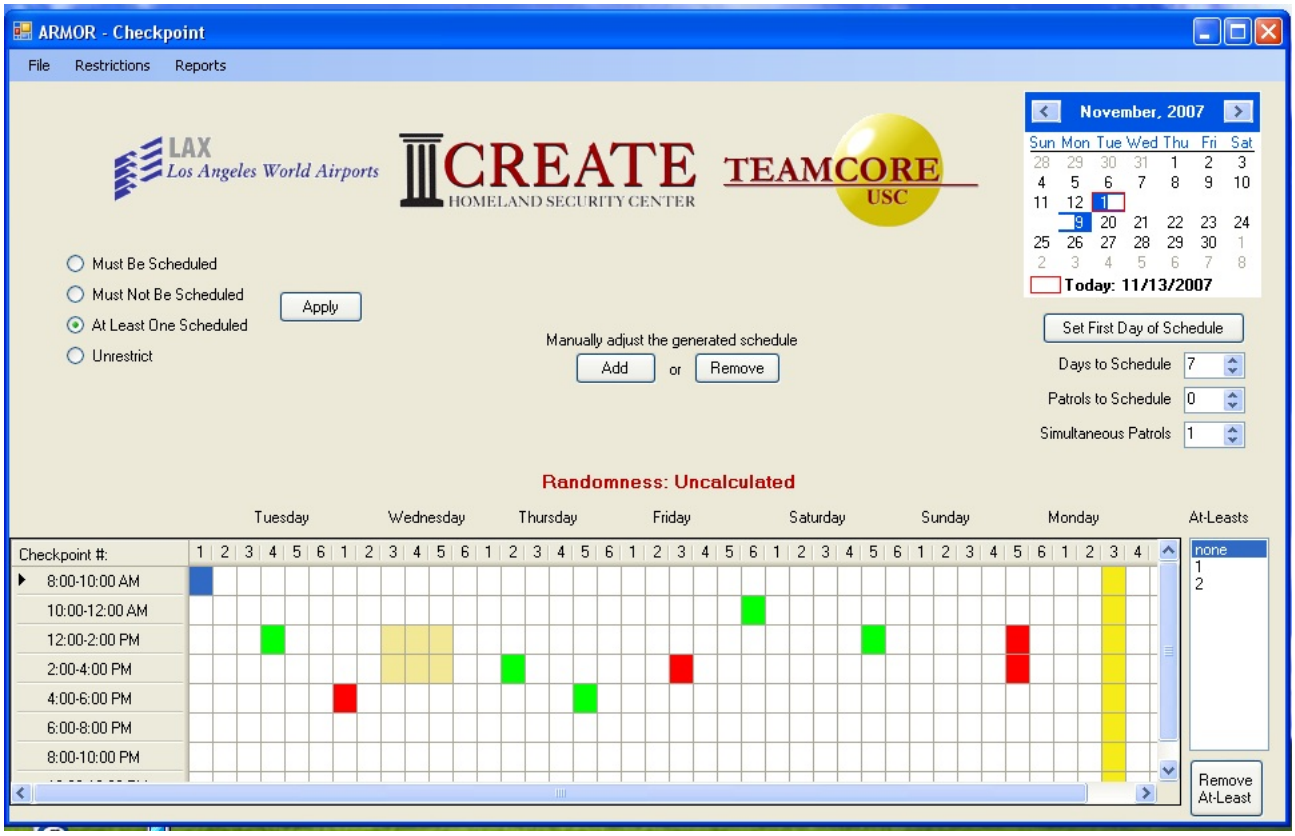
We rely on the users and domain experts to provide the knowledge required to specify the game model. While some elements of the model do not change over time, others change frequently. For these, we must provide the users a convenient way to enter the necessary values. The basic inputs that both ARMOR and IRIS require fall into four categories: (1) the number of available resources and their capabilities, (2) the set of targets, (3) payoff values for each target, and (4) supplemental data to improve the user experience (e.g., names and labels). Both applications allow users to save and re-use this information across multiple executions.

The balance of how much information is hard-coded and how much is entered by the user is quite different for ARMOR and IRIS. For example, in ARMOR the set of targets is hard-coded because the number of terminals at LAX changes very rarely. However, in IRIS the flight information may change every time the system is run, so this is part of the user input. Determining which parameters were necessary to expose to the user was a significant task, and required several iterations with the domain experts and end users to strike the right balance between the complexity of the inputs and the flexibility of the system to capture the necessary information.

5.1.1. ARMOR: The interface for the ARMOR canines program is shown in Figure 3(a). It consists of a file menu, options for varying number of available teams per day of the week, an option to change the number of days to create a schedule for, and a monthly calendar. These are the parameters that the LAX police want to edit on a daily basis. We made a special effort to keep the interface intuitive and user-friendly. The number of terminals at LAX is fixed, so this is not required as part of the input; it is hard-coded in the system. When the "Generate Schedule" button in the interface is pressed, the system takes the inputs and



(a) Canine Interface



(b) Checkpoint Interface

Figure 3 ARMOR Interface

generates the underlying game model. The model is solved, and a schedule is returned for the user to view. In the example shown, this would have 6 canines scheduled every morning/evening for 7 days.

The interface for the ARMOR checkpoints program shown in Figure 3(b) is very similar, and also provides options for the number of available resources, number of days to create a schedule for, time slots to schedule, and a monthly calendar. A spreadsheet is used to display the proposed schedule and provide additional opportunities for the end users to modify the schedules in a mixed-initiative setup. Three options are provided to alter the space of possible scheduling actions, and three others impose specific local constraints on the schedule. The three options to alter the action space are the following: (i) number of checkpoints allowed during a particular time slot; (ii) the time interval of each time slot; (iii) the number of days to schedule over. There are also three options that restrict certain actions in the generated schedule: (i) forced checkpoint; (ii) forbidden checkpoint; (iii) at least one checkpoint. These constraints are intended to be used sparingly to accommodate situations where a user, faced with exceptional circumstances and extra knowledge, wishes to influence the output of the game. The user may impose these restrictions by forcing specific actions in the schedule using the spreadsheet interface. Each restriction is represented by a different color in the spreadsheet.

ARMOR generates a different game for each time slot on each day. The number of defender resources in the model is the number of canine units/checkpoints specified by the user. The number of targets is the number of terminals for the canines system, and the number of inbound roads for the checkpoints system. Generating the game matrix also requires values for the payoffs associated with each possible target. These payoff values depend on a variety of conditions, such as passenger loads, cost of the infrastructure, publicity to the adversary, etc. Domain experts provided us with formulae to automatically generate payoff values for all possible combinations of such conditions, which we encode in ARMOR. The system is also provided with estimates of the passenger load and other elements (the details of these formulae and tools cannot be discussed due to security concerns). For any given day, ARMOR is able to take the conditions for this day and select appropriate payoff values for the targets. As a result, it is not necessary for LAX police officers to enter these values by hand to generate each schedule, which is both time-consuming and error-prone. The system still retains a high degree of flexibility because values are precomputed and stored for a wide range of possible conditions.

Intelligence information can be incorporated in ARMOR in a number of ways, depending on the nature of it. For example, if the security forces believe that there is a higher probability that an attack will be attempted on a particular target, they can increase its payoff value in the game matrix. This, in turn, would result in more security resources being deployed to protect that target. Or, if a new adversary type becomes known with a different set of values for the targets, a new attacker type can be added via the Bayesian framework. Probabilities of each adversary type can also be changed to incorporate intelligence information. Additionally, the mixed-initiative interface of ARMOR allows the security forces to manually add constraints like 'at least one unit' to incorporate additional information.

5.1.2. IRIS: The FAMS domain is considerably larger, and the information required to build a game model in this domain changes much more frequently. For these reasons the application is considerably more complex than ARMOR in terms of the user interface and the mechanisms required to input all of the necessary information. This additional complexity is necessary in this domain to accurately capture the situation, and provide all of the functionality requested by the end users. However, it does place a greater burden on the users to learn the system, and scheduling is a more time-consuming process than in ARMOR. Again, finding the right level of complexity to expose to the users was an iterative process that involved many discussions with the users and domain experts.

In the FAMS domain, we require information about the available air marshals, their scheduling constraints, the possible flights, and information about the risks/values to associate with each flight. The data about resources includes information about the number and location of air marshals, as well as the conditions that define legal flight schedules. Flight information includes various data about each flight, including flight number, carrier, origin, destination, aircraft type, etc. Finally, some information is collected to improve

usability, even though it is not strictly necessary for the game-theoretic analysis. This includes naming schemes for airports and airlines and other information that allows the system output schedules in a more usable format or to interface easily with other systems. IRIS also includes functionality to import data from existing databases with flight data and other information. This greatly reduces the amount of data entry necessary to create a schedule.

Specifying the payoff values for every possible flight was a particular challenge in this domain, since there are thousands of flights to consider. We use an attributed-based system to elicit these values, based on the Threat, Vulnerability and Consequence (TVC) model for estimating terrorism risk (50). By eliciting values for attributes of flights rather than specific flights, we are able to dramatically reduce the number of entries required by the user. Each flight is then given an aggregate value based on these components; the specific calculations used to determine flight risk are sensitive information, and cannot be revealed. The values of the attributes for each flight can be populated automatically from existing databases. To allow for specific intelligence or exceptional circumstances, the individual payoff values for any flight can also be directly edited by the end user. However, this is only rarely necessary and the majority of the analysis can be effectively automated.

This preference elicitation system of IRIS has substantially reduced the number of values that must be entered by the user. During a restricted test run on real data, the attribute-based approach called for a total of 114 values to input regardless of the number of flights. By contrast, there were 2,571 valid flights over a week, each requiring 4 payoff values, summing to 10,284 user-entered values without the attribute-based preference elicitation system. The attribute-based approach clearly requires far fewer inputs and remains constant as the number of flights increases, allowing for excellent scalability as we deal with larger and larger sets of flights. Equally importantly, attribute-based risk assessment is an intuitive and highly-scalable method that can be used in any problem where people must distill numerous attributes of a situation into a single value for a large number of situations that share the same attributes.

5.2. Game Generation and Solving

This module builds a specific instance of a Bayesian Stackelberg game, based on all of the data provided by domain experts and entered through the GUI by end users. Some of the necessary information is hard-coded in each system, while other inputs can be modified by the user during the scheduling process. The specifics (insofar as we are allowed to discuss) of how the inputs and domain knowledge are mapped into the underlying game models are described in Sections 4.2, 5.1 and the Appendix (Section 9).

Once an explicit game model has been generated it is passed as input to the ERASER-C mixed-integer program. This model can be solved using any standard solution package; we use the GLPK open source solver in these applications. ERASER-C returns an optimal mixed strategy for the defender — a probability distribution over the defender's actions — which represents a randomized policy for allocating the security resources of either LAX or FAMS. From the randomized schedule, we generate a sample schedule for the security forces. This sample schedule specifies exactly where and when each resource should be assigned to each target. If necessary, it is also possible to "re-sample" from the randomized schedule to get another specific schedule, though this capability is used rarely. The final schedules conform to the domain constraints input by the user. In particular, any specific constraints given in ARMOR, e.g. forbidden checkpoint as described in Section 5.1.1, are taken into consideration when final schedules are sampled.

5.3. Output Module

The output module presents the generated schedule to the user. The user can then review the schedule and accept it as is, or add additional constraints and run the scheduling process again.

5.3.1. ARMOR: The generated schedule of checkpoints and canines is presented to the user via a spreadsheet. Each row in the output spreadsheet corresponds to one hour. Each column in the sheet corresponds to a terminal. Each entry in the sheet represents a schedule generated by ARMOR. The familiarity of the police officers with spreadsheets helped in the acceptance of the ARMOR schedules.

If a schedule is presented to the user with any alerts (e.g. when ARMOR believes that user constraints are causing a very low likelihood checkpoint to be scheduled), the user may alter the schedule by altering the forbidden/required checkpoints, or possibly by directly altering the schedule. Both possibilities are accommodated in ARMOR. If the user simply adds or removes constraints, ARMOR can create a new schedule. Once the schedule is finalized, it can be saved for actual use, thus completing the system cycle. This full process was designed to specifically meet the requirements at LAX for checkpoint and canine allocation.

5.3.2. IRIS: The generated schedules are presented to the user via the application window. The schedule created is shown in the interface, and allows the users to view more detailed information about each target. The user is also able to output the schedule to a file which can then be used to analyze the schedule in more detail. The sample assignment of federal air marshals to flight schedules is exactly a schedule that could be used by the FAMS. At this point, the scheduling assistant allows the expert using the system to create numerous sample schedules based on the same optimal mixed strategy or simply change the assignment of federal air marshals to flight schedules by hand to create a final schedule that meets the needs of the FAMS. Of course, the user can also adjust any of the parameters entered and re-solve the game completely. The output of IRIS is in the same format as the other systems used by the FAMS officers. It has not been presented here for simplicity and because of security concerns.

5.4. Lessons Learned

The design and deployment of ARMOR and IRIS has posed numerous challenges. We outline some key lessons learned during the design and deployment of these tools. Firstly, there is a critical need for randomization in security operations. Security officials are aware that requiring humans to generate randomized schedules is unsatisfactory because as psychological studies have often shown (47, 43), humans have difficulty in randomizing, and they can also fall into predictable patterns. Instead, game-theoretic randomization that appropriately weighs the costs and benefits of different actions, and randomizes with appropriate weights leads to improved results. Security officials were therefore extremely enthusiastic in their reception of our research, and eager to apply it to their practices.

Secondly, organizational acceptance is a key issue. In creating solutions for people, we must be cognizant of how difficult it will be for a user to adopt our solution. Each deviation from existing methodology is a step away from the familiar that we must convince the user to accept. Instead of asking people to make numerous and sometimes unnecessary changes, minimizing these differences and complexities can help pave the way towards a successful implementation. For example, tweaking the GUI to achieve a look and feel that the user is familiar and comfortable with can help the user understand the system faster and better. Similarly, because infrastructural changes are often costly and/or time-consuming, ease of incorporating our work into their daily routine is essential. For example, using inputs and creating outputs that were in the same format as existing protocols minimized the additional work that our assistant would create for the security officers and lead to easier acceptance of the system.

Thirdly, it is important to provide the users with operational flexibility. When initially generating schedules for canine patrols, we created a very detailed schedule, micro-managing the patrols. This did not get as positive a reception from the officers. Instead, an abstract schedule that afforded the officers some flexibility to respond to situations on the ground was better received.

6. Evaluation

Evaluating the impact of the software assistants developed in this work on security systems is not simple. It is difficult to quantify the ‘bottom line’ in security applications, where safety usually trumps costs. There are also security concerns in making evaluations of security policies publicly available and ethical concerns in not providing the best security possible to a control group. Taylor et al. (40, 41) discuss the difficulties in such evaluations in greater detail.

To illustrate these difficulties, consider an evaluation of a security system through a long term study that requires prolonged periods, of six months to a year, of alternatively using the previous methods and our game-theoretic methods to create schedules. During this time, we would need all other variables that are relevant to security (e.g., number and behavior of adversaries, economic conditions, geopolitics, number of travelers, etc.) to remain constant. Additionally, we assume that the adversaries actions are influenced by the security measures they observe over a period of time. Having the security strategy change during this observation would generate different actions from the adversaries than would be observed in an actual deployment. For these reasons, we have not conducted such a long term study of the proposed security strategies in the real world.

Instead, we provide a multi-faceted evaluation that combines evidence from as many sources as possible to evaluate the system, including expert evaluations, data from the deployment at LAX, and data from various computer simulations. Perhaps, the most conclusive evidence of the merit of our software assistants is the adoption and continued use of these tools by the security forces at both LAX and FAMS. ARMOR has been deployed by the LAX police since August 2007 and IRIS began to be used by FAMS in October 2009 after undergoing a classified internal evaluation. We describe some of the reasons for their adoption as well as the reviews of domain experts on our systems in Subsection 6.1 below. In Subsection 6.1, we present arrest record data and feedback from domain experts from the actual use of ARMOR at LAX. We use simulations to evaluate various features of ARMOR and IRIS, using data inspired by the real applications but modified somewhat for security reasons. These experiments are described in Subsection 6.2, including comparisons of the game-theoretic schedules with both a uniform random benchmark and benchmarks reflective of previous scheduling practices at LAX.

The evaluation presented here does not include all of the analysis that has been done of the ARMOR system. One additional study of particular interest that is not presented here tests the scheduling methods used by ARMOR against human adversaries in a controlled laboratory setting. The results are reported in Pita et al. (34), and generally show that ARMOR's schedules perform very well against human adversaries (though there does appear to be potential for new methods that exploit certain predictable patterns in the responses of adversaries).

6.1. Adoption and Domain Expert Evaluation

The Federal Air Marshals Service conducted its own quantitative and qualitative internal evaluation of IRIS, and found that it met their requirements. While FAMS would not disclose any details of this evaluation, they have informed us that due to this evaluation they have begun scheduling federal air marshals in flights in a pilot phase. This is our only evidence of the recent adoption and use of IRIS by FAMS; in the case of ARMOR it has been adopted and in use for over two years now. Hence, in this section, we present a more extensive evaluation of ARMOR.

We discuss three sources of evidence that speak to the value that ARMOR provides for LAX in the real-world deployment. First, we describe and analyze the arrest records from vehicle checkpoints that are available from before and after the deployment of ARMOR. Second, we present a qualitative evaluation of ARMOR based on the response of the domain experts and the media. Finally, we give a comparative evaluation of the qualitative benefits of ARMOR, for example, ease of use and the man-hours needed to generate a schedule.

Arrest Data: We received summarized and actual reports from the LAX police regarding the number of violations that they detected at checkpoints in 2007, 2008 and January 2009. For example, we received the following report for January 2009:

1. January 3, 2009: Loaded 9mm pistol discovered
2. January 3, 2009: Loaded 9mm handgun discovered (no arrest)
3. January 9, 2009: 16 handguns, 4 rifles, 1 pistol, and 1 assault rifle discovered some fully loaded
4. January 10, 2009: 2 unloaded shotguns discovered (no arrest)
5. January 12, 2009: Loaded 22cal rifle discovered

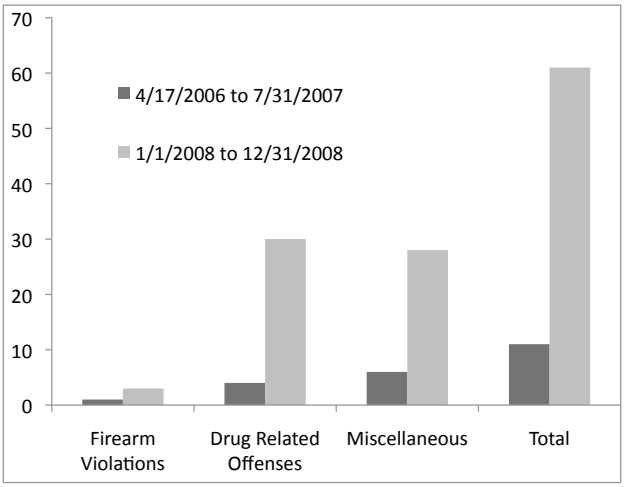


Figure 4 Number of Violations Detected at LAX Airport Checkpoints

6. January 17, 2009: Loaded 9mm pistol discovered
7. January 22, 2009: Unloaded 9mm pistol discovered (no arrest)

Figure 4 tabulates the number of violations for the year prior to the deployment of ARMOR and during 2008 when ARMOR was in use. The x-axis breaks down the violations into different types and the y-axis represents the number of violations. The number of violations is substantially higher at LAX after ARMOR was deployed than in the preceding period. For example, only 4 drug related offenses were detected before the deployment of ARMOR while 30 such offenses were detected after the deployment. While we must be careful about drawing too many conclusion from this data due to the large number of uncontrolled variables (for example, the number of checkpoints was not consistent during this period), the ARMOR checkpoints do appear to be quite effective.

Response of Domain Experts: As domain experts inform us, no security system – ARMOR and IRIS included — can provide 100% security, but it can aid security forces in making the best possible use of resources at their disposal. Qualitative internal and external security reviews indicate that ARMOR is both effective and highly visible. Obviously, without a positive assessment from domain experts, ARMOR would not be in continual use at LAX. Protecting what is sometimes referred to as a top terrorist target on the west coast of the US is a critical task and experts would disallow an ineffective scheduler. Director James Butts, LAX Police, reports that ARMOR "makes travelers safer" and even gives them "a greater feeling of police presence" (27). Erroll Southers, Assistant Chief of LAX Airport Police, told a Congressional Committee hearing that "LAX is safer today than it was eighteen months ago," due in part to ARMOR (3). A recent external study by international transportation security experts concluded that ARMOR was a key component of the LAX defensive setup. The ARMOR team has also been awarded Letters of Commendation from the city of Los Angeles in recognition of the efforts towards securing the Los Angeles International Airport (5). Thus, the domain experts have been highly supportive of ARMOR, and it would be very hard to deploy the system without their support. They are also likely to identify potential problems with the system quickly. While such studies are not very useful for quantifying ARMOR's benefit, they all suggest that the domain experts believe that ARMOR generates better schedules as compared to their previous approaches.

Ease of use: ARMOR has been instrumental in aiding the police forces to efficiently and more conveniently generate schedules to deploy more and more units. For example, consider a situation when only 2 canines need to be scheduled for 2 hours each over any of the 7 terminals. Each canine could be assigned to any of the 7 terminals each hour, making the search space as large as $7^4 (= 2401)$ combinations. This search space grows exponentially with the number of canines and the number of hours for which the schedule needs to be generated, making it impractical for human schedulers. Thus, ARMOR has played a significant role in reducing, if not completely eliminating, the loads on the officers to manually construct patrolling schedules.

Additionally, the use of ARMOR has also made it possible for the security officers to update the generated schedules, in case more resources become available or new constraints need to be incorporated. Furthermore, even though ARMOR was designed as a mixed initiative system, users have chosen not to modify ARMOR schedules in practice, which suggests that the output schedules are indeed high-quality, and that domain experts have not chosen to 'tweak' the system's decisions. These added benefits have themselves been a contributing factor towards the continued use of schedules generated by ARMOR.

6.2. Simulation Results

To better quantify the benefits of the game-theoretic schedules produced by ARMOR/IRIS, we turn to simulation results in more controlled settings. In Section 6.2.1, we compare the achieved quality (or the defender reward) of our strategy against uniformly random and other strategies reflective of the strategies used by the human schedulers. We also present runtime results in Section 6.2.2 comparing the runtime of ERASER-C with DOBSS (30) and Multiple-LPs approaches (17), the previous fastest solvers for Stackelberg games, and show that ERASER-C is significantly faster than both of them.

6.2.1. Quality Comparisons: We present three sets of simulation results comparing the solution quality of our approach with other approaches. First, we compare the randomized schedules produced by ARMOR/IRIS against a benchmark uniform random policy. We then describe a benchmark policy based on the previous scheduling procedures used by LAX police and compare against this benchmark. Finally, we compare solution quality of our randomized schedule with uniformly random schedules in the Bayesian case.

Comparison with Uniform Random Policy:

In evaluating our method, we compare the schedules generated using our methods against a uniform random policy, the most obvious alternative for randomization. In the uniform random policy, each defender action is covered with equal probability regardless of payoff. Solution quality is measured by calculating the highest expected payoff attainable by the defender, under the Strong Stackelberg Equilibrium assumption that the attacker chooses an optimal response and breaks ties in favor of the defender. The experiments were run using data from the deployment of canines from the LAX domain. We show the differences in defender reward obtained by scheduling canines units at LAX using ERASER-C and scheduling them with a uniform random strategy in Figure 5. The simulations used the actual game models for the ARMOR canines problems. In the uniform random strategy, canines are randomly assigned to terminals with equal probability. The x-axis represents games with different payoffs and the y-axis represents the reward obtained. We can see that ERASER-C performs better even with 3 canine units as compared to 6 canine units being scheduled using the uniform random strategy. For example, the reward of a uniformly random strategy with 6 canine units is -1.47 on average across the four games whereas the reward of 3, 5 and 6 canines with ERASER-C is 1.11, 3.38 and 4.41 respectively. These results show that ERASER-C randomization with even fewer resources provides better results as compared to uniformly random strategies in the same domain with more resources.

Similarly, we ran experiments using data from the FAMS domain. We used one week of real flight data for subregions of Region A and three separate sets of hypothetical FAMS home city data that vary the number of federal air marshals available. Hypothetical payoffs were used. Region A and the countries within it are actual places for which we have used real flight data, but due to security concerns we have anonymized them here. Region A is composed of five larger countries which we have designated 1-5 as well as a few small countries which are only included in the full region tests. Data about the size of each region is presented in Table 2. We created random values for the other inputs and held them constant throughout the experiments.

In Figure 6, the y-axis represents the normalized defender reward for both strategies, with all payoffs normalized to the maximum expected defender's payoff achievable under the strategy generated by our method. Across the x-axis, we show different regions such that there is an increase in the number of flights and flight schedules from left to right. Thus, the rightmost group of bars, from left to right, represents the maximum expected defender's payoff achievable under ERASER-C as 100% and under the uniform

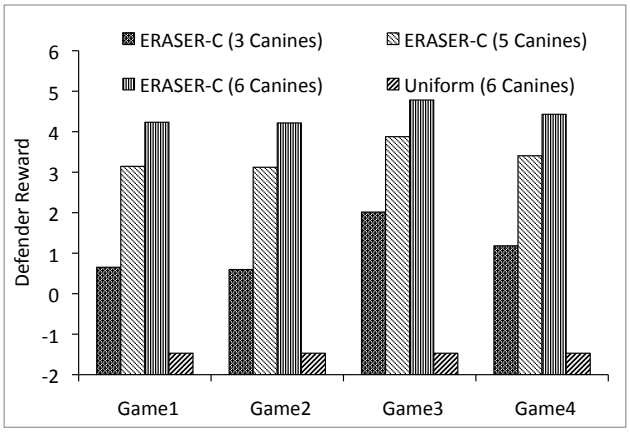


Figure 5 Comparison of ARMOR solution quality and uniform random solution quality for the canines problem.

randomization strategy as 19% worse than ERASER-C. ERASER-C's solution is superior to uniformly random in every region tested.

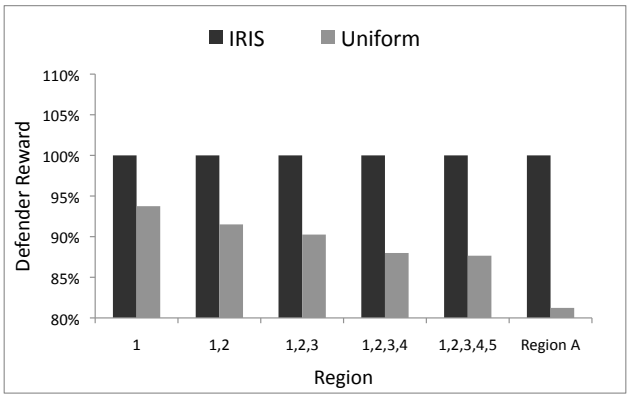


Figure 6 Comparison of IRIS solution quality and uniform random solution quality.

Region	Flights	Flight Schedules
1	873	1,181
1,2	1,147	1,403
1–3	1,528	1,660
1–4	1,845	1,975
1–5	2,033	2,114
Region A	2,571	2,416

Table 2 Sizes of different regions for FAMS experiments.

Comparisons with Previous Scheduling Methods:

The LAX officers informed us that they previously generated checkpoint schedules based on a cyclic strategy with random perturbations. A study of past schedules showed that patterns remained evident despite these random perturbations – no checkpoint was repeated for two consecutive days. Therefore, we also compared ERASER-C strategy against two strategies: (1) a 'cyclic' strategy where the checkpoints were

scheduled in a cyclic order on all inbound roads and, (2) a 'restricted uniform' strategy which was a uniformly random strategy with the additional restriction that no checkpoint was repeated on two consecutive days.

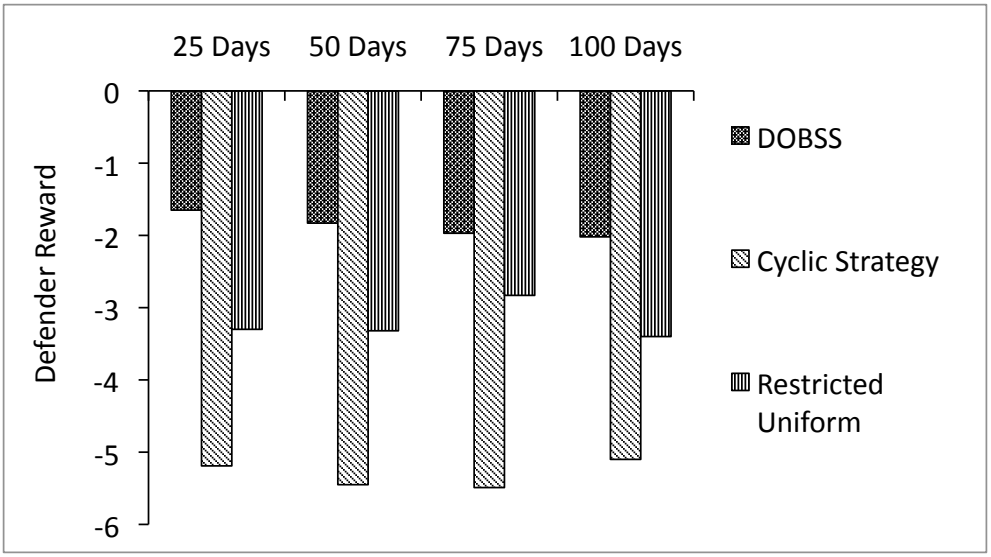


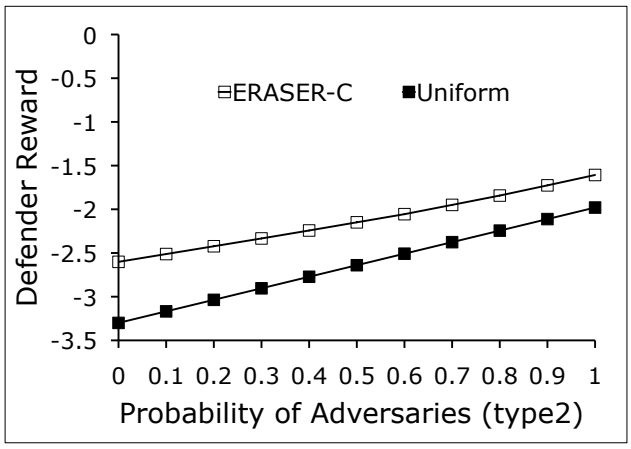
Figure 7 Comparison of ARMOR strategies and policies representative of previous methods.

We perform our experiments in the same setup used by LAX police in ARMOR – police officers place one checkpoint on any of the five inbound roads, and the rewards of the terminals were randomly chosen between 1 and 10. We vary the duration for which the adversary can make observations from 0 to 100 days, in increments of 25 days. We averaged our results over 100 trials. In these simulations, we assume that the adversary can use simple pattern recognition techniques to recognize patterns in the checkpoint schedules. In detail, the adversary maintains a history of observed checkpoints and generates confidence intervals over sequences of observations.

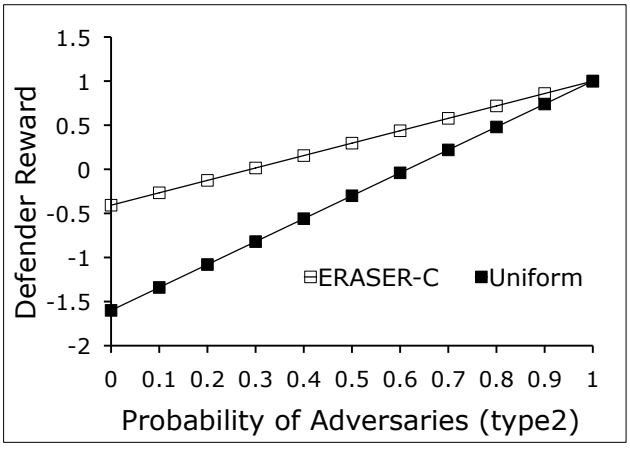
Figure 7 shows our experimental results. The x-axis represents the number of observations available to the adversary, and the y-axis represents the average defender reward. (In comparison with Figure 5, the expected defender reward is mostly negative, because here we focus on a single checkpoint consistent with previous scheduling approaches as opposed to multiple canines used for scheduling in Figure 5.) The ERASER-C strategy has a higher average defender reward compared to the other two strategies. The reason is that the adversary can better predict the defender action in case of 'cyclic' and 'restricted uniform' strategies as compared to the ERASER-C strategy. Therefore, simple pattern recognition techniques are sufficient for the adversary to exploit the patterns in the cyclic and restricted uniform strategies. These patterns can be avoided by the use of uniformly random strategies, but a uniform random strategy does not take into account the different preferences over targets of the defender. ARMOR provides weights for the different targets such that the average defender reward is the highest when compared against both 'cyclic' and 'restricted uniform' strategies. Additionally, ARMOR strategies are not just weighted random since they also account for the fact that the adversary observes the defender's strategy and then makes an informed rational choice.

Solution Quality in the Bayesian Case:

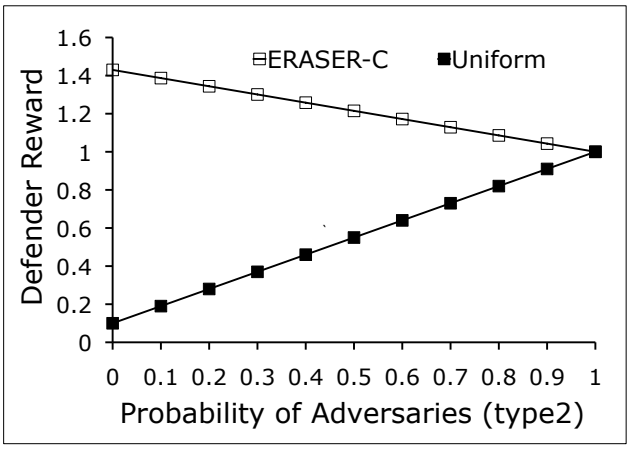
We also tested the performance of our algorithms in the Bayesian case, i.e., when multiple security resources were faced with multiple adversary types. The results of these experiments are shown in Figures 8(a), 8(b) and 8(c). These experiments were conducted for a problem from the LAX domain. In these experiments, we considered a scenario where 10 targets had to be defended against 2 types of adversaries by 1, 2 or 3 security units. The x-axis represents the probabilities of occurrence for type 1 and type 2



(a) 1 Checkpoint



(b) 2 Checkpoints



(c) 3 Checkpoints

Figure 8 Solution Quality in the Bayesian Case

adversaries. The x-axis shows the probability p of adversary type 2 (the probability of adversary type 1 is then obtained as $1-p$). The y-axis represents the reward obtained by the defender. Figure 8(a) shows the comparison when one resource (checkpoint) is placed. For example, when adversary of type 1 occurs with a probability of 0.1 and type 2 occurs with a probability of 0.9, the reward obtained by the ERASER-C strategy is -1.72 whereas the reward obtained by a uniform random strategy is -2.112. It is important to note that the reward of the ERASER-C strategy is strictly greater than the reward of the uniform random strategy for all probabilities of occurrence of the adversaries.

Figure 8(b) also has the probability distribution on the x-axis and the reward obtained on the y-axis. It shows the difference in the obtained reward when 2 resources (checkpoints) are available. Here also the reward in the case of the ERASER-C strategy is greater than the reward of the uniform random strategy. When we have 2 resources, the type 2 adversary chooses the action $none$ (to not attack). This leads to the observation that the rewards of the ERASER-C strategy and the reward of the uniform strategy are the same when only the type 2 adversary is present. Figure 8(c) presents the case of 3 resources (checkpoints). Here the reward values obtained by ERASER-C in the 3 resource case are always positive — this is because the chances of catching the adversary of type 1 improve significantly with 3 checkpoints. This also leads to the reward of ERASER-C decreasing with the decrease in the probability of occurrence of the adversary of type 1. Note that the type 2 adversary, as with the case of 2 resources, decides $none$ and hence the reward of the ERASER-C strategy and the uniformly random strategy are the same when only type 2 adversary is present.

6.2.2. Runtime Comparisons: For ARMOR and IRIS to be practically useful, it was necessary to develop solution algorithms that could solve very large problem instances in a reasonable amount of time. To demonstrate the scalability of our algorithms, we conducted experiments where we scaled three different parameters of the domain: the number of targets, the number of available resources and the number of adversaries. In all cases, ERASER-C was more scalable than the fastest previous methods, DOBSS (30) and Multiple-LPs (17). We also show results for a large realistic game from the FAMS domain with thousands of targets and hundreds of resources, showing that these games can be solved in less than fifteen minutes.

Scaling Number of Targets:

We first present results increasing the number of targets from 25 to 200, fixing both the number of resources and adversary types to 1. The results are shown in Figure 9. The x-axis on the figure shows the number of targets, whereas the y-axis shows the runtime in seconds on a logarithmic scale. For example, when there are 100 targets, the runtime of ERASER-C is 0.673 seconds, the runtime of DOBSS is 13.84 seconds and the runtime of Multiple-LPs is 4.89 seconds. The results are averaged over 30 trials with different randomly-generate game matrices. ERASER-C is faster than both other algorithms by a considerable margin. We also note that when the number of resources is also scaled with the number of targets, this effect is even more dramatic (22).

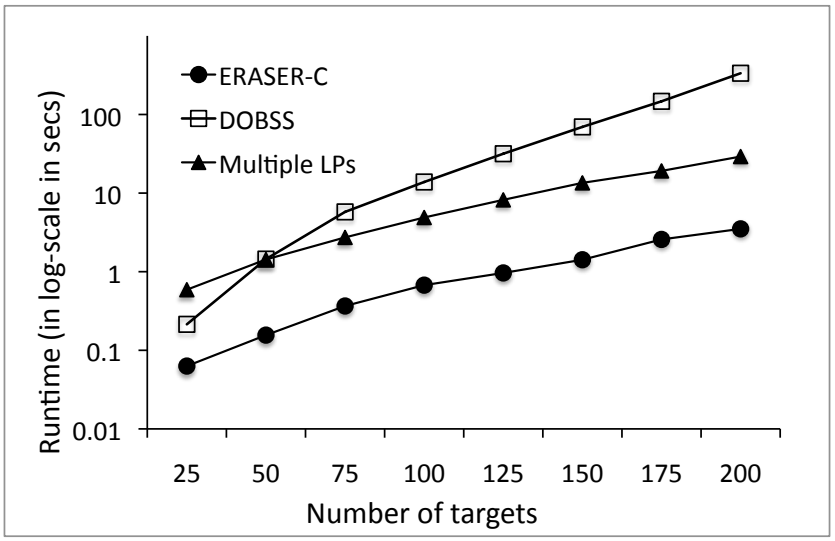


Figure 9 Comparison of runtime when number of targets is increased.

Scaling Number of Resources:

In this experiment, we vary number of resources from 1 to 7, using 15 targets and 1 adversary type. The results are shown in Figure 10. The x-axis on the figure shows the number of resources, and the y-axis shows the runtime in seconds on a logarithmic scale. For example, when there are 5 resources, the runtime of ERASER-C is 0.044 seconds, the runtime of DOBSS is 11.74 seconds and the runtime of Multiple-LPs is 3.06 seconds. The results are averaged over 30 trials with different randomly-generate game matrices. ERASER-C is must faster than the other two algorithms in this case. The effect of the combinatorial set of possible assignments that is enumerated by both DOBSS and Multiple-LPs is also clear in this plot. While ERASER-C solution times are constant on the log scale as the number of resources increase, the other two algorithms show substantial increases even on the log scale.

Scaling Number of Adversaries:

The number of adversaries was increased from 1 to 5 in this experiment (see Figure 11), with 10 targets and 1 resource. The x-axis in the figure shows the number of adversaries, and the y-axis shows the runtime in seconds on a logarithmic scale. For example, when the number of adversaries is 4, ERASER-C took

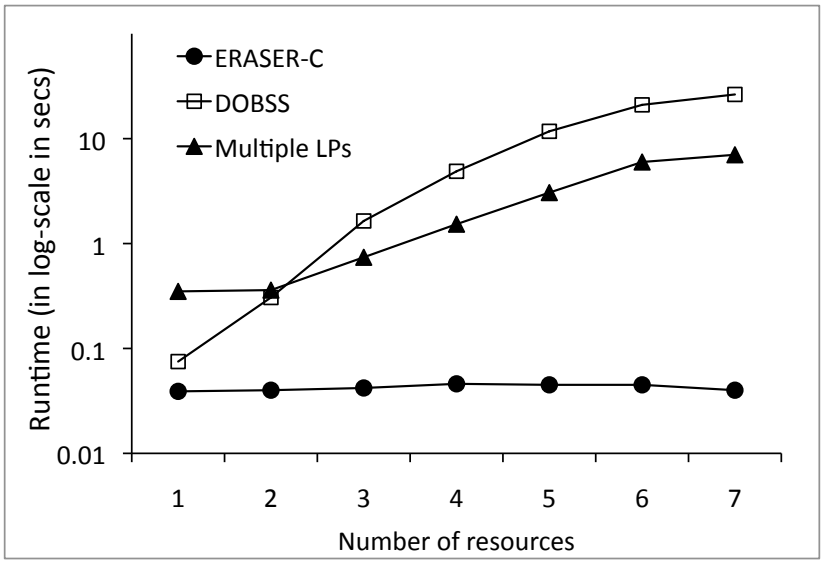


Figure 10 Comparison of runtime when number of resources is increased.

0.244 secs, DOBSS took 0.524 secs and Multiple-LPs was the slowest taking 5138.0 seconds. All results are averaged over 30 trials. ERASER-C is the fastest in this case as well, though DOBSS scales similarly in this case with a single resource. The Multiple-LPs method uses the Harsanyi transformation, resulting in much slower solution times as we increase the number of adversary types.

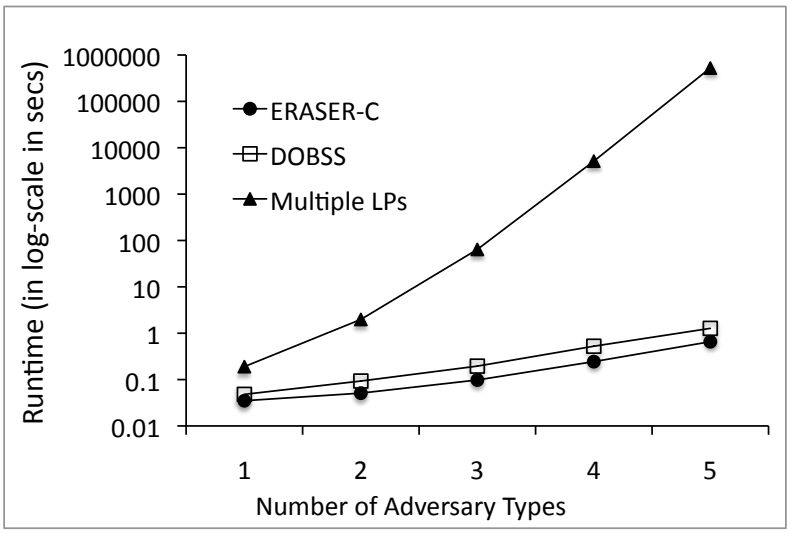


Figure 11 Comparison of runtime when number of adversary types is increased

Scaling to Large Games:

Finally, we present results on a realistic data set for the FAMS domain to show that ERASER-C is capable of solving these games in a reasonable amount of time. While the LAX domain has only a few terminals and checkpoints, FAMS has tens of thousands of flights and hundreds of air marshals. Here we show the runtime for ERASER-C to generate a schedule for different numbers of possible defense resources. The targets are defined using real flight data from several (unnamed) regions of different sizes, using all flights between the United States and the given region for a one-week period. Results are shown in Figure 12.

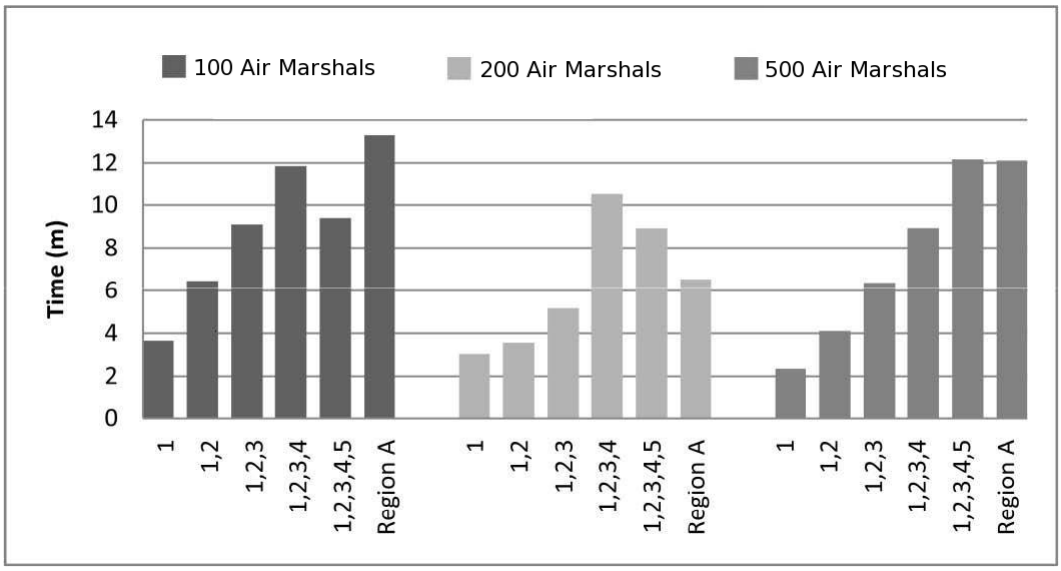


Figure 12 Runtimes results for realistic FAMS problems.

In Figure 12, the y-axis represents the time required for generating a schedule in minutes and the x-axis shows the region from which the flight data was considered. The game sizes and the region information used in this experiment is the same as in Figure 6 and is presented in Table 2. We tested with 100, 200 and 500 available air marshals, shown on the x-axis in Figure 12 in three separate sets respectively, since the actual number of federal air marshals is security sensitive information unavailable to us. For example, the first bar on the left represents an average runtime of 3.65 minutes to create a schedule for all flights to Country 1 within a one week period over 20 trials. In light grey, the first bar on the left represents the exact same experiment, but run with a medium number of federal air marshals. All of the games instances completed in under 15 minutes — well within the time guidance given by the FAMS for these problems.

7. Summary

Monitoring and patrolling are key components of law enforcement in security domains. In generating schedules for these patrols, it is important to account for varying weights of the targets being protected as well as the fact that potential attackers can often observe the procedures being used. This paper describes scheduling assistants for the LAX police, ARMOR, and the FAMS, IRIS, which provide game-theoretic solutions to this problem. The two systems assist the security forces in generating randomized patrols while ensuring that differences in importance of different targets are preserved.

ARMOR and IRIS make use of algorithmic advances in multi-agent systems research to solve the class of massive security games with complex constraints that were not previously solvable in realistic time-frames. Thus, although our applications were designed to be deployed at LAX and FAMS, they provide a general framework for solving patrolling scheduling problems in other domains as well.

Another domain in which our game-theoretic framework can be directly applied is the randomization of security procedures conducted by the Transportation Security Administration (TSA). TSA directs individual airports in the daily screening of passengers and bags on local flights. The agency also handles customs and border protection and conducts random employee screenings. The game theoretic model discussed here offers TSA an intelligent method of selecting and scheduling these security procedures. Other examples of potential domains include patrolling at ports and subway systems.

This research and these applications have been effective in helping in the security officers with scheduling and patrolling concerns. Thus, ARMOR and IRIS represent successful transitions of game theoretic advances to applications that have been in use and effective in the real-world.

8. Acknowledgements
The development of ARMOR and IRIS has only been so successful due to the exceptional collaboration with the Los Angeles Airport Police and the United States Federal Air Marshal Service. This research was supported by the United States Department of Homeland Security through the Center for Risk and Economic Analysis of Terrorism Events (CREATE) under grant number 2007- ST-061-000001. However, any opinions, findings, and conclusions or recommendations in this document are those of the authors and do not necessarily reflect views of the United States Department of Homeland Security.

9. Appendix
9.1. Solution Method
We introduce the ERASER-C algorithm (Efficient Randomized Allocation of SEcurity Resources with Constraints), which takes as input a security game in the compact form described in Section 3.2 and solves for an optimal coverage vector corresponding to a SSE strategy for the defender. We allow resources to be assigned to *schedules* covering multiple targets. The set of legal schedules $S = \{s_1 \dots s_l\}$ is a subset of the power set of the targets, with restrictions on this set representing scheduling constraints. We define the relationship between targets and schedules with the function $M : S \times T \rightarrow \{0, 1\}$, which evaluates to 1 if and only if t is covered in s. The defender's strategy is now an assignment of resources to schedules, rather than targets. Another important notion is the presence of *resource types*, $\Omega = \{\omega_1, \dots, \omega_v\}$, each with the capability to cover a different subset of S. The number of available resources of each type is given by the function $\mathcal{R}(\omega)$. Coverage capabilities for each type are given by the function $Ca : S \times \Omega \rightarrow \{0, 1\}$, which is 1 if the type is able to cover the given schedule and 0 otherwise.[2]

The combination of schedules and resource types captures key elements of the security domains. For example, in FAMS, federal air marshals are resources, and flights are potential targets, with payoff values defined by risk analysis of the flight. However, a marshal cannot be on all possible flights due to location and timing constraints, for example, a marshal in New York cannot board flights flying out of Los Angeles. Legal schedules can be used to define the set of possible flights that a federal air marshal could feasibly fly, given these constraints. Resource types are use to define the initial state (notably, location) of a marshal, which defines a subset of legal schedules that any given marshal could fly.

The effect of adding scheduling and resource coverage constraints is to constrain the space of feasible coverage vectors. Consider an example with a single federal air marshal defending three flights. Suppose that there are two legal schedules, covering targets $\{1, 2\}$ and $\{2, 3\}$. Given only these schedules, it is not possible to implement a coverage vector that places 50% probability on both targets 1 and 3, with no coverage of target 2.

The algorithm is a mixed-integer linear program (MILP), presented in Equations 4–15. The notation for the different symbols used the MILP are presented in Table 3. Equations 5 and 9 force the attack vector to choose a single target with probability 1. Equation 6 restricts the coverage vector to probabilities in the range $[0, 1]$, and Equation 12 constraints the coverage by the number of available resources. The coverage of each schedule must sum to the contributions of the individual resource types, specified in Equation 10. The mapping between the coverage of schedules and coverage of targets is enforced in Equation 11. Equation 12 restricts the schedule so that only the available number of resources of each type are used. No probability may be assigned to disallowed schedules for each resource type, which is explicitly enforced by Equation 13. Equation 14 defines the defender's expected payoff, contingent on the target attacked in A^γ where γ is the follower type. Since the objective maximizes d^γ, for any optimal solution $d^\gamma = U^\gamma_\Theta(C, A^\gamma)$. This also implies that C is maximal, given A^γ for any optimal solution, since d^γ is maximized. In a similar way, Equation 15 forces the attacker to select a strategy in the attack set of C. If the attack vector specifies a target that is not maximal, this constraint is violated. Therefore, taken together, the objective and Equations 14–15

[2] Our current implementation uses complete matrices to represent M and Ca, but sparse representations could offer additional performance improvements.

imply that C and A^γ are mutual best-responses for the defender and the adversary in any solution. Thus, the defender mixed strategy C and the adversary attack vector A^γ for each adversary type γ form a strong Stackelberg equilibrium of the security Stackelberg game.

Symbol	Meaning
d^γ	Reward of defender against adversary of type γ
k^γ	Reward of adversary type γ
p^γ	Probability of occurrence of adversary of type γ
Γ	Set of adversary types
T	Set of targets
A^γ	Attack vector for the adversary of type γ
a_t^γ	Probability of adversary of type γ attacking target t
C	Coverage vector of the defender
c_t	Probability of defender covering target t
$h(s,\omega)$	Probability of coverage of schedule s by defender type ω
q_s	Total coverage probability over schedule s
S	Set of valid schedules
Ω	Set of resource types
$Ca(s,\omega)$	Capability: 1 if type ω can cover schedule s; 0 otherwise
$\mathcal{R}(\omega)$	Number of available resources of type ω
$M(s,t)$	Mapping: 1 if schedule s covers target t; 0 otherwise
Z	Huge positive constant
$U_\Theta^\gamma(t,C)$	Utility of the defender when facing adversary type γ who attacks target t when defender coverage is C
$U_\Psi^\gamma(t,C)$	Utility of the adversary of type γ when target t is attacked and defender coverage is C

Table 3 Description of symbols

$$\max_{a,c,q,h,d,k} \sum_{\gamma\in\Gamma} d^\gamma p^\gamma \tag{4}$$
$$a_t^\gamma \in \{0,1\} \quad \forall t\in T, \gamma\in\Gamma \tag{5}$$
$$c_t \in [0,1] \quad \forall t\in T \tag{6}$$
$$q_s \in [0,1] \quad \forall s\in S \tag{7}$$
$$h_{s,\omega} \in [0,1] \quad \forall s,\omega\in S\times\Omega \tag{8}$$
$$\sum_{t\in T} a_t^\gamma = 1 \quad \forall \gamma\in\Gamma \tag{9}$$
$$\sum_{\omega\in\Omega} h_{s,\omega} = q_s \quad \forall s\in S \tag{10}$$
$$\sum_{s\in S} q_s M(s,t) = c_t \quad \forall t\in T \tag{11}$$
$$\sum_{s\in S} h_{s,\omega} Ca(s,\omega) \le \mathcal{R}(\omega) \quad \forall \omega\in\Omega \tag{12}$$
$$h_{s,\omega} \le Ca(s,\omega) \quad \forall s,\omega\in S\times\Omega \tag{13}$$
$$d^\gamma - U_\Theta^\gamma(t,C) \le (1-a_t^\gamma)\cdot Z \quad \forall t\in T,\gamma\in\Gamma \tag{14}$$
$$0 \le k^\gamma - U_\Psi^\gamma(t,C) \le (1-a_t^\gamma)\cdot Z \quad \forall t\in T,\gamma\in\Gamma \tag{15}$$

The payoff values $U_\Theta^\gamma(t,C)$ and $U_\Psi^\gamma(t,C)$ are calculated based on Equations 1 and 2. The values of $U_\theta^{\gamma,c}$ and $U_\theta^{\gamma,u}$ used in these equations are the payoff values to the defender when a target is covered and uncovered respectively. These values are provided by the domain experts, as described in Section 5. Similarly, the payoff values for the adversaries are also provided by the domain experts.

The values of other model parameters are calculated based on the user input and the game specification. Police officers and canines are the resources for ARMOR for checkpoint and ARMOR for canine respectively. ARMOR does not differentiate between different resources (for example, all canines are assumed to be equally capable), and hence there is exactly one resource type Ω. The number of resources $\mathcal{R}$, i.e. checkpoints or canines, is directly input by the user in the system. In case of ARMOR, the set of legal schedules is an assignment of a checkpoint to an inbound road, and is automatically generated by the system since ARMOR is aware of the road map of the airport. The capability matrix Ca in ARMOR consists of all ones since any resource could be assigned to any target. For example, any canine could be scheduled to any terminal.

Similarly, all the model parameters are defined based on user input and domain constraints in IRIS. The federal air marshals are the resources for IRIS. In IRIS, the different FAM Offices form the different resource types. This information has already been supplied to IRIS by the domain experts. The numbers of resources of each type $\mathcal{R}$, that is the number of federal air marshals in each office, is directly input in IRIS by the end users. The set of legal schedules S is provided as an input to the system by the FAMS in IRIS. Each schedule in IRIS is a sequence of flights that a federal air marshal can take to complete a tour. In IRIS, the capability matrix Ca is defined based on resource types; for example, federal air marshals at the FAM office based in Los Angeles can only cover schedules flying out of Los Angeles, and hence only those schedules would have their capabilities set to 1. The mapping M is also calculated by the systems based on the domain specifications. For example, in IRIS, if schedule s is to take flight f1 followed by flight f2, then the row in M corresponding to s would have ones only for columns corresponding to f1 and f2.

Kiekintveld et al. (22) have shown that the ERASER-C MILP corresponds to a SSE of the security game. The intuition behind the proof are two claims: (1) the coverage probability of the leader and the attack set of the follower are mutual best-responses by the construction of the MILP, and (2) the coverage probability of the leader gives the leader the optimal utility.

References

[1]General Description: Just the Facts. 2007. `http://www.lawa.org/lax/justTheFact.cfm`.

[2]Federal Air Marshal Service. 2008. `http://en.wikipedia.org/wiki/Federal_Air_Marshal_Service`.

[3]The Resilient Homeland - Broadening the Homelands Security Strategy. Congressional Committee Hearing on Homeland Security, May 06, 2008. `http://homeland.house.gov/Hearings/index.asp?ID=134`.

[4]TSA: Federal Air Marshals. 2008. `http://www.tsa.gov/lawenforcement/programs/fams.shtm`.

[5]LAXPD Bestows LA Honors on Viterbi School Security System Builders, 2009. `http://viterbi.usc.edu/news/news/2009/laxpd-honors-viterbi.htm`.

[6]N. Agmon, V. Sadov, G. A. Kaminka, and S. Kraus. The Impact of Adversarial Knowledge on Adversarial Planning in Perimeter Patrol. In *AAMAS*, volume 1, pages 55–62, 2008.

[7]R. Avenhaus, B. von Stengel, and S. Zamir. Inspection Games. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory*, volume 3, chapter 51, pages 1947–1987. North-Holland, Amsterdam, 2002.

[8]L. Babu, L. Lin, and R. Batta. Passenger Grouping Under Constant Threat Probability in an Airport Security System. In *European Journal of Operational Research*, volume 168, pages 633 – 644, 2006.

[9]J. F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*, volume 30 of *Nonconvex Optimization and Its Applications*. Springer, 1999.

[10]T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, San Diego, CA, 2nd edition, 1995.

[11]V. M. Bier. Choosing What to Protect. *Risk Analysis*, 27(3):607–620, 2007.

[12]M. Blanco, A. Valino, J. Heijs, T. Baumert, and J. G. Gomez. The Economic Cost of March 11: Measuring the direct economic cost of the terrorist attack on March 11, 2004 in Madrid. *Terrorism and Political Violence*, 19(4):489–509, 2007.

[13]M. Breton, A. Alg, and A. Haurie. Sequential Stackelberg Equilibria in Two-Person Games. *Optimization Theory and Applications*, 59(1):71–97, 1988.

[14]G. Brown, M. Carlyle, J. Kline, and K. Wood. A Two-Sided Optimization for Theater Ballistic Missile Defense. In *Operations Research*, volume 53, pages 263–275, 2005.

[15]G. Brown, M. Carlyle, J. Royset, and K. Wood. On The Complexity of Delaying an Adversary's Project. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Next Wave in Computing, Optimization and Decision Technologies*, pages 3–17. Springer, 2005.

[16]G. Brown, M. Carlyle, J. Salmeron, and K. Wood. Defending Critical Infrastructure. In *Interfaces*, volume 36, pages 530 – 544, 2006.

[17]V. Conitzer and T. Sandholm. Computing the Optimal Strategy to Commit to. In *ACM EC-06*, pages 82–90, 2006.

[18]D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[19]N. Gatti. Game Theoretical Insights in Strategic Patrolling: Model and Algorithm in Normal-form. In *ECAI-08*, pages 403–407, 2008.

[20]J. C. Harsanyi and R. Selten. A Generalized Nash Solution for Two-person Bargaining Games With Incomplete Information. *Management Science*, 18(5):80–106, 1972.

[21]A. Jiang and K. Leyton-Brown. A Polynomial-time Algorithm for Action-Graph Games. In *Artificial Intelligence*, pages 679–684, 2006.

[22]C. Kiekintveld, M. Jain, J. Tsai, J. Pita, M. Tambe, and F. Ordóñez. Computing Optimal Randomized Resource Allocations for Massive Security Games. In *AAMAS-09*, 2009.

[23]D. Koller and B. Milch. Multi-agent Influence Diagrams for Representing and Solving Games. *Games and Economic Behavior*, 45(1):181–221, 2003.

[24]R. Larson. A Hypercube Queueing Modeling for Facility Location and Redistricting in Urban Emergency Services. In *Journal of Computers and Operations Research*, volume 1, pages 67–95, 1974.

[25]G. Leitmann. On Generalized Stackelberg Strategies. *Journal of Optimization Theory and Applications*, 26(4):637–643, 1978.

[26]R. Looney. Economic Costs to the United States Stemming From the 9/11 Attacks. *Strategic Insights*, 1(6), August 2002.

[27]A. Murr. The Element of Surprise. *Newsweek National News*, 28 September 2007. `http://www.msnbc.msn.com/id/21035785/site/newsweek/page/0/`.

[28]Nie, R. Batta, Drury, and Lin. Optimal Placement of Suicide Bomber Detectors. In *Military Operations Research*, volume 12, pages 65–78, 2007.

[29]M. J. Osbourne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[30]P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordóñez, and S. Kraus. Playing Games with Security: An Efficient Exact Algorithm for Bayesian Stackelberg games. In *AAMAS-08*, pages 895–902, 2008.

[31]P. Paruchuri, J. P. Pearce, M. Tambe, F. Ordóñez, and S. Kraus. An Efficient Heuristic Approach for Security Against Multiple Adversaries. In *AAMAS*, 2007.

[32]P. Paruchuri, M. Tambe, F. Ordóñez, and S. Kraus. Safety in Multiagent Systems by Policy Randomization. In *SASEMAS*, 2005.

[33]P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus. Security in Multiagent Systems by Policy Randomization. In *AAMAS*, 2006.

[34]J. Pita, M. Jain, M. Tambe, F. Ordóñez, S. Kraus, and R. Magori-Cohen. Effective Solutions for real-world Stackelberg games. In *AAMAS*, 2009.

[35]J. Pita, M. Jain, C. Western, C. Portway, M. Tambe, F. Ordóñez, S. Kraus, and P. Parachuri. Deployed ARMOR protection: The application of a game-theoretic model for security at the Los Angeles International Airport. In *AAMAS-08 (Industry Track)*, 2008.

[36]S. Ruan, C. Meirina, F. Yu, K. R. Pattipati, and R. L. Popp. Patrolling in a Stochastic Environment. In *10th Intl. Command and Control Research and Tech. Symp.*, 2005.

[37]T. Sandler and D. G. A. M. Terrorism and Game Theory. *Simulation and Gaming*, 34(3):319–337, 2003.

[38]V. Srivastava, J. Neel, A. B. MacKenzie, R. Menon, L. A. Dasilva, J. E. Hicks, J. H. Reed, and R. P. Gilles. Using Game Theory to Analyze Wireless Ad Hoc Networks. *IEEE Communications Surveys and Tutuorials*, 7(4), 2005.

[39]D. Stevens and et. al. Implementing Security Improvement Options at Los Angeles International Airport. 2006. `http://www.rand.org/pubs/documented_briefings/2006/RAND_DB499-1.pdf`.

[40]M. E. Taylor, C. Kiekintveld, C. Western, and M. Tambe. Is There a Chink in Your ARMOR? Towards Robust Evaluations for Deployed Security Systems. In *IJCAI Workshop on Quantitative Risk Analysis for Security Applications (QRASA)*, 2009.

[41]M. E. Taylor, C. Kiekintveld, C. Western, and M. Tambe. A Framework for Evaluating Deployed Security Systems: Is There a Chink in Your ARMOR? In *Informatica*, 2010.

[42]P. Thornton. London Bombings: Economic Cost of Attacks Estimated at 2bn, July 2005. `http://www.independent.co.uk/news/business/news/economic-cost-of-attacks-estimated-at-1632bn-499281.html`.

[43]M. Treisman and A. Faulkner. Generation of Random Sequences by Human Subjects: Cognitive Operations or Psychological Process? *Journal of Experimental Psychology*, 116(4):337–355, 1987.

[44]J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe. IRIS - A Tool for Strategic Security Application in Transportation Networks. In *AAMAS Industry Track*, 2009.

[45]H. von Stackelberg. *Marktform und Gleichgewicht*. Springer, Vienna, 1934.

[46]B. von Stengel and S. Zamir. Leadership with Commitment to Mixed Strategies. Technical Report LSE-CDAM-2004-01, CDAM Research Report, 2004.

[47]W. A. Wagenaar. Generation of Random Sequences by Human Subjects: A Critical Survey of Literature. *Psychological Bulletin*, 77(1):65–72, 1972.

[48]K. wei Lye and J. M. Wing. Game Strategies in Network Security. *International Journal of Information Security*, 4(1–2):71–86, 2005.

[49]L. M. Wein. Homeland Security: From Mathematical Models to Policy Implementation. In *Operations Research*, 2008.

[50]H. Willis, A. Morral, T. Kelly, and J. Medby. *Estimating Terrorism Risk*. RAND Corporation, 2005.