Interfaces
Vol. 40, No. 4, July–August 2010, pp. 267–290
ISSN 0092-2102 | EISSN 1526-551X | 10 | 4004 | 0267

informs

DOI 10.1287/inte.1100.0505

Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshal Service

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

Fernando Ordóñez
Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, California 90089; and Department of Industrial Engineering, University of Chile, 8330111 Santiago, Chile, fordon@usc.edu

The increasing threat of terrorism makes security at major locations of economic or political importance a major concern. Limited security resources prevent complete security coverage, allowing adversaries to observe and exploit patterns in patrolling or monitoring, and enabling them to plan attacks that avoid existing patrols. The use of randomized security policies that are more difficult for adversaries to predict and exploit can counter their surveillance capabilities. We describe two applications, ARMOR and IRIS, that assist security forces in randomizing their operations. These applications are based on fast algorithms for solving large instances of Bayesian Stackelberg games. Police at the Los Angeles International Airport deploy ARMOR to randomize the placement of checkpoints on roads entering the airport and the routes of canine unit patrols within the airport terminals. The Federal Air Marshal Service has deployed IRIS in a pilot program to randomize the schedules of air marshals on international flights. This paper examines the design choices, information, and evaluation criteria that were critical to developing these applications.

Key words: game theory: Stackelberg games; programming: integer, applications; programming: integer, theory.
History: This paper was refereed.

Protecting critical infrastructure and targets, such as political figures and airports, historical landmarks, and power generation facilities, 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, also making them a prime target for terrorists and extremely difficult for law enforcement agencies to protect. 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 (Looney 2002) and almost 3,000 lives lost. The 2004 Madrid commuter train bombings resulted in 191 lives lost and 1,755 people wounded; its estimated cost was €212 million (Buesa et al. 2007). In the 2005 London subway and bus bombings, 52 lives were lost, 700 people were wounded, and the estimated cost was £2 billion (Thornton 2005).

Measures for protecting potential target areas include monitoring entrances or inbound roads, checking inbound traffic, and adding patrols aboard transportation vehicles. However, limited resources typically make it impossible to provide full security coverage. Furthermore, adversaries can observe security arrangements over time and exploit predictable patterns to their advantage. One way to mitigate their ability to exploit patterns is the judicious use of randomization in scheduling the actions of security forces. We developed two software assistants, Assistant for Randomized Monitoring Over Routes (ARMOR) and Intelligent Randomization In Scheduling (IRIS), that address many difficulties of

randomization and provide an easy-to-use solution for security forces.

An important issue that must be addressed in randomizing security operations is how to weight the security actions. An obvious approach is to use a uniform random policy that treats all targets or entry points the same. However, this approach fails to consider that some targets are more attractive or vulnerable than others. As a result, valuable security resources might be used to protect relatively unimportant targets. The defense strategy should emphasize the protection of high-value targets. For example, a sophisticated security policy could weight the protection provided for each target based on that target's value. However, this weighted randomization still fails to consider intelligent attackers who can alter their strategies based on knowledge of the security strategy.

Manually generating a random security policy is a costly and labor-intensive process. Humans are not skilled at generating truly random security schedules (Wagenaar 1972, Treisman and Faulkner 1987) and can easily fall into predictable patterns. Furthermore, scheduling security forces in transportation networks and other security domains is a prohibitively large problem, even if we do not consider randomization.

ARMOR and IRIS address these difficulties by using game-theoretic models and solution algorithms to determine randomization strategies that consider target values and assume intelligent adversary responses to security measures. Game theory is a well-established paradigm for addressing situations with multiple self-interested decision makers (Fudenberg and Tirole 1991). In our solutions, we model security games as Stackelberg games (von Stackelberg 1934) between a defender (i.e., the security forces) and an attacker (i.e., a terrorist adversary). Stackelberg games are bilevel models (Bard 1999) that consider an attacker's ability 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 (Conitzer and Sandholm 2006) 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 then be used to generate specific schedules for security patrols.

ARMOR (Pita et al. 2008), which was developed for the Los Angeles International Airport (LAX) police, randomizes checkpoints on the roadways entering the airport and the canine patrol routes within the airport terminals. The airport police have used it since August 2007; as of January 2009, ARMOR-scheduled checkpoints had inspected 285,589 vehicles. IRIS (Tsai et al. 2009) was developed to assist the Federal Air Marshal Service (FAMS) with randomly scheduling air marshals on flights. FAMS has deployed it in limited use since October 2009. Both are interactive software assistants that allow domain experts to change domain parameters when necessary. A model of the domain as a Bayesian Stackelberg game and fast solution algorithms for computing an optimal solution to the game model underlie each tool. 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 (Paruchuri et al. 2006, 2007; Kiekintveld et al. 2009).

Our approach of using Stackelberg games to model real-world security problems is applicable in a wide range of domains with similar attributes; these include (1) intelligent players, (2) player strategies that are observable by other players, (3) varying preferences among targets, and (4) infeasibility of fully covering all targets.

Some examples of similar security environments include computer networks, checkpoints at subway stations, inspections at ports, and monitoring of other mediums of public transport. A deterministic strategy is a weakness for the defender when coverage is incomplete. Furthermore, because variations in target values are ubiquitous in these domains, intelligent and weighted randomization can provide a greater degree of protection.

We have organized the remainder this paper as follows. *Related Work* addresses game-theoretic and non-game-theoretic literature in the security domain. *Methodology* discusses Bayesian Stackelberg game methodology, its formal definition, and the technical formulation of the security game. *Modeling the LAX and FAMS Domains* describes the LAX and FAMS domains. In *Software Assistants*, we present the system

architecture of ARMOR and IRIS. In the *Evaluation* section, we discuss assessments of the system. We conclude with a *Summary* section.

Related Work

Much of the related work in the game-theoretic and non-game-theoretic literature that studies the security domain is theoretical analysis of hypothetical scenarios; however, our work focuses on developing tools for use in real-world security operations. This requires us to address many practical aspects of the problem that arise only in the field.

The related work is in three main areas. The first applies optimization techniques to model the security domain; it does not address the strategic aspects of the problem. Although these methods provide a randomization strategy for the defender, they do not consider the adversary's ability to observe the defender's actions and accordingly adjust behavior. Examples of such approaches include methods that are based on learning, Markov decision processes (MDPs), and partially observable MDPs, as Ruan et al. (2005) and Paruchuri et al. (2006) discuss. As part of this work, these authors model the patrolling problem with varying incident rates in each location and solve for optimal routes using an MDP framework. Another example is the "hypercube queueing model" (Larson 1974), which is based on queueing theory and depicts the detailed spatial operation of urban police departments and emergency medical services. Examples of its application include police-beat design and allocation of patrolling time. 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 consider the intelligent attacker who can observe and exploit patterns in the security policy. Because a policy that is based on the historical frequency of attacks is essentially reactive, an intelligent attacker will always be one step ahead of a defender.

A second set of work uses Stackelberg games to model a variety of security domains. Bier (2007) strongly endorses this type of modeling. Game-theoretic models have been applied in many homeland security settings, including protecting critical infrastructure (Brown et al. 2006, Pita et al. 2008, Nie et al. 2007). Wein (2008) applies Stackelberg games to screening visitors entering the United States. These works model the US government as the leader that specifies the biometric identification strategy to maximize the detection probability using fingerprint matching and the follower as the terrorist who can manipulate the fingerprint's image quality. They have also been used for studying missile defense systems (Brown et al. 2005a) and the development of an adversary's weapon systems (Brown et al. 2005b). Inspection games, a family of Stackelberg games, are closely related to the security games; they include models of arms inspections and border patrols (Avenhaus et al. 2002). Other recent works address randomized security patrolling using Stackelberg games for generic "police-and-robber" scenarios (Gatti 2008) and perimeter patrols (Agmon et al. 2008). Our work differs in two main aspects. First, we use a new, more efficient game representation and mixed-integer linear program (MILP) for modeling and solving the Stackelberg games to enable us to scale systems to represent 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 in which we are interested.

A third area of related work is the application of game-theoretic techniques, which are not based on Stackelberg games, to security applications. Security problems ranging from computer network security (Lye and Wing 2005, Srivastava et al. 2005) to terrorism (Sandler and Arce 2003) are increasingly being studied using game-theoretic analysis. Babu et al. (2006) used linear programming approaches to model passenger security systems at US airports; however, their objective was to classify the passengers into various groups and then screen them based on the group to which they belong. Thus, game theory has been used in security domains; in contrast, our work focuses on overcoming the challenges that arise from its application in the real world.

Methodology

ARMOR and IRIS build on the game-theoretic foundations to address multiple types of players—the police force and the adversary—to provide a randomized security policy. The algorithms used build on

several years of research reported in the Autonomous Agents and Multiagent Systems (AAMAS) conference main track and workshops (Paruchuri et al. 2005, 2006, 2007). Although the major developments in this 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.

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 could be either individuals or groups, e.g., a police force or terrorist organization, cooperating to execute a joint strategy. Each player has a set of possible *pure strategies*, which we denote as $\sigma_\Theta \in \Sigma_\Theta$ and $\sigma_\Psi \in \Sigma_\Psi$. A *mixed strategy* allows a player to play a probability distribution over pure strategies, denoted as $\delta_\Theta \in \Delta_\Theta$ and $\delta_\Psi \in \Delta_\Psi$. The payoffs for each player are defined over the space of all possible joint pure-strategy outcomes: $\Omega_\Theta, \Omega_\Psi$: $\Sigma_\Psi \times \Sigma_\Theta \to \mathcal{R}$ for the defender and 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 that optimizes its own payoffs. Stated formally, the attacker's strategy in a Stackelberg security game becomes a function that selects a strategy for each possible leader strategy: F_Ψ: $\Delta_\Theta \to \Delta_\Psi$.

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 (Osbourne and Rubinstein 1994). Stackelberg equilibrium is a refinement of a Nash equilibrium that is specific to Stackelberg games. It is a form of subgame perfect equilibrium in that it requires each player to 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 noncredible threats off the equilibrium path. Although subgame perfection is a natural requirement, it does not guarantee a unique solution in cases in which the follower is indifferent among a set of strategies. The literature contains two forms of Stackelberg equilibria that identify unique outcomes, as Leitmann (1978) proposed, and are typically called "strong" and "weak" (Breton et al. 1988). The strong form assumes that the follower will always choose the optimal strategy for the leader in cases of indifference; the weak form assumes that the follower will choose the worst strategy for the leader. Unlike the weak form, strong Stackelberg equilibria (SSEs) are known to exist in all Stackelberg games (Başar and Olsder 1999). 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 (von Stengel and Zamir 2004). We use an SSE partly for these reasons and because it is the most commonly used solution concept in the related literature (Osbourne and Rubinstein 1994, Conitzer and Sandholm 2006, Paruchuri et al. 2008).

Definition 1. A set of strategies (δ_Θ, F_Ψ) forms an SSE if the strategies satisfy the following constraints.

1. The leader plays a best response:

$$\Omega_\Theta(\delta_\Theta, F_\Psi(\delta_\Theta)) \geq \Omega_\Theta(\delta'_\Theta, F_\Psi(\delta'_\Theta)) \quad \forall\, \delta'_\Theta \in \Delta_\Theta.$$

2. The follower plays a best response:

$$\Omega_\Psi(\delta_\Theta, F_\Psi(\delta_\Theta)) \geq \Omega_\Psi(\delta_\Theta, \delta_\Psi) \quad \forall\, \delta_\Theta \in \Delta_\Theta,\ \delta_\Psi \in \Delta_\Psi.$$

3. The follower breaks ties optimally for the leader:

$$\Omega_\Theta(\delta_\Theta, F_\Theta(\delta_\Theta)) \geq \Omega_\Theta(\delta_\Theta, \delta_\Psi) \quad \forall\, \delta_\Theta \in \Delta_\Theta,\ \delta_\Psi \in \Delta^*_\Psi(\delta_\Theta),$$

where $\Delta^*_\Psi(\delta_\Theta)$ is the set of follower best responses, as noted above.

Whether 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 good or bad for the leader. In the "rock, paper, and scissors" game (Gintis 2009), forcing commitment to a pure strategy would guarantee a loss; however, von Stengel and Zamir (2004) show that the ability to commit to a mixed strategy always weakly increases the leader's payoffs in equilibrium profiles of the game. In the context of a Stackelberg security game, a deterministic policy is a liability for the defender (the leader) and a credible randomized security policy is an advantage. Our model allows the defender to commit to mixed strategies.

The Bayesian extension to the Stackelberg game allows multiple types of players; each type is associated with its own payoff values. For the security

games discussed in this paper, we assume only one leader type (e.g., only one police force) but multiple follower types (e.g., multiple adversary types trying to infiltrate security). The set of follower types is denoted by Γ and each type γ is represented by a payoff matrix. The leader does not know the follower's type. The goal is to find the optimal mixed strategy to which the leader can commit, given that each follower type knows the mixed strategy of the leader when choosing its own strategy. Payoffs for each type are defined over all possible joint pure-strategy outcomes: Ω_Θ: $\Sigma_\Psi^\Gamma \times \Sigma_\Theta \to \mathcal{R}$ 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_Ψ^γ: $\Delta_\Theta \to \Delta_\Psi^\gamma$, which still satisfies constraints (2) and (3).

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 the description of a Stackelberg game if we map the attackers to the follower's role and the defender to the leader's role (Avenhaus et al. 2002, Brown et al. 2006). 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 infrastructure entity. The strategy for the leader is a mixed strategy spanning the various possible actions.

Using conventional methods to represent security games in normal form has two major problems. First, many solution methods require the use of a Harsanyi transformation when dealing with Bayesian games (Harsanyi and Selten 1972). The Harsanyi transformation converts a Bayesian game into a normal-form game; however, the new game may be exponentially larger than the original Bayesian game. Our compact representation avoids this Harsanyi transformation; we directly operate on the Bayesian game. This 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 game matrix for the individual follower types (see the appendix); Paruchuri et al. (2008) provide additional detail.

The second problem is that the defender has many possible resources to schedule. 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, with 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, with each row corresponding to whether the corresponding target was covered. Such a representation is equivalent to the normal-form representation for the class of problems we address in this work; Kiekintveld et al. (2009) provide additional details. This compactness in our representation is possible because the payoffs for the leader in these games depend on whether the attacked target was covered—not on which other targets were covered (or not covered). The representation we use avoids both potential problems by using methods similar to other compact representations for games (Koller and Milch 2003, Jiang and Leyton-Brown 2006).

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, $R = \{r_1, \ldots, r_m\}$; e.g., in the FAMS domain, targets could be flights and resources could be federal air marshals. Table 1 shows four payoffs that define the possible outcomes for an attack on a target. When facing an adversary of type γ, the defender's payoff is denoted $U_\Theta^{\gamma,u}(t)$ for an uncovered attack and $U_\Theta^{\gamma,c}(t)$ for a covered attack. 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: The table illustrates payoffs for defender and attacker for an attack on a target.

A crucial feature of the model is that payoffs depend only on the target attacked and whether it is covered by the defender. The payoffs do not depend on the remaining aspects of the schedule, e.g., 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 terminal 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 c_t, the probability that each target is covered. 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 a probability of 1. This is without loss of generality because an SSE solution still exists under this restriction (Paruchuri et al. 2008). Thus, the follower of type γ can choose any pure strategy $\sigma_\Psi^\gamma \in \Sigma_\Psi^\gamma$, i.e., 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). By replacing Θ with Ψ, the same notation applies for each follower type. Thus, $U_\Psi^\gamma(t, C)$ gives the payoff to the attacker when a target t is attacked by an adversary of type γ. $U_\Psi^\gamma(t, C)$ and $U_\Theta^\gamma(t, C)$ are used in the MILP discussed below and provided in detail in the appendix. In Equation (3), we also define the useful notion of the attack set $\Lambda^\gamma(C)$, 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 an SSE. 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.

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

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

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

In an SSE, 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 $\hat{U}_\Theta^\gamma(C) = U_\Theta^\gamma(t^*, C) \times p^\gamma$; for the attacker, $\hat{U}_\Psi^\gamma(C) = U_\Psi^\gamma(t^*, C)$.

Modeling the LAX and FAMS Domains

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

Domain Description

The LAX and FAMS security scenarios share important characteristics. In both, a leader–follower dynamic exists between the security forces and terrorist adversaries because LAX and FAMS are both concerned with surveillance and insider threats and have limited resources to protect a very large number of possible targets, making it impossible to provide complete coverage. Finally, the targets in question clearly have different values and vulnerabilities in each domain. Therefore, Stackelberg games are an ideal model for accurately capturing the interaction between security forces and adversaries in both domains.

Our solutions must also consider differences in these domains. The first difference is the scale of the problem. LAX has eight terminals that must be protected; the air marshals are responsible for protecting tens of thousands of commercial flights each day. Second, domain experts for LAX must specify a fairly small number of payoffs; FAMS must evaluate the values of thousands of potential targets. This leads us to consider different methods and interfaces for constructing game models for each case. Finally, the FAMS domain requires more complex reasoning about spatial and temporal constraints in how resources are scheduled (e.g., a given marshal cannot be assigned to two flights with overlapping time schedules). We now describe each domain in detail.

LAX Domain. LAX is the fifth busiest and the largest destination airport in the United States; it serves 60–70 million passengers per year (Los Angeles World Airports 2010, Stevens et al. 2006). LAX is known to be a prime terrorist target on the west coast of the United States; multiple plotters have been arrested attempting to attack it (Stevens et al. 2006). To protect LAX, airport police have designed a security system that uses multiple rings of protection, e.g., vehicular checkpoints, police units patrolling the roads to the terminals, canines patrolling inside the terminals, passenger security screening, and bag checks. Airport police use intelligent randomization within two of these rings; they place vehicle checkpoints on inbound roads that service the LAX terminals (Figure 1(a)) and schedule patrols for bomb-sniffing canine units within the terminals (Figure 1(b)).

The numbers of available vehicle checkpoints and canine units are limited by resource constraints; therefore, randomization is used to increase the effectiveness of these resources while avoiding patterns in the scheduled deployments.

The eight LAX terminals have different characteristics. Some terminals serve international flights; others serve only domestic flights. They also vary in physical size, passenger loads, and levels of foot traffic. Thus, to determine a security policy, each terminal must be assessed based on its specific value and risk. In addition, airport police identified uncertainty about the adversary as a major problem to address. For example, attackers may be hard-line, well-funded international terrorists or amateur individuals. The payoff values for different attack scenarios may depend on the attacker's type and capabilities.

FAMS Domain. To dissuade potential aggressors and prevent attacks (Transportation Security Administration 2008), FAMS places undercover law enforcement personnel aboard flights originating in and departing from the United States. It does not disclose the exact methods it uses to evaluate the risks that terrorists pose on individual flights; however, we can identify many factors that might influence such an evaluation. Flights vary in their numbers of passengers; some fly over densely populated areas and others do not, and international flights serve different countries, some of which pose higher risks.

(a) LAX checkpoint

(b) Canine patrol

Figure 1: The pictures illustrate the deployment of security checkpoints and canine patrols at LAX.

The scale of the domain is massive. Tens of thousands of commercial flights are scheduled each day, and public records indicate that FAMS employs thousands of air marshals who must be scheduled on tours of flights that obey various constraints (e.g., the time required to board, fly, and disembark). Determining air marshal schedules that meet all these constraints is a computational challenge. The task is especially difficult because it must also consider the values of each flight.

Game Models

We now describe the instantiation of each domain 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. Because these values can change over time, we provide interfaces to allow the domain experts to enter key parameters, which we then use to populate the game model at run time. Once we have a game model, we compute a solution using the Efficient Randomized Allocation of SEcurity Resources with Constraints (ERASER-C) method described in the appendix. This model returns optimal coverage probabilities for each target. By sampling based on these probabilities, we obtain an explicit schedule for the security forces (i.e., checkpoints or air marshals).

LAX Game Model. We modeled the problem of scheduling vehicle checkpoints at the LAX airport as a Bayesian Stackelberg game; we omitted the canine model because it is similar to modeling checkpoints. LAX has several inbound roads at which 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 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, 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, the police receive a payoff of $U_\Theta^{c,\gamma}(i)$ and the adversary receives a payoff of $U_\Theta^{c,\gamma}(i)$. The payoff values used in the model depend on several factors: (1) the likelihood of a LAX police checkpoint intercepting an adversary crossing that checkpoint (which may depend on traffic volume); (2) the damage the adversary can cause if it attacks via a particular inbound road; and (3) the type of adversary, i.e., adversary capability. These factors, which domain experts provide as system inputs, are used to calculate the payoff values for each road. However, the game is not necessarily zero-sum. Even an adversary who is caught might derive some benefit, e.g., publicity. Ideally, we would be able to obtain estimates of the adversaries' payoffs directly from these players. Because this is impossible in practice, we rely on the domain experts to provide the most informed estimates based on intelligence information.

The attacker types in the Bayesian model represent distinct groups of adversaries. For example, a hard-core, well-financed adversary could inflict significant damage on LAX; therefore, the payoff values for an attack by this adversary type 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 the LAX police are facing only these two types of adversaries, the probability that they are facing a hard-core adversary is 20 percent, and the probability they are facing an amateur attacker is 80 percent, we would represent these probabilities in our model by a 20–80 split.

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

We model the air marshal assignment constraints by introducing the concepts 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; each air marshal type can cover a different set of schedules. The appendix gives additional details about schedules and resource types. The user can enter various parameters that define which schedules are legal, how many marshals are available, and what schedules the marshals can fly. During a preprocessing phase, these parameters are translated into the specific constraints in the game model.

The payoffs for the FAMS game, which are defined similarly to those for the ARMOR game, are based on whether a marshal is on flight i, which the adversary attacks. As we discussed in *FAMS Domain*, the exact

methods used to arrive at these values are confidential. However, we can use many intuitive factors to estimate payoff values, including information about the flight and the capabilities of the air marshals and potential adversaries. The relevant parameters were defined in discussions with domain experts; they provide the specific values as input in the scheduling process.

Software Assistants

We now describe the system architecture for each software assistant, focusing primarily on ARMOR but providing some discussion of IRIS as a comparison. During the development process, we paid particular attention to organizational acceptance. ARMOR and IRIS must be simple enough so that users (i.e., security officers) are comfortable using them regularly. Therefore, we designed them to hide as much of the complexity of the game-theoretic models as possible while still giving security officers enough flexibility to enter parameters that change frequently. Building functionality into IRIS to allow security officers to import data from other systems to facilitate data entry (e.g., importing flight information from existing databases) was also important. Finally, we had to present the schedules that the system produces in an easy-to-understand format and provide tools that allow modifications.

Both ARMOR and IRIS are stand-alone desktop applications. ARMOR was developed using the Microsoft .NET framework; IRIS is a stand-alone Java application. Because of security concerns, both systems run on machines that are not connected to any network. The underlying solution methods use the open source GNU Linear Programming Kit (GLPK) to solve the mixed-integer programs. Additional information on GLPK is available at its website, http://www.gnu.org/software/glpk/. The core architecture of the two applications (Figure 2) can be divided into these three modules.

1. Input: The input module includes an interface that allows the user to enter parameters and domain information.

2. Back-end: User inputs are translated into a game model that is passed to the Bayesian Stackelberg game solver and then to a final process that generates a sample schedule based on the computed probabilities.

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

User Input

We rely on the security officers and other domain experts to provide the knowledge required to specify the game model. Some elements of the model do not change over time; however, because other elements change frequently, we must provide the officers with a convenient way to enter the necessary values. The basic input parameters that both ARMOR and IRIS require are in four categories: (1) number of available resources and their capabilities; (2) 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 officers to save and reuse this information across multiple executions.

ARMOR and IRIS differ in the information that is preprogrammed (i.e., "hardcoded") and the information that the user must enter. For example, the set of targets is preprogrammed because the number of LAX terminals is fixed. However, the IRIS user must enter the flight information because it can change each time the system is run. Determining which parameters the officers had to set was a significant task; we modified and ran both systems multiple times before the domain experts and security officers were satisfied that we had achieved the right balance between the complexity of the input parameters and the flexibility of the systems to capture the necessary information.

ARMOR. The interface for the ARMOR canine program (Figure 3(a)) consists of a file menu, options for varying the number of available teams per day of the week, an option to change the number of days for which to create a schedule, and a monthly calendar—the parameters that the LAX police desired to edit on a daily basis. We made a special effort to develop an intuitive and user-friendly interface. When the user presses the "Generate Schedule" button, the system takes the input parameters, generates the underlying game model, and returns a schedule for the user to view. The example shown schedules six canines each morning and evening for seven days.

The interface for the ARMOR checkpoint program (Figure 3(b)) is similar; it also provides a monthly calendar and options for the number of available

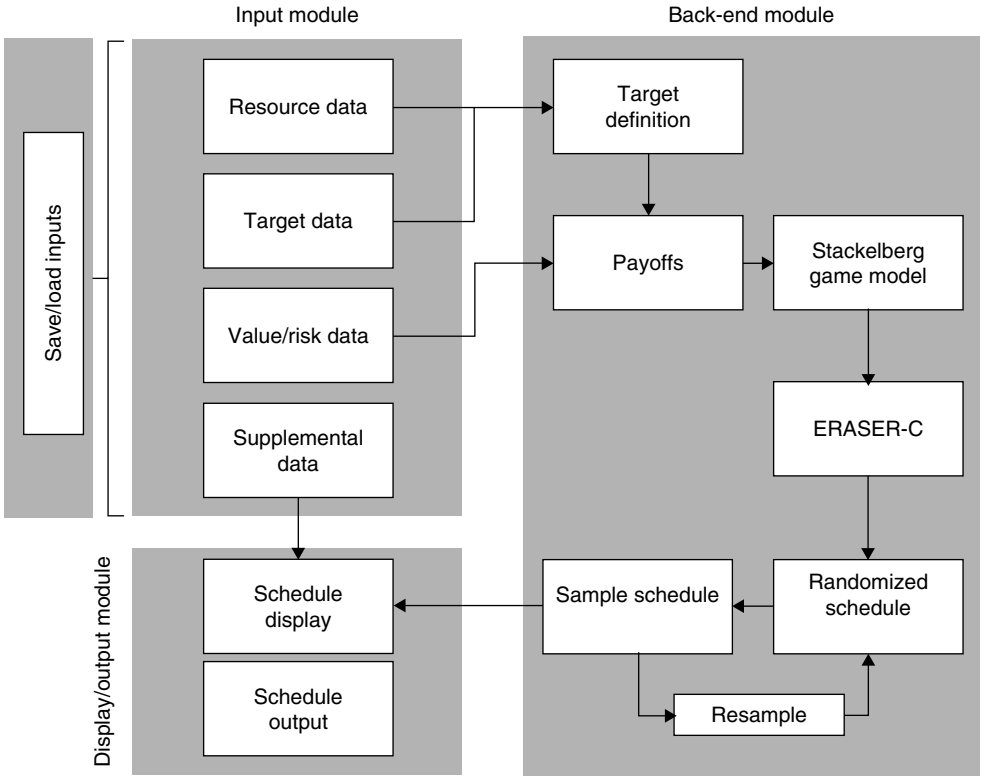
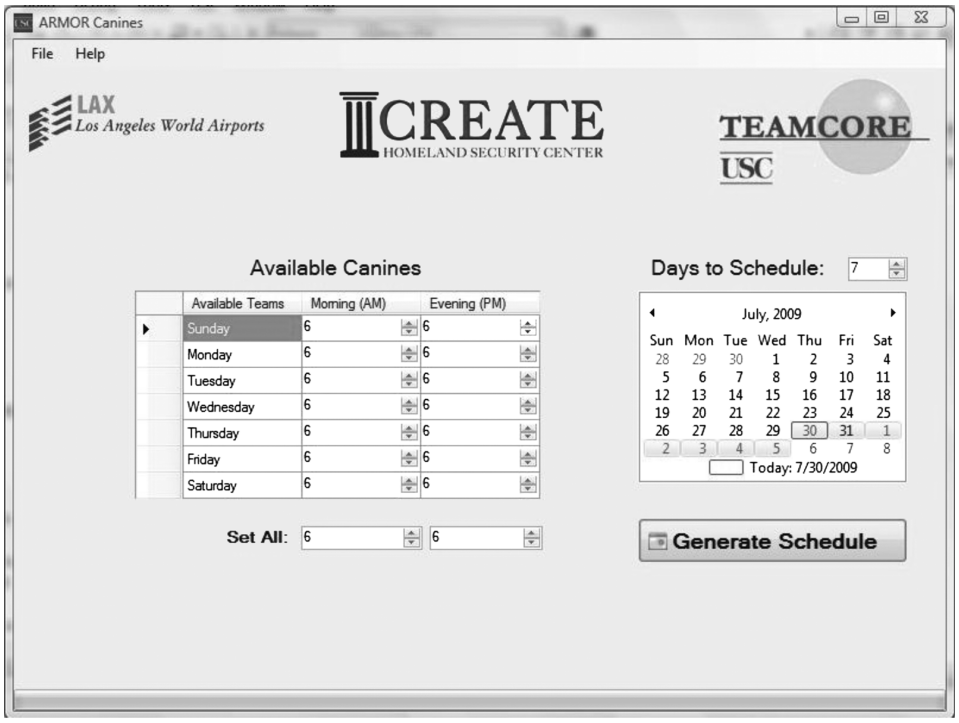


Figure 2: The figure illustrates the general structure of the security assistants.

resources, number of days for which to create a schedule, and time slots to schedule. A spreadsheet displays the proposed schedule and provides additional opportunities for the security officers to modify the schedules in a mixed-initiative setup. Three options allow the user to alter the time-scheduling actions: (1) number of checkpoints allowed during a particular time slot, (2) time interval of each time slot, and (3) number of days to schedule. Three options also restrict specific actions in the generated schedule: (1) forced checkpoint, (2) forbidden checkpoint, and (3) at least one checkpoint. These constraints, which are intended to be used sparingly, accommodate situations in which a user is faced with exceptional circumstances or has specific knowledge and wishes to influence the game's output. The spreadsheet uses a different color to represent each restriction.

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 and checkpoints specified by the user. The number of targets is the number of terminals for the canine 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 values depend on a variety of conditions, including passenger loads, cost of the infrastructure, and publicity to the adversary. Domain experts provided us with formulae, which we encoded in ARMOR, to automatically generate payoff values for all possible combinations of such conditions. Estimates of passenger load and other elements are also entered into ARMOR (details of these formulae and tools are confidential). For any given day, ARMOR can use that day's conditions to select appropriate payoff values for the targets, making it unnecessary for LAX police officers to manually enter these values and generate each schedule—a time-consuming and error-prone process. The system still retains a high degree of flexibility because values are precomputed and stored for a wide range of possible conditions.

(a) Canine interface



(b) Checkpoint interface

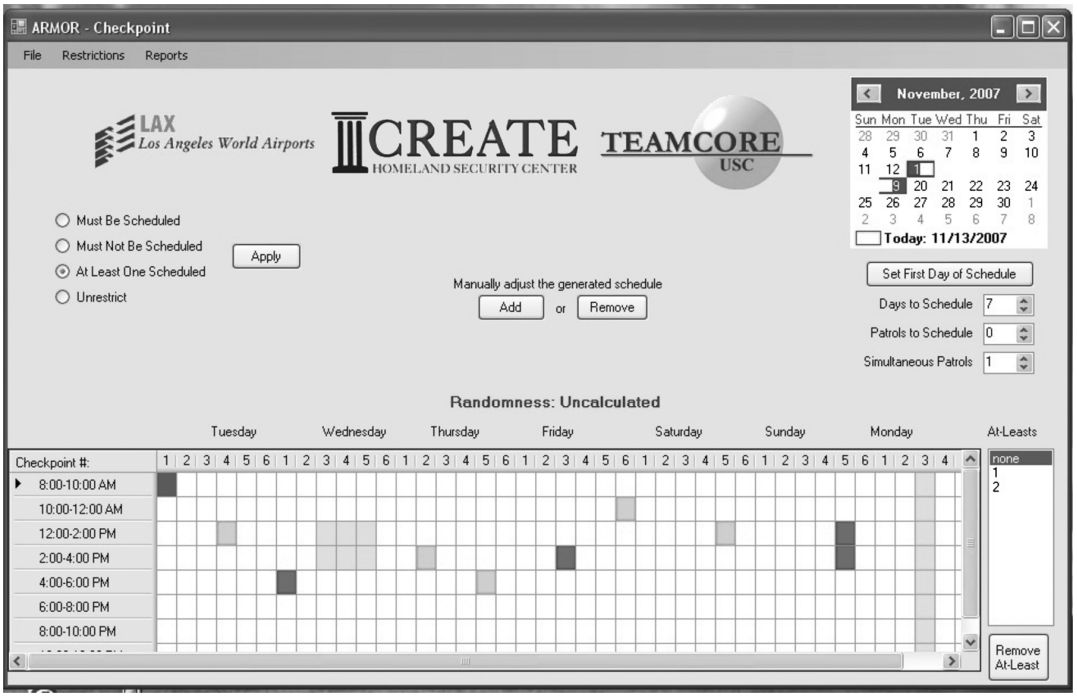


Figure 3: The screenshots show the ARMOR canine (a) and checkpoint (b) interfaces.

Intelligence information can be incorporated in ARMOR in several ways, depending on its nature. For example, if the security forces believe that there is a high probability that an attack will be attempted on a particular target, they can increase its payoff value in the game matrix, causing the deployment of more security resources to protect that target. If they learn of a new adversary type with a different set of values for the targets, they can use the Bayesian framework to add a new attacker type. Probabilities of each adversary type can also be changed to incorporate intelligence information. In addition, ARMOR's mixed-initiative interface allows security forces to manually add constraints to incorporate additional information.

IRIS. The FAMS domain is considerably larger than ARMOR's domain and the information required to build its game model changes more frequently. Therefore, the FAMS user interface and the mechanisms required to enter all necessary information are considerably more complex than ARMOR's. This additional complexity is necessary to accurately capture the environment and provide all the functionality requested by the security officers. However, it places a greater burden on the officers to learn the system; in addition, scheduling is more time-consuming than in ARMOR. As with ARMOR, finding the right level of complexity was an iterative process that involved many discussions with the security officers and other domain experts.

In the FAMS domain, we require information about the available air marshals, their scheduling constraints, the possible flights, and the risks and values to associate with each flight. The data about resources include information about the number and location of air marshals and the conditions that define legal flight schedules. Flight information includes various data about each flight, e.g., flight number, carrier, origin, destination, and aircraft type. Finally, to improve usability, we collect some information that is not strictly necessary for the game-theoretic analysis. This includes naming schemes for airports and airlines and other information that allows the system to generate schedules with a more usable format or to interface easily with other systems.

Specifying the payoff values for every possible flight is a particular challenge in this domain because we must consider thousands of flights. To elicit these values, we use an attribute-based system that is based on the threat, vulnerability, and consequence (TVC) model for estimating terrorism risk (Willis et al. 2005). By eliciting values for attributes of flights rather than specific flights, we dramatically reduce the number of entries that the user must enter. Each flight is given an aggregate value based on these attributes (the specific calculations used to determine flight risk are confidential). The values of the attributes for each flight can be populated automatically from existing databases. To allow for specific intelligence or exceptional circumstances, the user can also edit individual payoff values for any flight; however, this is rarely necessary and most of the analysis can be automated.

This preference elicitation system in IRIS has substantially reduced the number of values that the user must enter. During a restricted test run on real data, the attribute-based approach required the user to enter 114 values—regardless of the number of flights. By contrast, the non-attribute-based system required 10,284 user-entered values—2,571 valid flights over a week, each of which required four payoff values. The attribute-based approach clearly requires far fewer input parameters and remains constant as the number of flights increases, allowing for excellent scalability as we deal with larger sets of flights. Equally important, attribute-based risk assessment is an intuitive and highly scalable method that can be used in any problem in which people must distill numerous attributes of an environment into a single value for a large number of situations that share the same attributes.

Game Generation and Solution

This module, which builds a specific instance of a Bayesian Stackelberg game, is based on all data provided by domain experts and entered by security officers through the graphical user interfaces (GUIs). The specifics (insofar as we are allowed to discuss them) of how the input parameters and domain knowledge are mapped into the underlying game models are described in the *Game Models* and *User Input* sections and in the appendix.

When 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. ERASER-C returns an optimal mixed strategy for the defender—a probability distribution over the defender's actions—that 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 "resample" from the randomized schedule to generate another specific schedule; however, this capability is used rarely. The final schedules conform to the user-entered domain constraints. In particular, any specific constraints given in ARMOR, e.g., forbidden checkpoint mentioned in the *ARMOR* subsection above, are considered when final schedules are sampled.

Output Module

The output module presents the generated schedule to the user. The user can accept it or add additional constraints and run the scheduling program again.

ARMOR. The generated schedule of checkpoints and canines is presented to the user via a spreadsheet. Each row in the spreadsheet corresponds to one hour, each column corresponds to a terminal, and each entry represents a schedule generated by ARMOR. The familiarity of the police officers with spreadsheets was instrumental in their quick acceptance of the ARMOR schedules.

If a schedule includes any alerts (e.g., if ARMOR believes that user constraints are causing a very low-likelihood checkpoint to be scheduled), the user may alter the schedule by modifying the forbidden or required checkpoints or by directly modifying the schedule. If the user adds or removes constraints, ARMOR can create a new schedule. When the schedule has been finalized, it can be saved for actual use, thus completing the system cycle.

IRIS. The generated schedules are presented to the user via the application window. The user can view more detailed information about each target or can save the schedule in a file and use that file to analyze the schedule in more detail. FAMS could actually use the sample assignment of federal air marshals to flight schedules. The scheduling assistant allows the security officers to create numerous sample schedules based on the same optimal mixed strategy or manually change the assignment of federal air marshals to flight schedules to create a final schedule that meets FAMS needs. The user can also adjust any parameter entered and solve the game again. The IRIS output has the same format as the existing systems that FAMS officers use (we do not discuss IRIS output because of security concerns).

Lessons Learned

We learned some important lessons as we designed and deployed ARMOR and IRIS. First, a critical need for randomization in security operations exists. Security officials are aware that requiring humans to generate randomized schedules is unsatisfactory because, as psychological studies show (Wagenaar 1972, Treisman and Faulkner 1987), humans have difficulty in randomizing and can fall into predictable patterns. Game-theoretic randomization that appropriately weighs the costs and benefits of different actions improves randomization results. Therefore, security officials received our research enthusiastically and were eager to apply it in their practices.

Second, organizational acceptance is critical. We must be cognizant of the security officer's ease (or difficulty) in adopting a solution. Instead of asking officers to make numerous and sometimes unnecessary changes, minimizing these differences and complexities can help ensure a successful implementation. For example, tweaking the GUI to achieve a look and feel that is familiar to the user can help that user to more easily understand the system. Similarly, because changes to the infrastructure are often costly and (or) time-consuming, ease of incorporating the changes into their daily routine is essential. For example, using input parameters and creating outputs that use the same format as existing protocols minimized the additional work that our assistants would create for the security officers and led to faster acceptance.

Third, providing the security officers with operational flexibility is important. When we initially generated schedules for canine patrols and created a very detailed schedule in which we micromanaged the patrols, the officers responded negatively; however, when we created schedules that afforded them some flexibility to respond to situations on the ground, they responded more positively.

Evaluation

In security applications, safety usually trumps costs; therefore, quantifying the cost is difficult. Security and ethical concerns also exist. Taylor et al. (2009, 2010) discuss the difficulties in making such evaluations.

To illustrate these difficulties, consider an evaluation of a security system that involves a long-term study requiring prolonged periods (e.g., six months to a year) of creating schedules by alternatively using the previous methods and our game-theoretic methods. During this time, all other variables that are relevant to security (e.g., number and behavior of adversaries, economic conditions, geopolitics, and number of travelers) must remain constant. In addition, we assume that the adversaries' actions are influenced by the security measures they observe over a given period. A security strategy that changes during this observation would generate different actions from the adversaries than would be observed in an actual deployment. Therefore, we did not conduct such a long-term study of the proposed security strategies in the real world.

Instead, we provided a multifaceted evaluation that combines evidence from as many sources as possible, including expert evaluations, data from the LAX deployment, and data from various computer simulations. Perhaps the most conclusive evidence of our software assistants' merits is their adoption and continued use by the security forces at LAX and FAMS. In *Adoption and Evaluation* below, we describe some reasons for their adoption and present arrest-record data and feedback from domain experts based on ARMOR's actual use at LAX. We use simulations to evaluate various features of ARMOR and IRIS; we also use real application data, which we modified for security reasons. We describe these experiments in *Simulation Results* and include comparisons of the game-theoretic schedules with both a uniform random benchmark and benchmarks that reflect previous LAX scheduling practices.

The evaluations we discuss in this paper do not include all the analysis that has been done on ARMOR. For example, Pita et al. (2009) test the scheduling methods that ARMOR uses against human adversaries in a controlled laboratory setting. The results generally show that ARMOR's schedules perform very well (although the potential for new methods that exploit certain predictable patterns in the responses of adversaries seems to exist).

Adoption and Evaluation

FAMS conducted a quantitative and qualitative internal evaluation of IRIS and determined that the system meets its requirements. Although it would not disclose any details of this evaluation, FAMS informed us that because of this evaluation, it has begun scheduling federal air marshals in flights in a pilot phase.

ARMOR has been in use for more than two years. Hence, in this section, we present a more extensive evaluation of ARMOR by discussing three sources of evidence that prove ARMOR's value to LAX in its real-world deployment: arrest data, domain expert evaluation, and ease of use.

Arrest Data. We received summarized and actual reports from the LAX police showing the number of violations detected at checkpoints in 2007, 2008, and January 2009. The violations listed in the January 2009 report are as follows:

1. January 3, 2009: Loaded 9-millimeter (9mm) pistol discovered;
2. January 3, 2009: Loaded 9mm handgun discovered (no arrest);
3. January 9, 2009: 16 handguns, four rifles, one pistol, and one assault rifle discovered (some fully loaded);
4. January 10, 2009: Two unloaded shotguns discovered (no arrest);
5. January 12, 2009: Loaded 22-calibre (22cal) rifle discovered;
6. January 17, 2009: Loaded 9mm pistol discovered;
7. January 22, 2009: Unloaded 9mm pistol discovered (no arrest).

In Figure 4, we tabulate the number of violations for the 15 months prior to ARMOR's deployment and for 2008 when ARMOR was in use for the full year. The x-axis shows the types of violations; the y-axis shows the numbers of violations detected, which are substantially higher at LAX after ARMOR's deployment than in the preceding period. For example, 30 drug-related offenses were detected following ARMOR's deployment; only 4 such offenses were detected before its deployment. We must be careful not to draw too many conclusions from these

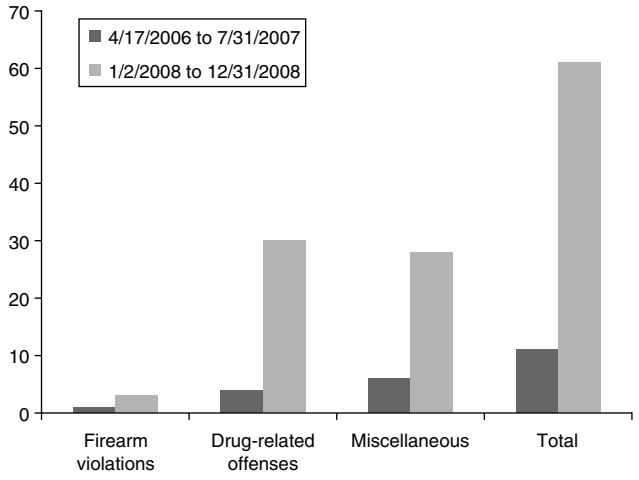


Figure 4: This graph shows a comparison of the number of violations detected at LAX checkpoints before and after ARMOR's deployment.

data because of the large number of uncontrolled variables (e.g., the number of checkpoints was not consistent during this period); however, the ARMOR checkpoints appear to be effective.

Domain Expert Evaluation. No security system—ARMOR and IRIS included—can provide 100 percent security; however, it can aid security forces in making the best possible use of the resources at their disposal. Qualitative internal and external security reviews indicate that ARMOR is both effective and highly visible. Director James Butts of the LAX police force reports that ARMOR "makes travelers safer" and gives "a greater feeling of police presence" (Murr 2007). 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 (Committee on Homeland Security 2008). A recent external study by international transportation security experts concluded that ARMOR is 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 its efforts toward securing the Los Angeles International Airport (University of Southern California 2009).

Thus, the domain experts have been highly supportive of ARMOR. Although such studies are not very useful for quantifying ARMOR's benefits, they suggest that the domain experts believe that ARMOR generates better schedules than their previous approaches.

Ease of Use. ARMOR has been instrumental in aiding police forces to efficiently and more conveniently generate schedules to deploy additional units. For example, consider a situation in which only two canines need to be scheduled for two hours each over any of the seven terminals. Each canine could be assigned to any of the seven terminals each hour, making the search space as large as 7^4 (2,401) combinations. This search space grows exponentially with the number of canines and the number of hours for which the schedule must be generated, making it impractical for human schedulers to generate.

ARMOR has played a significant role in reducing, if not completely eliminating, the tasks of the security officers to manually construct patrolling schedules. In addition, its use has allowed them to modify the generated schedules. Furthermore, although ARMOR was designed as a mixed-initiative system, security officers have chosen not to modify its schedules and domain experts have chosen not to tweak its decisions, suggesting that they view the output schedules as high quality.

Simulation Results

To better quantify the benefits of the game-theoretic schedules produced by ARMOR and IRIS, we look at simulation results in more controlled settings. In the following subsections, we compare the achieved quality (or the defender reward) of our strategy against uniformly random and other strategies that reflect the strategies that human schedulers use. We also present run-time results in the *Run-Time Comparisons* subsection, comparing the ERASER-C run times with the previous fastest solvers for Stackelberg games—DOBSS (Paruchuri et al. 2008) and Multiple-LP approaches (Conitzer and Sandholm 2006); ERASER-C is significantly faster than either approach. To compare our solution quality with that of other approaches, we describe three sets of simulation results below.

Comparison with Uniform Random Policy. A uniform random policy is the most obvious alternative for randomization. In this policy, each defender action has equal probability, regardless of payoff. We measure solution quality by calculating the highest expected payoff attainable by the defender and

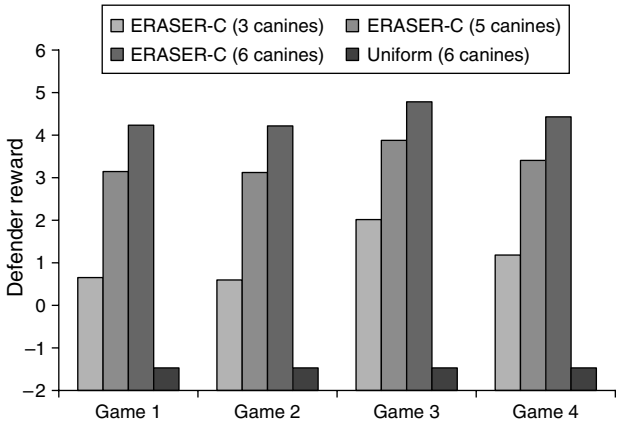


Figure 5: In this graph, we show a comparison of an ARMOR solution and a uniformly random solution for canine patrols.

use the SSE assumption that the attacker chooses an optimal response and breaks ties in favor of the defender. We ran the experiments using data from the deployment of canines in the LAX domain. Figure 5 shows the differences in defender reward obtained by scheduling canine units at LAX using ERASER-C and scheduling them using a uniform random strategy. The simulations used the actual game models for the ARMOR canine 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. ERASER-C clearly performs better using three canine units than the uniform random strategy does using six canine units. For example, the reward of a uniformly random strategy with six canine units is -1.47 on average across the four games; the reward of three, five, and six canines with ERASER-C is 1.11, 3.38, and 4.41, respectively. Thus, ERASER-C randomization provides better results using fewer resources than uniformly random strategies in the same domain.

We ran similar tests using data from the FAMS domain. We used one week of real flight data for subregions of a region that we called Region A, three separate sets of hypothetical FAMS home-city data that vary the number of federal air marshals available, and hypothetical payoffs. Region A, which we show as anonymous because of security concerns, comprises five large countries that we have designated 1–5 and a few small countries. Table 2 shows

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: This table shows the number of flights and flight schedules related to different regions used for FAMS experiments.

data on the number of flights and flight schedule for each region. We created random values for the other input parameters and held them constant throughout our testing.

In Figure 6, the y-axis represents the normalized defender reward for both strategies; all payoffs are 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 the number of flights and flight schedules increase 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 percent and under the uniform randomization strategy as 81 percent (i.e., 19 percent worse than ERASER-C). ERASER-C's solution is superior to the uniformly random solution in every region we tested.

Comparisons with Previous Scheduling Methods. The LAX security officers informed us that they had previously generated checkpoint schedules based on

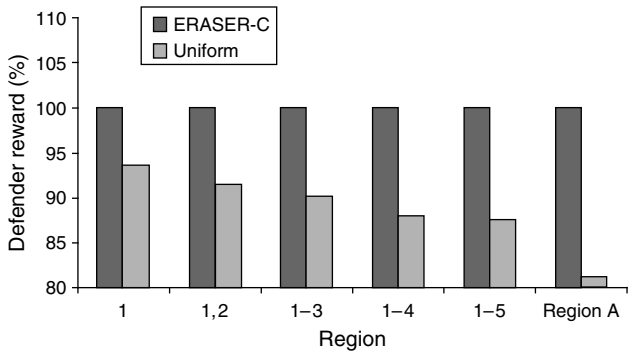


Figure 6: In this graph, we compare the ERASER-C solution quality with the uniform random solution quality.

a cyclic strategy with random perturbations. A study of their previous schedules showed clear patterns despite these random perturbations—no checkpoint was repeated for two consecutive days. Therefore, we also compared the ERASER-C strategy against two strategies: (1) a "cyclic" strategy, in which 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.

We performed our tests using the same setup that the LAX police used in ARMOR—police officers place one checkpoint on any of the five inbound roads and the rewards of the terminals are randomly chosen as numbers between 1 and 10. We varied the duration for which the adversary can make observations from 0 to 100 days in increments of 25 days and averaged our results over 100 trials. In these simulations, we assumed 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 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 we are focusing on a single checkpoint consistent with previous scheduling approaches—not on the multiple canines used for scheduling in Figure 5. The ERASER-C strategy has a higher average defender reward compared with the other two strategies because the adversary can better predict the defender–action cyclic and restricted uniform strategies than it can predict 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. Although these patterns can be avoided by using uniformly random strategies, such strategies do not consider the different preferences over targets of the defender. ARMOR provides weights for the targets such that the average defender reward is highest when compared with both cyclic and restricted uniform strategies. In addition, ARMOR strategies are not only weighted random; they also consider that the adversary observes the defender's strategy and then makes an informed, rational choice.

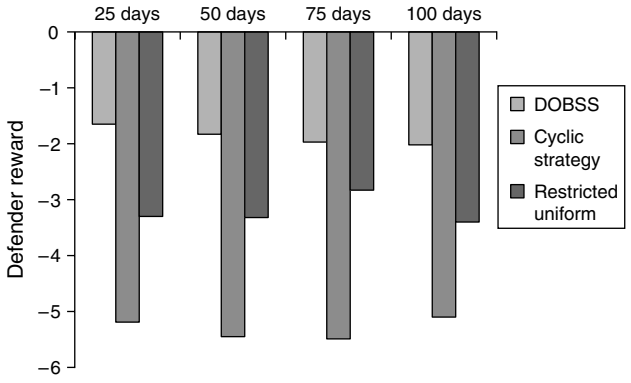


Figure 7: In this graph, we compare the defender reward for the ARMOR strategy and policies representative of previous methods.

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. Figures 8(a), 8(b), and 8(c) show the results of these experiments, which we conducted using the LAX domain. We considered a scenario in which one, two, or three security units had to defend 10 targets against two types of adversaries. The x-axis represents the probabilities of occurrence for type 1 and type 2 adversaries. The x-axis shows the probability p of adversary type 2 (the probability of adversary type 1 is then $1 - p$). 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 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 and the reward obtained by a uniform random strategy is -2.112. 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) shows the probability distribution on the x-axis and the reward obtained on the y-axis. It also shows the difference in the reward obtained when two resources (checkpoints) are available—the reward of the ERASER-C strategy is greater than the reward of

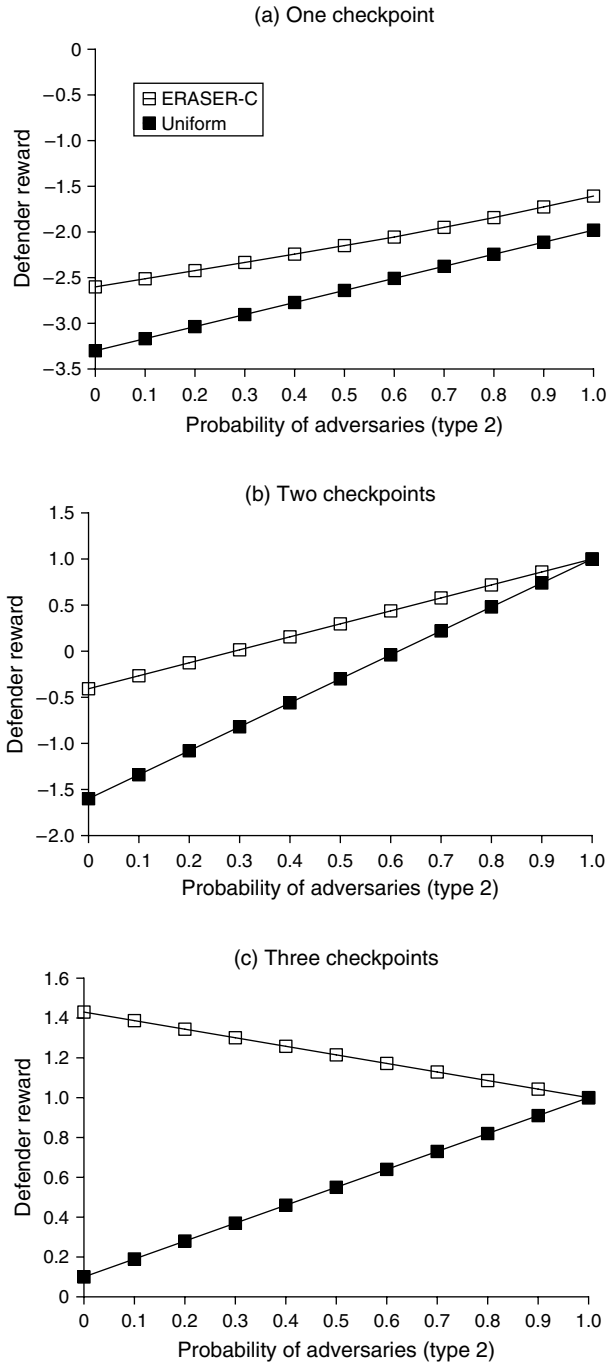


Figure 8: In these graphs, we compare the ERASER-C's solution quality with that of the uniform random strategy in the Bayesian case.

the uniform random strategy. In the two-resource situation, the type 2 adversary chooses the action *none* (to not attack). We observe that the rewards of the ERASER-C and the uniform strategies are the same when only the type 2 adversary is present. Figure 8(c) presents the case of three resources (checkpoints). The reward values obtained by ERASER-C in such a case are always positive because the chances of catching the type 1 adversary improve significantly with three checkpoints. The reward of ERASER-C decreases as the probability of occurrence of the type 1 adversary decreases. Note that the type 2 adversary, as in the case of two resources, decides *none*; hence, the reward of the ERASER-C strategy and the uniformly random strategy are the same when only the type 2 adversary is present.

Run-Time Comparisons. For ARMOR and IRIS to be useful, their solution algorithms must be able to solve very large problem instances in a reasonable amount of time. To demonstrate the scalability of our algorithms, we conducted experiments in which we scaled three different parameters of the domain: the number of targets, the number of available resources, and the number of adversaries. In each case, ERASER-C was more scalable than DOBSS (Paruchuri et al. 2008) and Multiple-LPs (Conitzer and Sandholm 2006). In Figures 9 through 12, we show results for a large realistic game with thousands of targets and hundreds of resources from the FAMS domain; these games can be solved in less than 15 minutes.

Scaling the Number of Targets. We first present the results of increasing the number of targets from 25 to 200, fixing both the number of resources and adversary types to 1 (Figure 9). The *x*-axis shows

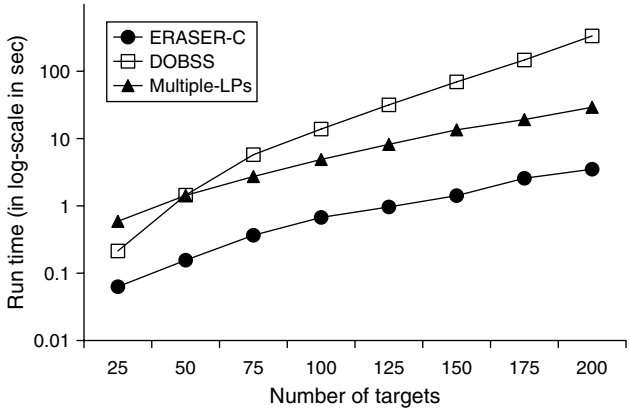


Figure 9: In this graph, we compare the run times of ERASER-C with previous methods as we increase the number of targets.

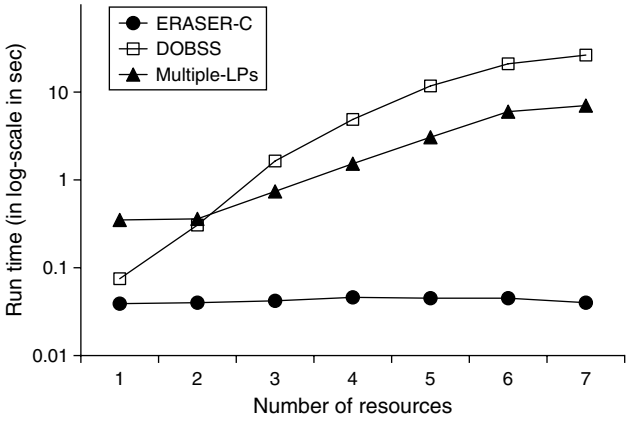


Figure 10: In this graph, we compare the run times of ERASER-C with previous methods as we increase the number of resources.

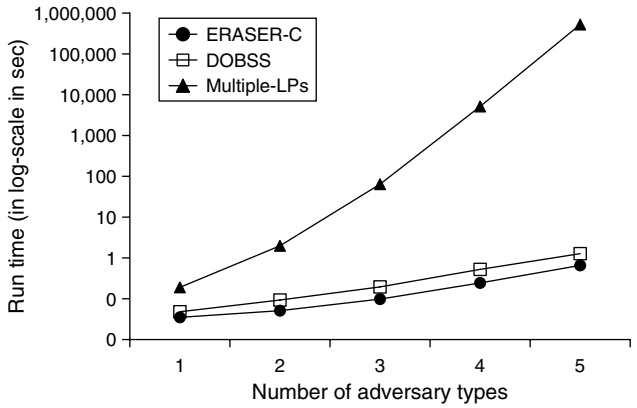


Figure 11: In this graph, we compare the run times of ERASER-C with previous methods when the number of adversary types is increased.

the number of targets; the y-axis shows the run time in seconds on a logarithmic scale. For example, for 100 targets, the run times of ERASER-C, DOBSS, and Multiple-LPs are 0.673, 13.84, and 4.89 seconds, respectively. We averaged the results over 30 trials with different randomly generated game matrices. ERASER-C is considerably faster than either algorithm. Note also that when the number of resources is scaled proportionately with the number of targets, this effect is even more dramatic (Kiekintveld et al. 2009).

Scaling the Number of Resources. In this experiment, we vary the number of resources from 1 to 7, using 15 targets and one adversary type (Figure 10). The x-axis shows the number of resources; the y-axis shows the run times in seconds on a logarithmic scale. For example, for five resources, the run times of ERASER-C, DOBSS, and Multiple-LPs are 0.044, 11.74, and 3.06 seconds, respectively. When we averaged the results over 30 trials with different randomly generated game matrices, we again found that ERASER-C was much faster than either algorithm. The effect of the combinatorial set of possible assignments that is enumerated by both DOBSS and Multiple-LPs is also clear in this plot. ERASER-C solution times are constant on the log scale as the number of resources increases; however, the other algorithms show substantial increases—even on the log scale.

Scaling the Number of Adversaries. We increased the number of adversaries from 1 to 5 (Figure 11) using 10 targets and one resource. The x-axis shows the number of adversaries; the y-axis shows run times in seconds on a logarithmic scale. For example, for four adversaries, ERASER-C and DOBSS took 0.244 and 0.524 seconds, respectively; Multiple-LPs was the slowest, taking 5,138.0 seconds. When we averaged all results over 30 trials, we found that ERASER-C was also the fastest, although DOBSS scaled similarly with a single resource. The Multiple-LPs method uses the Harsanyi transformation, which results in much slower solution times as we increase the number of adversary types.

Scaling to Large Games. Finally, we present results on a real data set for the FAMS domain to show that ERASER-C is capable of solving these games in a reasonable amount of time. We show the run time for ERASER-C to generate a schedule for different numbers of possible defense resources. The targets are defined using actual flight data from several (unnamed) regions of various sizes; all flights are between the United States and the given regions during a one-week period (Figure 12).

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 taken. We used the same game-size region information from Figure 6 and as Table 2 shows. Our tests included 100, 200, and 500 air marshals in three separate sets, respectively, as the x-axis in Figure 12 shows; for security reasons, the actual number of federal air marshals is unavailable to us. The first bar (in dark gray) on the left represents an average run time of 3.65

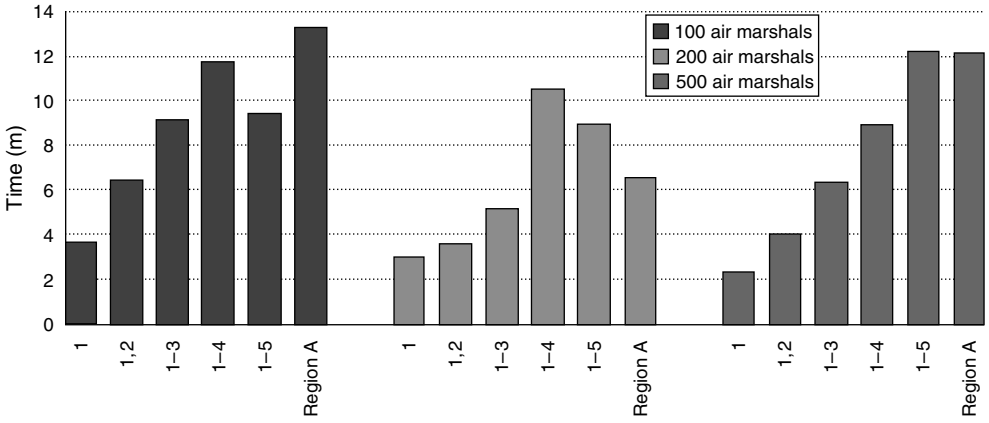


Figure 12: In this graph, we show the run-time results for realistic FAMS problems.

minutes to create a schedule for all flights to country 1 within a one-week period over 20 trials. The first bar on the left (in light gray) represents the same experiment; however, we ran it using a medium number of federal air marshals. All the game instances were completed in less than 15 minutes, which is well within FAMS' time requirements for these problems.

Summary

Monitoring and patrolling are key components of law enforcement in security domains. In generating schedules for these patrols, it is important to consider the varying weights of the targets being protected and to also realize that potential attackers often observe the procedures being used. This paper describes two scheduling assistants, ARMOR for the LAX police and IRIS for FAMS; each provides game-theoretic solutions to this problem. They assist the security forces in generating randomized patrols and ensure that differences in the value of individual targets are considered.

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

Our game-theoretic framework could also be applied directly to the randomization of security procedures that the Transportation Security Administration (TSA) conducts. TSA directs individual airports in the daily screening of passengers and bags on local flights, handles customs and border protection, and conducts random employee screenings. Our game-theoretic model could give TSA an intelligent method of selecting and scheduling these security procedures. Other examples of potential domains include patrolling at ports and subway systems.

In summary, ARMOR and IRIS represent successful transitions of game-theoretic advances to applications that are valuable in the real world.

Appendix

Solution Method

The Efficient Randomized Allocation of SEcurity Resources with Constraints (ERASER-C) algorithm takes as input a security game in the compact form described in the *Security Game Representation* section and solves for an optimal coverage vector corresponding to an SSE strategy for the defender. We allow resources to be assigned to *schedules* covering multiple targets. The set of legal schedules, $S = \{s_1 \ldots s_l\}$, 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 by using 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*, $\Omega = \{\omega_1, \ldots, \omega_v\}$, 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 $\mathcal{C}$: $S \times \Omega \to \{0, 1\}$, which is one if the type can cover the given schedule and zero otherwise. Our current implementation uses complete matrices to represent M and $\mathcal{C}$, but sparse representations could offer additional performance improvements.

The combination of schedules and resource types captures key elements of the security domains. For example, in FAMS, federal air marshals are resources, flights are potential targets, and payoff values are defined by risk analysis of the flight. However, a marshal cannot be on all possible flights because of location and timing constraints; e.g., a marshal in New York cannot board a flight 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 used to define the initial state (i.e., location) of a marshal, which defines a subset of legal schedules that determine the flights 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 two legal schedules cover targets $\{1, 2\}$ and $\{2, 3\}$. Given only these schedules, it is impossible to implement a coverage vector that places 50 percent probability on both targets 1 and 3, with no coverage of target 2.

The algorithm is an MILP, which we present in Equations (4)–(15). Table A.1 shows the notations for the symbols we use in the MILP. Equations (5) and (9) force the attack vector to choose a single target with a probability of 1. Equation (6) restricts the coverage vector to probabilities in the range $[0, 1]$, and Equation (12) constrains the coverage by the number of available resources. The coverage of each schedule must sum to the contributions of the individual resource types, as Equation (10) specifies. 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 is used. No probability may be assigned to disallowed schedules for each resource type, which Equation (13) enforces explicitly. Equation (14) defines the defender's expected payoff, contingent on the target attacked in A^γ, where γ is the follower type. The objective maximizes d^γ; therefore, for any optimal solution, $d^\gamma = U_\Theta^\gamma(C, A^\gamma)$. This also implies that C is maximal, given A^γ for any optimal solution because d^γ is maximized. Similarly, 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) and (15) 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 an SSE 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
$\mathcal{C}(s, \omega)$	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 A.1: In this table, we describe the notation used in Equations (4)–(15).

$$\max_{a, c, q, h, d, k} \sum_{\gamma \in \Gamma} d^\gamma p^\gamma \tag{4}$$

$$a_t^\gamma \in \{0, 1\} \quad \forall\, t \in T,\ \gamma \in \Gamma, \tag{5}$$

$$c_t \in [0, 1] \quad \forall\, t \in T, \tag{6}$$

$$q_s \in [0, 1] \quad \forall\, s \in S, \tag{7}$$

$$h_{s, \omega} \in [0, 1] \quad \forall\, s,\ \omega \in S \times \Omega, \tag{8}$$

$$\sum_{t \in T} a_t^\gamma = 1 \quad \forall \gamma \in \Gamma, \tag{9}$$

$$\sum_{\omega \in \Omega} h_{s,\omega} = q_s \quad \forall s \in S, \tag{10}$$

$$\sum_{s \in S} q_s M(s,t) = c_t \quad \forall t \in T, \tag{11}$$

$$\sum_{s \in S} h_{s,\omega} Ca(s,\omega) \le \mathcal{R}(\omega) \quad \forall \omega \in \Omega, \tag{12}$$

$$h_{s,\omega} \le Ca(s,\omega) \quad \forall s,\ \omega \in S \times \Omega, \tag{13}$$

$$d^\gamma - U_\Theta^\gamma(t,C) \le (1 - a_t^\gamma) \cdot Z \quad \forall t \in T,\ \gamma \in \Gamma, \tag{14}$$

$$0 \le k^\gamma - U_\Psi^\gamma(t,C) \le (1 - a_t^\gamma) \cdot Z \quad \forall t \in T,\ \gamma \in \Gamma. \tag{15}$$

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. The domain experts provided these values (see *Software Assistants*). Similarly, they provided the payoff values for the adversaries.

The values of other model parameters are calculated based on user input and the game specification. Police officers and canines are the resources for ARMOR for both checkpoints and canines. ARMOR does not differentiate between resources (e.g., all canines are assumed to be equally capable); hence, there is exactly one resource type Ω. The user enters the number of resources $\mathcal{R}$, i.e., checkpoints or canines. In ARMOR, the set of legal schedules is an assignment of a checkpoint to an inbound road; the system automatically generates it because ARMOR is aware of the airport's road map. Its capability matrix Ca consists of all ones because any resource could be assigned to any target; e.g., any canine could be assigned to any terminal.

Similarly, all model parameters in IRIS are defined based on user input and domain constraints. The federal air marshals are its resources; the FAM offices form the resource types—information supplied to IRIS by the domain experts. The security officers enter the number of resources of each type $\mathcal{R}$, i.e., the number of federal air marshals in each office, into IRIS. FAMS provides the set of legal schedules S as input to the system. Each schedule 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; e.g., federal air marshals at the Los Angeles FAM office can only cover schedules flying out of Los Angeles. Hence, only those schedules would have their capabilities set to one. The system also calculates the mapping M based on the domain specifications. For example, in IRIS, if schedule s specifies taking flight f1 followed by flight f2, then the row in M corresponding to s would have ones only in columns corresponding to f1 and f2.

The ERASER-C MILP corresponds to an SSE of the security game. The proof is based on the following 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. The complete details of the proof can be found in Kiekintveld et al. (2009).

Acknowledgments

The development of ARMOR and IRIS has been successful because of the exceptional collaboration of 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 2007-ST-061-000001. However, any opinions, findings, conclusions, or recommendations in this document are those of the authors and do not necessarily reflect the views of the United States Department of Homeland Security. F. Ordóñez would also like to acknowledge the support of FONDECYT through Grant 1090630.

References

Agmon, N., V. Sadov, G. A. Kaminka, S. Kraus. 2008. The impact of adversarial knowledge on adversarial planning in perimeter patrols. *Proc. 7th Internat. Conf. Autonomous Agents Multiagent Systems,* Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 55–62.

Avenhaus, R., B. von Stengel, S. Zamir. 2002. Inspection games. R. J. Aumann, S. Hart, eds. *Handbook of Game Theory with Economic Applications,* Vol. 3. North-Holland, Amsterdam, 1947–1987.

Babu, V. L. L., R. Batta, L. Lin. 2006. Passenger grouping under constant threat probability in an airport security system. *Eur. J. Oper. Res.* **168**(2) 633–644.

Bard, J. F. 1999. *Practical Bilevel Optimization: Algorithms and Applications* (*Nonconvex Optimization and Its Applications*). Kluwer Academic Publishers, Norwell, MA.

Başar, T., G. J. Olsder. 1999. *Dynamic Noncooperative Game Theory*, 2nd ed. Academic Press, San Diego.

Bier, V. M. 2007. Choosing what to protect. *Risk Anal.* **27**(3) 607–620.

Breton, M., A. Alj, A. Haurie. 1988. Sequential Stackelberg equilibria in two-person games. *Optim. Theory Appl.* **59**(1) 71–97.

Brown, G., M. Carlyle, J. O. Royset, R. K. Wood. 2005a. On the complexity of delaying an adversary's project. B. L. Golden,

S. Raghavan, E. A. Wasil, eds. *The Next Wave in Computing, Optimization and Decision Technologies*. Springer, New York, 3–17.

Brown, G., M. Carlyle, J. Salmeron, K. Wood. 2006. Defending critical infrastructure. *Interfaces* **36**(6) 530–544.

Brown, G., M. Carlyle, D. Diehl, J. Kline, K. Wood. 2005b. A two-sided optimization for theater ballistic missile defense. *Oper. Res.* **53**(5) 745–763.

Buesa, M., A. Valiño, J. Heijs, T. Baumert, J. González Gómez. 2007. The economic cost of March 11: Measuring the direct economic cost of the terrorist attack on March 11, 2004 in Madrid. *Terrorism Political Violence* **19**(4) 489–509.

Committee on Homeland Security. 2008. The resilient homeland—Broadening the homeland security strategy. Hearing (May 6), United States House of Representatives, Washington, DC. http://homeland.house.gov/Hearings/index.asp?ID=134.

Conitzer, V., T. Sandholm. 2006. Computing the optimal strategy to commit to. *Proc. 7th ACM Conf. Electronic Commerce, Ann Arbor, MI,* ACM, New York, 82–90.

Fudenberg, D., J. Tirole. 1991. *Game Theory*. MIT Press, Cambridge, MA.

Gatti, N. 2008. Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. *Proc. 18th Eur. Conf. Artificial Intelligence* (*ECAI 2008*), Ios Press, Amsterdam, 403–407.

Gintis, H. 2009. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*, 2nd. ed. Princeton University Press, Princeton, NJ.

Harsanyi, J. C., R. Selten. 1972. A generalized Nash solution for two-person bargaining games with incomplete information. *Management Sci.* **18**(5, Part 2) 80–106.

Jiang, A. X., K. Leyton-Brown. 2006. A polynomial-time algorithm for action-graph games. *Proc. 21st Natl. Conf. Artificial Intelligence*, AAAI Press, Menlo Park, CA, 679–684.

Kiekintveld, C., M. Jain, J. Tsai, J. Pita, F. Ordóñez, M. Tambe. 2009. Computing optimal randomized resource allocations for massive security games. *Proc. Eighth Internat. Conf. Autonomous Agents Multiagent Systems,* International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 689–696.

Koller, D., B. Milch. 2003. Multi-agent influence diagrams for representing and solving games. *Games Econom. Behav.* **45**(1) 181–221.

Larson, R. C. 1974. A hypercube queueing model for facility location and redistricting in urban emergency services. *J. Comput. Oper. Res.* **1**(1) 67–95.

Leitmann, G. 1978. On generalized Stackelberg strategies. *J. Optim. Theory Appl.* **26**(4) 637–643.

Looney, R. 2002. Economic costs to the United States stemming from the 9/11 attacks. *Strategic Insights* **1**(6), http://www.ciaonet.org/olj/si/si_1_6/si_1_6_lor01.pdf.

Los Angeles World Airports. 2010. LAX—Airport information: General description—Just the facts. Retrieved April 1, http://www.lawa.org/welcome_LAX.aspx?id=44.

Lye, K., J. M. Wing. 2005. Game strategies in network security. *Internat. J. Inform. Security* **4**(1–2) 71–86.

Murr, A. 2007. The element of surprise. *Newsweek* (September 28), http://www.newsweek.com/id/41845.

Nie, X., R. Batta, C. Drury, L. Lin. 2007. Optimal placement of suicide bomber detectors. *Military Oper. Res.* **12** 65–78.

Osbourne, M. J., A. Rubinstein. 1994. *A Course in Game Theory*. MIT Press, Cambridge, MA.

Paruchuri, P., M. Tambe, F. Ordóñez, S. Kraus. 2006. Security in multiagent systems by policy randomization. *Proc. Fifth Internat. Conf. Autonomous Agents Multiagent Systems,* ACM, New York, 273–280.

Paruchuri, P., D. Dini, M. Tambe, F. Ordóñez, S. Kraus. 2005. Safety in multiagent systems by policy randomization. *Proc. SASE-MAS Workshop, AAMAS 2005, Utrecht, The Netherlands.*

Paruchuri, P., J. P. Pearce, M. Tambe, F. Ordóñez, S. Kraus. 2007. An efficient heuristic approach for security against multiple adversaries. *Proc. Sixth Internat. Conf. Autonomous Agents Multiagent Systems,* ACM, New York, Article 181.

Paruchuri, P., J. P. Pearce, J. Marecki, M. Tambe, F. Ordóñez, S. Kraus. 2008. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. *Proc. Seventh Internat. Conf. Autonomous Agents Multiagent Systems,* International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 895–902.

Pita, J., M. Jain, F. Ordóñez, M. Tambe, S. Kraus, R. Magori-Cohen. 2009. Effective solutions for real-world Stackelberg games. *Proc. Eighth Internat. Conf. Autonomous Agents Multiagent Systems,* International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 369–376.

Pita, J., M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Parachuri, S. Kraus. 2008. Deployed ARMOR protection: The application of a game theoretic model for security at the Los Angeles International Airport. *Proc. Seventh Internat. Conf. Autonomous Agents Multiagent Systems,* International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 125–132.

Ruan, S., C. Meirina, F. Yu, K. R. Pattipati, R. L. Popp. 2005. Patrolling in a stochastic environment. *Proc. 10th Internat. Command Control Res. Tech. Symp., McLean, VA.* http://www.dodccrp.org/events/10th_ICCRTS/CD/papers/278.pdf.

Sandler, T., D. G. M. Arce. 2003. Terrorism and game theory. *Simulation Gaming* **34**(3) 319–337.

Srivastava, V., J. Neel, A. B. MacKenzie, R. Menon, L. A. DaSilva, J. E. Hicks, J. H. Reed, R. P. Gilles. 2005. Using game theory to analyze wireless ad hoc networks. *IEEE Comm. Surveys Tutorials* **7**(1–4) 46–56.

Stevens, D., T. Hamilton, M. Schaffer, D. Dunham-Scott, J. J. Medby, E. W. Chan, J. Gibson et al. 2006. Implementing security improvement options at Los Angeles International Airport. RAND, Santa Monica, CA. Retrieved November 1, 2009, http://www.rand.org/pubs/documented_briefings/2006/RAND_DB499-1.pdf.

Taylor, M. E., C. Kiekintveld, C. Western, M. Tambe. 2009. Is there a chink in your ARMOR? Towards robust evaluations for deployed security systems. Retrieved April 1, 2010, http://teamcore.usc.edu/QRASA-09/SubmissionsFinal/Paper_1.pdf.

Taylor, M. E., C. Kiekintveld, C. Western, M. Tambe. 2010. A framework for evaluating deployed security systems: Is there a chink in your ARMOR? Presentation, Workshop on Quantitative Risk Analysis for Security Applications, Los Angeles. http://teamcore.usc.edu/papers/2010/TaylorEtAl_PInformatica.pdf.

Thornton, P. 2005. Economic cost of attacks estimated at £2bn. *Independent.co.uk* (July 18), http://www.independent.co.uk/news/business/news/economic-cost-of-attacks-estimated-at-1632bn-499281.html.

Transportation Security Administration. 2008. Federal air marshals. Retrieved April 1, 2010, http://www.tsa.gov/lawenforcement/programs/fams.shtm.

Treisman, M., A. Faulkner. 1987. Generation of random sequences by human subjects: Cognitive operations or psychological process? *J. Experiment. Psych: General* **116**(4) 337–355.

Tsai, J., S. Rathi, C. Kiekintveld, F. Ordóñez, M. Tambe. 2009. IRIS—A tool for strategic security application in transportation networks. *Proc. Eighth Internat. Conf. Autonomous Agents Multiagent Systems,* International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 37–44.

University of Southern California. 2009. LAXPD bestows LA honors on Viterbi School security system builders. Retrieved April 1, 2010, http://viterbi.usc.edu/news/news/2009/laxpd-honors-viterbi.htm.

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

von Stengel, B., S. Zamir. 2004. Leadership with commitment to mixed strategies. Technical Report LSE-CDAM-2004-01, Centre for Discrete and Applicable Mathematics, London School of Economics and Political Science, London.

Wagenaar, W. A. 1972. Generation of random sequences by human subjects: A critical survey of literature. *Psych. Bull.* **77**(1) 65–72.

Wein, L. M. 2008. Homeland security: From mathematical models to policy implementation: The 2008 Philip McCord Morse Lecture. *Oper. Res.* **57**(4) 801–811.

Willis, H. H., A. R. Morral, T. K. Kelly, J. J. Medby. 2005. *Estimating Terrorism Risk*. RAND, Santa Monica, CA.

Erroll G. Southers, Assistant Chief Airport Police, Office of Homeland Security and Intelligence, 9841 Airport Boulevard, Los Angeles, CA 90045, writes: "This letter is to verify that the ARMOR (Assistant for Randomized Monitoring Over Routes) system has indeed been successfully deployed at the Los Angeles International Airport (LAX). It is being utilized by the Los Angeles World Airports (LAWA) Police Division, for randomized scheduling of both vehicle checkpoints on inbound roads inbound to LAX and our explosives detection canine teams. ARMOR is a result of collaboration between LAWA police and researchers at the Department of Homeland Security Center for Risk and Economic Analysis of Terrorism Events (CREATE) at the University of Southern California. In addition to my position as Chief of Homeland Security and Intelligence, I am also the Associate Director of Special Programs at CREATE.

"ARMOR was originally deployed in August 2007 and is now an essential element of our counterterrorism strategy at LAX. The ARMOR system provides numerous benefits which include but are not limited to the following: (a) randomized scheduling to avoid deterministic scheduling policies which can be exploited by vigilant adversaries; (b) appropriately weighted randomization so that certain quality constraints in scheduling may be met; (c) a method for scheduling that is both fast and economical, thus alleviating the scheduling task from LAWA officers; (d) a method of randomization that has the potential for increased security with less resources.

"Since its inception, the ARMOR system has achieved international recognition and helped facilitate the detection and subsequent arrest of persons attempting to bring weapons and narcotics into the airport. In one well documented incident, an individual was arrested, transporting more than 16 weapons, including loaded assault rifles and more than 1,000 rounds of ammunition in his vehicle when identified at our checkpoint.

"ARMOR is living proof of the benefits of an interdisciplinary approach to the evolving threat of terrorism. We continue to seek new and innovative solutions to enhance the protection of one of the regions' most important critical infrastructures."

James B. Curren, Special Assistant, Office of Flight Operations, Federal Air Marshal Service, U.S. Department of Homeland Security, 601 South 12th Street, Arlington, VA 22202-4220, writes: "This letter is to verify that the IRIS (Intelligent Randomization in Scheduling) system is indeed under testing by the Federal Air Marshal Service (FAMS) in assisting with the process of assigning Air Marshals on commercial airline flights. IRIS is the result of collaboration between the FAMS and researchers at the University of Southern California.

"An initial version of IRIS was originally delivered to the FAMS in December 2008 and has since gone through two major updates, the latest of which was delivered in April 2009. The IRIS system provides numerous benefits which include but are not limited to the following: (i) randomized scheduling to avoid deterministic scheduling policies which can be exploited by vigilant adversaries; (ii) appropriately weighted randomization so that certain quality constraints in scheduling may be met; and (iii) a method for scheduling that is both fast and economical, thus reducing the need for manual intervention by FAMS schedulers."