Robust Strategy against Unknown Risk-averse Attackers in Security Games

Yundi Qian
University of Southern California
yundi.qian@usc.edu

William B. Haskell
National University of Singapore
isehwb@nus.edu.sg

Milind Tambe
University of Southern California
tambe@usc.edu

ABSTRACT

Stackelberg security games (SSGs) are now established as a powerful tool in security domains. In this paper, we consider a new dimension of security games: the risk preferences of the attacker. Previous work assumes a risk-neutral attacker that maximizes his expected reward. However, extensive studies show that the attackers in some domains are in fact risk-averse, e.g., terrorist groups in counter-terrorism domains. The failure to incorporate the risk aversion in SSG models may lead the defender to suffer significant losses. Additionally, defenders are uncertain about the degree of attacker's risk aversion. Motivated by this challenge this paper provides the following five contributions: (i) we propose a novel model for security games against risk-averse attackers with uncertainty in the degree of their risk aversion; (ii) we develop an intuitive MIBLP formulation based on previous security games research, but find that it finds locally optimal solutions and is unable to scale up; (iii) based on insights from our MIBLP formulation, we develop our scalable BeRRA algorithm that finds globally ϵ-optimal solutions; (iv) our BeRRA algorithm can also be extended to handle other risk-aware attackers, e.g., risk-seeking attackers; (v) we show that we do not need to consider attacker's risk attitude in zero-sum games.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithm, Security, Robust

Keywords

Risk-averse, Robust Stackelberg equilibrium, Optimization, Security Games.

1. INTRODUCTION

Stackelberg security games (SSGs) are now established as a successful tool in the security domain [10, 3, 12]. In this paper, we focus on a critical dimension of SSGs that has not yet been studied — the risk preferences of the attacker. Previous work on game theory for SSGs emphasizes a risk neutral attacker that is trying to maximize his expected reward. However, if the attacker is not risk-neutral but is actually risk-averse, then the failure to model this risk attitude may lead the defender to suffer significant losses in solution quality.

A major motivating example of our work is the application of security games to the counter-terrorism domain, and there is a thread of work that studies terrorist risk attitudes [19, 18, 20]. In [19], portfolio theory is applied to study a terrorist group's decision making process, and this research argues that terrorist strategies are risk-averse and are highly sensitive to the group's level of risk aversion. While this finding of risk aversion may appear to be counter-intuitive, notice that it is the terrorist groups (and the planners in these groups) that are found to be risk-averse due to resource limitation; not the individuals in the organization who finally launch an attack. [18] studies the risk preferences of Al Qaeda specifically and concludes that the group is risk-averse and consistently displays the same degree of risk aversion in their activities. This work is further extended in [20] where the degree of risk aversion for Al Qaeda is estimated empirically based on data of attacks over the last decade.

Risk aversion encompasses a wide range of behavior — so to say that attackers are risk-averse is not enough for the defender. To address this issue, we compute a robust defender strategy against risk-averse attackers with uncertainty in the degree of risk aversion [1]. In this process, we provide the following contributions in this paper. First, we build a robust SSG framework against an attacker with uncertainty in level of risk aversion. Second, building on previous work on SSGs in mixed-integer programs, we provide a novel mixed-integer bilinear programming problem (MIBLP), and find that it only finds locally optimal solutions. While the MIBLP formulation is also unable to scale up, it provides key intuition for our new algorithm. This new algorithm, BeRRA (**B**inary search based **R**obust algorithm against **R**isk-**A**verse attackers) is our third contribution, and it finds globally ϵ-optimal solutions by solving $\mathcal{O}(n\log(\frac{1}{\epsilon})\log(\frac{1}{\delta}))$ linear feasibility problems. The key idea of our BeRRA algorithm is to reduce the problem from maximizing the reward with a given number of resources to minimizing the number of resources needed to achieve a given reward. This transformation allows BeRRA to scale up via the removal of the bilinear terms and integer variables as well as the utilization of key theoretical properties that prove correspondence of its potential "attack sets" [10] with that of the maximin strategy. Fourth, although the BeRRA algorithm is designed for risk-averse attackers, it also applies to other risk-aware attackers, e.g., risk-seeking attackers. Finally, we also show that we do not need to consider attacker's risk attitude in zero-sum games. Our experimental results show the solution quality and runtime advantages of our robust model and BeRRA algorithm.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

2. MODEL

2.1 Stackelberg Security Games

An SSG [10, 3, 12] is a two-player game between a defender and an attacker. We consider the problem with n targets where $\mathbb{T} = \{1, 2, \ldots, n\}$ is the set of targets. The defender has a total number of m resources to allocate among these n targets to protect them from attack. The defender commits to a mixed strategy $\mathbf{c}$ to protect these targets, where $c_i \in [0, 1]$ is the probability that target i is protected. We have the resource constraint $\sum_{i \in \mathbb{T}} c_i \leq m$. The attacker observes the defender's strategy $\mathbf{c}$ and then chooses one target to attack. If the attacker attacks a protected target i, this attack is unsuccessful and the attacker receives utility[1] $U_a^c(i)$ while the defender receives utility $U_d^c(i)$. If the attacker attacks an unprotected target i, this attack is successful and the attacker receives utility $U_a^u(i)$ while the defender receives utility $U_d^u(i)$. Necessarily, $U_d^u(i) < U_d^c(i)$ and $U_a^c(i) < U_a^u(i), \forall i \in \mathbb{T}$. If $U_d^u(i) + U_a^u(i) = 0$ and $U_d^c(i) + U_a^c(i) = 0, \forall i \in \mathbb{T}$, this SSG is a zero-sum game.

We define $U_a(i, \mathbf{c}) \triangleq c_i U_a^c(i) + (1 - c_i) U_a^u(i)$ to be the expected utility for the attacker when the defender's strategy is $\mathbf{c}$ and the attacker chooses to attack target i; similarly, $U_d(i, \mathbf{c}) \triangleq c_i U_d^c(i) + (1 - c_i) U_d^u(i)$ is the expected utility for the defender. Given the defender strategy $\mathbf{c}$, the attacker would attack the target that maximizes his expected utility. When there are ties, the attacker is assumed to break ties in favor of the defender. Thus, a mixed integer linear program (MILP) can be formulated to compute the defender's optimal strategy, as is shown in Problem (1). Here, $\{q_i\}_{i \in \mathbb{T}}$ are auxiliary variables to represent if target i is chosen by the attacker, and M is a constant orders of magnitude larger than all target utilities. The solution $\mathbf{c}$ is called the Strong Stackelberg Equilibrium (SSE) strategy [7, 21, 11] of the game.

$$
\begin{aligned}
\max_{\mathbf{c}, \{q_i\}_{i \in \mathbb{T}}, v, d} \quad & v \\
s.t. \quad & 0 \leq c_i \leq 1, \forall i \in \mathbb{T} \qquad \sum_{i \in \mathbb{T}} c_i \leq m \\
& q_i \in \{0, 1\}, \forall i \in \mathbb{T} \qquad \sum_{i \in \mathbb{T}} q_i = 1 \\
& v \leq U_d(i, \mathbf{c}) + (1 - q_i) M, \forall i \in \mathbb{T} \\
& 0 \leq d - U_a(i, \mathbf{c}) \leq (1 - q_i) M, \forall i \in \mathbb{T}
\end{aligned}
\tag{1}
$$

2.2 Stackelberg Security Games with Unknown Risk-averse Attackers

The SSE strategy provides the optimal defender strategy when the attacker is risk-neutral. However, as previously discussed, attackers are risk-averse rather than risk-neutral in several key domains. If the defender executes the SSE strategy against a risk-averse attacker, then the defender may suffer significant losses in solution quality. We show in Example 1 that these losses can be arbitrarily large.

EXAMPLE 1. *Suppose there are two targets, t_1 and t_2, in the game, and their utilities are as shown in Table 1. The defender has only* 1 *resource. The SSE strategy of the game is $c_1 = 0.4, c_2 = 0.6$. If the attacker is risk-averse, he would choose to attack t_1 (these two targets are identical to the attacker in terms of expected utility, but a risk-averse attacker prefers a small reward with high probability versus a high reward with low probability), and the defender's reward would be $0.4 + 0.6x$ for the SSE strategy. However, if the defender executes the strategy of $c_1 = 1, c_2 = 0$, then the attacker would attack t_2 and the defender would receive reward -1. Compared with -1, the loss of the SSE strategy can be arbitrarily large since x can be arbitrarily small.*

Table 1: Utility Example

	U_d^c	U_d^u	U_a^c	U_a^u
t_1	1	x	-1	1
t_2	1	-1	-1	2

This example strongly motivates the need to consider risk-averse attackers. However, real world defenders are uncertain about the attacker's degree of risk aversion, and the defender may suffer significant losses if she incorrectly estimates it. Therefore we focus on a robust strategy in this paper, i.e., our aim is to compute a defender strategy that is robust against all possible risk-averse attackers.

In literature on risk, the utility function f, which maps values to utilities, is used to specify the risk preference. f is concave for the risk-averse case and is convex for the risk-seeking case, while the risk-neutral case corresponds to the function $y = Cx, C > 0$. The agent makes decisions based on the mapped utilities.

In our problem, we define the mapping function $\widehat{U}$ that maps the utilities $U_a^c(i)$ and $U_a^u(i)$ to the attacker's mapped utilities. We denote $\widehat{U}_a(i, \mathbf{c}) \triangleq c_i \widehat{U}(U_a^c(i)) + (1 - c_i) \widehat{U}(U_a^u(i))$ as the attacker's expected utility under the mapping $\widehat{U}$. We restrict $\widehat{U}$ to be strictly increasing, concave and satisfying the equality $\widehat{U}(0) = 0$ — strictly increasing reflects the preference for more to less; concavity corresponds to risk aversion; and $\widehat{U}(0) = 0$ distinguishes between gains and losses. According to this definition, the risk-averse case includes the risk-neutral case.

We define $\mathcal{U}$ to be the set of all valid mapping functions $\widehat{U}$. Problem (2) describes the robust defender strategy through a bilevel optimization problem. In the upper level, the defender chooses $\mathbf{c}$ to maximize her expected utility $U_d(k, \mathbf{c})$. The constraint $k \in \arg\max_{i \in \mathbb{T}} \widehat{U}_a(i, \mathbf{c})$ requires target k to have the highest expected utility for the attacker under the utility mapping $\widehat{U}$ when the defender's strategy is $\mathbf{c}$. The lower level demonstrates that the defender maximizes her worst-case reward over all possible attacker responses with utility mapping functions $\widehat{U} \in \mathcal{U}$. The lower level also suggests that the attacker breaks ties against the defender due to the concept of robustness. We define the solution $\mathbf{c}$ to be the Robust Stackelberg Equilibrium (RSE) strategy of the game.

$$
\begin{aligned}
\max_{\mathbf{c}} \min_{\widehat{U} \in \mathcal{U}, k} \quad & \left\{ U_d(k, \mathbf{c}) : k \in \arg\max_{i \in \mathbb{T}} \widehat{U}_a(i, \mathbf{c}) \right\} \\
s.t. \quad & 0 \leq c_i \leq 1, \forall i \in \mathbb{T} \qquad \sum_{i \in \mathbb{T}} c_i \leq m
\end{aligned}
\tag{2}
$$

3. RELATED WORK

Our work is related to handling uncertainty in SSGs. Previous approaches can be divided into two categories: 1) model uncertainty in terms of different attacker types and solve a resulting Bayesian Stackelberg game [17, 24]; 2) apply robust optimization techniques to optimize the worst case for the defender over the range of model uncertainty [23, 9, 15].

Bayesian Stackelberg Game Bayesian Stackelberg game models uncertainty by allowing different attacker types, where there is some prior probability corresponding to each attacker type. Although this method is used to model payoff uncertainty in previous work [17, 24], it can also be used to model different degrees of

[1] Literature on risk denotes $U_a^c(i)/U_d^c(i)$ as values. However, we use the terminology utilities and mapped utilities for consistency with previous work on SSGs.

attacker risk aversion in SSGs. However, this approach requires a prior distribution of attacker types, which is usually inapplicable for many real-world security domains [15]. In addition, it is difficult to apply this approach to infinitely many attacker types. Therefore, we focus on the robust approach in this work.

Robust Approach The robust approach for SSGs, in line with robust optimization, computes a defender strategy that optimizes the worst case over the model uncertainty. Yin et al. [23] computes a defender strategy that is robust against defender execution uncertainty as well as uncertainty in the attacker's observations of the defender's strategy. Kiekintveld et al. [9] focus on interval uncertainty in the attacker's payoffs. Nguyen et al. [15] develop a robust strategy that takes the attacker's bounded rationality into account as well as the uncertainties [23, 9] discuss.

The previous work has addressed neither attacker risk aversion nor ambiguity about the attacker risk profile. Although Kiekintveld et al. [9] try to capture uncertainty in attacker's utilities, they are unable to fully capture the attacker's risk aversion. The mapped utilities are coupled in our problem since they are mapping with the same utility function $\widehat{U}$, and interval uncertainty is unable to model that. For example, Suppose target t_1 is of reward 1 and penalty -2; target t_2 is of reward 2 and penalty -1. The coverage probability $c_1 = c_2 = 0.5$. A risk-averse attacker will always attack t_2 since $\widehat{U}$ must be strictly increasing. However, the model with interval uncertainty 1 would consider both t_1, t_2 to be potential targets for attack. The weakness of the interval uncertainty model is also shown numerically with experiments later.

A third thread of related work is the research in game theory that explores human's bounded rationality in decision making — humans do not necessarily choose the strategy that provides them the highest expected utility [6]. Quantal response [13, 14] argues that human are more likely to choose the strategy with a higher expected utility. Yang et al. [22] apply the concept of quantal response to security games. Nguyen et al. [16] propose the SUQR model by extending the quantal response concept with subjective utilities in security games. However, these approaches do not model risk aversion, and nor do they model uncertainty in risk aversion that we model in this paper. In fact, models such as SUQR essentially address concerns that are orthogonal to the issue of risk aversion; future research may thus consider integrating bounded rationality models with risk aversion.

4. PRELIMINARIES

In its current form, the optimization problem (2) is not tractable because it is a bilevel programming problem that requires the solution of uncountably many inner optimization problems indexed by $\mathcal{U}$ [2]. To take steps towards tractability, in Section 4.1 and 4.2, we provide key concepts that are used in our MIBLP formulation (Section 5) and our BeRRA algorithm (Section 6).

4.1 Risk Aversion Modeling

In this section, we write the condition $\widehat{U} \in \mathcal{U}$ in a computationally tractable way via linear constraints. For any utility function $\widehat{U} \in \mathcal{U}$, we are actually only interested in its values at 0 and at the points of the attacker's utility set $U_a^c(i)$ and $U_a^u(i)$, which we denote as Θ:

$$\Theta = \{U_a^u(i), U_a^c(i), \forall i \in \mathbb{T}\} \bigcup \{0\} = \{\theta_1, \dots, \theta_I\},$$

where $\theta_1 < \theta_2 < \cdots < \theta_I$.

LEMMA 1. *Choose $\epsilon_u > 0$.*[2] *$\widehat{U} \in \mathcal{U}$ is equivalent to satisfying the linear constraints (3) on the values $\left\{\widehat{U}(\theta)\right\}_{\theta\in\Theta}$, i.e., $\forall \widehat{U} \in \mathcal{U}$, $\widehat{U}$ satisfies the constraints (3); $\forall \left\{\widehat{U}'(\theta)\right\}_{\theta\in\Theta}$ that satisfies constraints (3), $\exists \widehat{U} \in \mathcal{U}$ such that $\left\{\widehat{U}(\theta) = \widehat{U}'(\theta)\right\}_{\theta\in\Theta}$.*

$$\begin{aligned} \frac{\widehat{U}(\theta_2) - \widehat{U}(\theta_1)}{\theta_2 - \theta_1} &\geq \frac{\widehat{U}(\theta_3) - \widehat{U}(\theta_2)}{\theta_3 - \theta_2} \\ &\geq \dots \geq \frac{\widehat{U}(\theta_I) - \widehat{U}(\theta_{I-1})}{\theta_I - \theta_{I-1}} \geq \epsilon_u \qquad (3) \\ \widehat{U}(0) &= 0 \end{aligned}$$

PROOF. If $\widehat{U} \in \mathcal{U}$, then $\left\{\widehat{U}(\theta)\right\}_{\theta\in\Theta}$ satisfies constraints (3) by definition. Conversely, if $\left\{\widehat{U}'(\theta)\right\}_{\theta\in\Theta}$ satisfies constraints (3), the piecewise linear function that connects $\{(\theta_1, \widehat{U}'(\theta_1)), (\theta_2, \widehat{U}'(\theta_2))\}$, $\{(\theta_2, \widehat{U}'(\theta_2)), (\theta_3, \widehat{U}'(\theta_3))\}, \dots, \{(\theta_{I-1}, \widehat{U}'(\theta_{I-1})), (\theta_I, \widehat{U}'(\theta_I))\}$ belongs to $\mathcal{U}$. □

Based on Lemma 1, the condition $\widehat{U} \in \mathcal{U}$ is completely captured by constraints (3). From now on we denote the constraints (3) compactly as $\widehat{U} \in \mathcal{U}$.

4.2 Possible Attack Set

In this section, to better understand Problem (2) we study the "possible attack set" $S_p(\mathbf{c})$ and its complement $S_i(\mathbf{c}) = \mathbb{T} - S_p(\mathbf{c})$.

DEFINITION 1. *Given the coverage probability $\mathbf{c}$, Possible Attack Set $S_p(\mathbf{c})$ is defined to be the set of targets that may be attacked by a risk-averse attacker, i.e., it is the set of targets that have the highest expected utility for the attacker for some $\widehat{U} \in \mathcal{U}$.*

$S_i(\mathbf{c}) = \mathbb{T} - S_p(\mathbf{c})$ is defined to be the set of targets that the attacker will never attack, i.e., the set of targets that for any $\widehat{U} \in \mathcal{U}$, there always exists another target $i \in S_p(\mathbf{c})$ with a higher expected utility for the attacker.

Given the coverage probability $\mathbf{c}$, we can compute $S_p(\mathbf{c})$ and $S_i(\mathbf{c})$ by testing the feasibility of the following constraints for every target.

$$\begin{aligned} &\widehat{U}_a(i, \mathbf{c}) \geq \widehat{U}_a(j, \mathbf{c}), \forall j \in T, j \neq i \\ &\widehat{U} \in \mathcal{U} \end{aligned} \qquad (4)$$

If these constraints are feasible for a target i, there exists a mapping $\widehat{U} \in \mathcal{U}$ under which target i has the highest expected utility for the attacker, and thus $i \in S_p(\mathbf{c})$; otherwise, $i \in S_i(\mathbf{c})$.

In Problem (2), when the defender's strategy $\mathbf{c}$ is given, the defender's (worst case) reward is:

$$\min_{\widehat{U}\in\mathcal{U},k} \left\{U_d(k, \mathbf{c}) : k \in \arg\max_{i\in\mathbb{T}} \widehat{U}_a(i, \mathbf{c})\right\}$$

which is equivalent to:

$$\min_{i\in S_p(\mathbf{c})} \{U_d(i, \mathbf{c})\}$$

So Problem (2) can be written as

$$\begin{aligned} &\max_{\mathbf{c}} \min_{i\in S_p(\mathbf{c})} \{U_d(i, \mathbf{c})\} \\ &\quad s.t. \quad 0 \leq c_i \leq 1, \forall i \in \mathbb{T} \qquad \sum_{i\in\mathbb{T}} c_i \leq m \end{aligned} \qquad (5)$$

[2] Since Problem (2) is invariant under scaling of $\widehat{U}$, i.e., the attacker makes the same decision under either $\widehat{U}$ or $\alpha\widehat{U}, \forall\alpha > 0$. Thus, the value of ϵ_u does not affect the result.

5. MIBLP FORMULATION

In this section, we formulate Problem (5) as an MIBLP problem to find the RSE strategy for the defender. While this approach does not scale up to large-scale games, it provides several insights for our BeRRA algorithm. As in Problem (1), we use integer variables $\{q_i\}_{i\in\mathbb{T}}$ to denote if target i belongs to $S_p(\mathbf{c})$: we set $q_i = 1$ if $i \in S_p(\mathbf{c})$ and $q_i = 0$ if $i \in S_i(\mathbf{c})$. Problem (5) can then be converted to the formulation below

$$\begin{aligned}\max_{\mathbf{c}} \quad & v \\ s.t. \quad & 0 \le c_i \le 1, \forall i \in \mathbb{T} \qquad \sum_{i\in\mathbb{T}} c_i \le m \\ & q_i \in \{0,1\}, \forall i \in \mathbb{T} \\ & v \le U_d(i,\mathbf{c}) + (1-q_i)M, \forall i \in \mathbb{T} \\ & i \in S_p(\mathbf{c}) \Leftrightarrow q_i = 1 \\ & i \in S_i(\mathbf{c}) \Leftrightarrow q_i = 0\end{aligned} \tag{6}$$

When $i \in S_p(\mathbf{c})$, constraints (4) are feasible for target i. When $i \in S_i(\mathbf{c})$, for any utility mapping $\widehat{U} \in \mathcal{U}$, there is always another target with a higher expected utility for the attacker. We approximate this strict inequality with a small $\epsilon_c > 0$:

$$\min_{\widehat{U}\in\mathcal{U}} \left\{ \max_{j\in\mathbb{T}} \widehat{U}_a(j,\mathbf{c}) - \widehat{U}_a(i,\mathbf{c}) \right\} \ge \epsilon_c$$

which states that for any $\widehat{U} \in \mathcal{U}$, there exists a target $j \in \mathbb{T}$ whose expected utility for the attacker is at least ϵ_c more than the expected utility for target i. By substituting $\max_{j\in\mathbb{T}} \widehat{U}_a(j,\mathbf{c})$ with the slack variable λ, the preceding bilevel optimization problem can be reduced to:

$$\begin{aligned}\min_{\widehat{U},\lambda} \quad & \lambda - \widehat{U}_a(i,\mathbf{c}) \\ s.t. \quad & \widehat{U}_a(j,\mathbf{c}) \le \lambda, \forall j \in \mathbb{T} \\ & \widehat{U} \in \mathcal{U}\end{aligned} \tag{7}$$

If the solution of Problem (7) is larger than ϵ_c, then $i \in S_i(\mathbf{c})$. Otherwise, $i \in S_p(\mathbf{c})$ (subject to the approximation error introduced by ϵ_c). Since Problem (7) is a minimization problem, it cannot substitute the constraint $i \in S_i(\mathbf{c}) \Leftrightarrow q_i = 0$ in Problem (6). So, we take the Lagrangian dual of Problem (7) to convert it into a maximization problem:

$$\begin{aligned}\max_{\boldsymbol{\alpha},\beta,\boldsymbol{\gamma},\kappa} \quad & \beta\epsilon_u \\ s.t. \quad & \sum_{j\in\mathbb{T}} \gamma_j = 1 \\ & \sum_{k\in\mathbb{T}} \gamma_k c_k \mathbf{1}\{\theta_j = U_a^c(k)\} + \gamma_k(1-c_k)\mathbf{1}\{\theta_j = U_a^u(k)\} \\ & \quad - c_i\mathbf{1}\{\theta_j = U_a^c(i)\} - (1-c_i)\mathbf{1}\{\theta_j = U_a^u(i)\} \\ & \quad + \frac{\alpha_{j-2}\mathbf{1}\{j \ge 3\}}{\theta_j - \theta_{j-1}} - \frac{\alpha_{j-1}\mathbf{1}\{I-1 \ge j \ge 2\}}{\theta_{j+1} - \theta_j} \\ & \quad - \frac{\alpha_{j-1}\mathbf{1}\{I-1 \ge j \ge 2\}}{\theta_j - \theta_{j-1}} + \frac{\alpha_j\mathbf{1}\{j \le I-2\}}{\theta_{j+1} - \theta_j} \\ & \quad - \frac{\beta\mathbf{1}\{j = I\}}{\theta_j - \theta_{j-1}} + \frac{\beta\mathbf{1}\{j = I-1\}}{\theta_{j+1} - \theta_j} \\ & \quad + \kappa\mathbf{1}\{\theta_j = 0\} = 0, \forall j \in \{1,2,\ldots,I\} \\ & \alpha_j \ge 0, \forall j \in \{1,2,\ldots,I-2\} \\ & \beta \ge 0 \\ & \gamma_j \ge 0, \forall j \in \mathbb{T}\end{aligned} \tag{8}$$

For succinct notation, we denote the constraints on the variables $(\boldsymbol{\alpha},\beta,\boldsymbol{\gamma},\kappa)$ in the above formulation as $(\boldsymbol{\alpha},\beta,\boldsymbol{\gamma},\kappa) \in \mathcal{D}$. By applying Problem (4) and Problem (8) to every target i to put constraints on q_i, we summarize our final MIBLP formulation in the next theorem.

THEOREM 1. *Problem (2) is (approximately)*[3] *equivalent to*

$$\begin{aligned}\max \quad & v \\ s.t. \quad & 0 \le c_i \le 1, \forall i \in \mathbb{T} \qquad \sum_{i\in\mathbb{T}} c_i \le m \\ & q_i \in \{0,1\}, \forall i \in \mathbb{T} \\ & v \le U_d(i,\mathbf{c}) + (1-q_i)M, \forall i \in \mathbb{T} \\ & \widehat{U}_a^i(j,\mathbf{c}) \le \widehat{U}_a^i(i,\mathbf{c}) + (1-q_i)M, \forall i \in \mathbb{T}, \forall j \in \mathbb{T}, j \ne i \\ & \widehat{U}^i \in \mathcal{U}, \forall i \in \mathbb{T} \\ & \beta^i\epsilon_u \ge \epsilon_c - q_i M, \forall i \in \mathbb{T} \\ & \left(\boldsymbol{\alpha}^i, \beta^i, \boldsymbol{\gamma}^i, \kappa^i\right) \in \mathcal{D}, \forall i \in \mathbb{T}\end{aligned} \tag{9}$$

where the superscript i in $\left(\boldsymbol{\alpha}^i, \beta^i, \boldsymbol{\gamma}^i, \kappa^i\right)$ and $\widehat{U}^i$ marks different variables. $\widehat{U}_a^i(j,\mathbf{c})$ is attacker's expected utility for target j under the mapping $\widehat{U}^i$ and defender's strategy $\mathbf{c}$.

PROOF. If $q_i = 1$, the constraints $\widehat{U}_a^i(j,\mathbf{c}) \le \widehat{U}_a^i(i,\mathbf{c}) + (1-q_i)M, \forall j \in \mathbb{T}, j \ne i$ and $\widehat{U}^i \in \mathcal{U}$ ensure that $i \in S_p(\mathbf{c})$; if $q_i = 0$, then these constraints are always feasible and can be ignored.

If $q_i = 0$, the constraints $\beta^i\epsilon_u \ge \epsilon_c - q_i M$ and $\left(\boldsymbol{\alpha}^i, \beta^i, \boldsymbol{\gamma}^i, \kappa^i\right) \in \mathcal{D}$ ensure that $i \in S_i(\mathbf{c})$ (approximately) since there exists a solution $\left(\boldsymbol{\alpha}^i, \beta^i, \boldsymbol{\gamma}^i, \kappa^i\right) \in \mathcal{D}$ that satisfies $\beta^i\epsilon_u \ge \epsilon_c$. Thus, the objective of Problem (8) is larger than ϵ_c, and $i \in S_i(\mathbf{c})$. For the other direction, if the objective of Problem (8) is larger than ϵ_c, then these two constraints are also satisfied; if $q_i = 1$, these constraints are always feasible and can be ignored. □

In summary, we have converted Problem (2) into Problem (9), which is an MIBLP: $\{q_i\}_{i\in\mathbb{T}}$ are integer variables; $\widehat{U}_a^i(j,\mathbf{c}) = c_j\widehat{U}^i(U_a^c(j)) + (1-c_j)\widehat{U}^i(U_a^u(j))$ contains bilinear terms since both c_j and $\widehat{U}^i(U_a^c(j))/\widehat{U}^i(U_a^u(j))$ are variables. Problem (9) is a non-convex optimization problem and lacks efficient solvers. We used a powerful nonlinear solver — KNITRO to search for local optimal solutions to Problem (9). However, this approach does not scale up — the two-target scenario takes about 1 minute and the three-target scenario takes about 15 minutes to solve. Faced with this scalability issue, we develop the BeRRA algorithm that finds the ϵ-optimal solution and provides significant scalability.

6. BeRRA ALGORITHM

Problem (9) has two main hindrances to scaling up: the presence of $\Theta(n^2)$ bilinear terms and the presence of n integer variables. Thus, eliminating these bilinear terms and integer variables should allow us to scale the problem up. The bilinear terms in Problem (9) have two components: the coverage probability c_i and the mapped attacker utilities $\widehat{U}(U_a^c(i))/\widehat{U}(U_a^u(i))$. Intuitively, we can avoid the bilinearity by fixing one of these two terms. In addition, if the coverage probability $\mathbf{c}$ is fixed, then $S_p(\mathbf{c})$ is also fixed and we no longer need the integer variables $\{q_i\}_{i\in\mathbb{T}}$ to represent if $i \in S_p(\mathbf{c})$. Based on the idea of fixing the coverage probability $\mathbf{c}$, we develop the BeRRA algorithm. This algorithm computes an ϵ-optimal RSE strategy where ϵ can be made arbitrarily small.

[3] The approximation is due to the introduction of ϵ_c.

The main idea of the BeRRA algorithm is to reduce the problem to computing the minimum amount of resources needed to achieve a given reward, which can be solved efficiently by using special properties of the problem. With this reduction, we use binary search to find the highest reward that the defender can achieve with the given number of resources. The high-level intuition of this reduction is that a fixed defender's reward leads to fixed defender maximin strategy, which eliminates the bilinear terms and integer variables. Additionally, optimal strategy can be derived efficiently from the maximin strategy via the property $S_p(\mathbf{c^{max}}) = S_p(\mathbf{c^{opt}})$.

6.1 Binary Search Reduction

Algorithm 1 lists the steps of our BeRRA algorithm. The input to Algorithm 1 is the number of defender resources m and the defender's and the attacker's utilities $\mathbf{U}$. The output is the defender's RSE strategy $\mathbf{c}$ and her reward lb. The lower bound lb and upper bound ub are first set to be the lowest and the highest possible rewards, respectively, that the defender may achieve (Line 2). The function MinimumResources(r, $\mathbf{U}$) returns the strategy $\mathbf{p}$ that uses the minimum number of resources for the defender to achieve reward r. This function will be discussed in detail in Section 7. During the binary search phase (Lines $3 \sim 11$), the lower bound is set to be the defender's achievable reward (the strategy $\mathbf{p}$ returned by the MinimumResources function is the solution) and the upper bound is set to be an unachievable reward. Therefore, the BeRRA algorithm achieves the ϵ-optimal solution and we can set ϵ arbitrarily small to get arbitrarily near-optimal solutions.

Algorithm 1 BeRRA Algorithm

```
 1: function BeRRA ($m, \mathbf{U}$)
 2:     $lb \leftarrow \min_{i\in\mathbb{T}} U_d^u(i), ub \leftarrow \max_{i\in\mathbb{T}} U_d^c(i)$
 3:     while $ub - lb \geq \epsilon$ do
 4:         $\mathbf{p} \leftarrow$ MinimumResources($\frac{lb+ub}{2}$, $\mathbf{U}$)
 5:         if $\sum_{i\in T} p_i \leq m$ then
 6:             $lb \leftarrow \frac{lb+ub}{2}$
 7:             $\mathbf{c} \leftarrow \mathbf{p}$
 8:         else
 9:             $ub \leftarrow \frac{lb+ub}{2}$
10:         end if
11:     end while
12:     return $(\mathbf{c}, lb)$
13: end function
```

7. MINIMUM RESOURCES

We present Algorithm 2 in this section. This algorithm computes the defender strategy that requires as few resources as possible to achieve a given reward r, i.e., the MinimumResources function in Algorithm 1. We call this resource-minimizing strategy the optimal strategy and denote it as $\mathbf{c^{opt}}$ for succinctness.

Algorithm 2 Minimum Resources

```
1: function MinimumResources($r, \mathbf{U}$)
2:     $(flag, \mathbf{c^{max}}, S_p(\mathbf{c^{max}}), S_i(\mathbf{c^{max}})) \leftarrow$ Maximin($r, \mathbf{U}$)
3:     if $flag = false$ then
4:         return $(\infty, \infty, \ldots, \infty)^\top$
5:     end if
6:     $\mathbf{c^{opt}} \leftarrow$ Reduce($\mathbf{U}, \mathbf{c^{max}}, S_p(\mathbf{c^{max}}), S_i(\mathbf{c^{max}})$)
7:     return $\mathbf{c^{opt}}$
8: end function
```

Algorithm 2 consists of two functions: Maximin and Reduce. The Maximin function computes the maximin strategy $\mathbf{c^{max}}$ for which the defender achieves reward r, as well as the corresponding sets $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$. The variable $flag$ is set to $false$ when the input reward is not achievable for any amount of defender resources. In this case, Algorithm 2 returns $(\infty, \infty, \ldots, \infty)^\top$ (Lines $3 \sim 5$) so that Algorithm 1 knows r is not achievable. We will prove in Theorem 2 that if the reward r is achievable, then $S_p(\mathbf{c^{max}}) = S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c^{max}}) = S_i(\mathbf{c^{opt}})$. Based on this property, the Reduce function derives the optimal strategy $\mathbf{c^{opt}}$ from the maximin strategy $\mathbf{c^{max}}$. Section 7.1 and Section 7.2 discuss these two functions in detail.

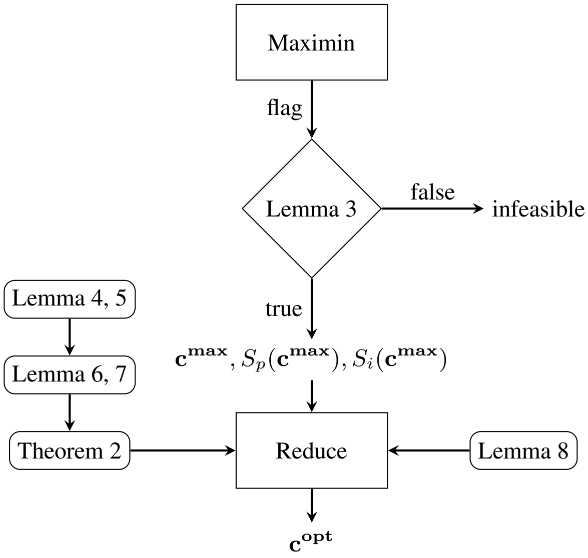


Figure 1: Algorithm 2 and Lemmas (Theorems)

We present a flowchart in Figure 1 that overviews the results we use to establish the correctness of Algorithm 2. Lemma 3 detects if a reward r is achievable or not. Lemmas 4 and 5 help prove Lemmas 6 and 7, which are then used to prove Theorem 2. By Theorem 2 and Lemma 8, the Reduce function is guaranteed to derive the optimal strategy $\mathbf{c^{opt}}$ from the maximin strategy $\mathbf{c^{max}}$.

7.1 Maximin Function

The Maximin function is summarized in Algorithm 3. It first computes the maximin strategy $\mathbf{c^{max}}$ for which the defender achieves reward r (Lines $2 \sim 4$) and then it assigns each target to either $S_p(\mathbf{c^{max}})$ or $S_i(\mathbf{c^{max}})$ (Lines $5 \sim 15$). If the reward r is not achievable for any amount of resources, then it returns $flag = false$ (Line 10).

Lines $2 \sim 4$ compute the maximin strategy for a given reward r. Given a coverage probability $\mathbf{c}$, the maximin setting assumes that the attacker attacks target $i = \arg\min_{j\in\mathbb{T}} U_d(j, \mathbf{c})$, and thus the defender's reward will be $\min_{i\in\mathbb{T}} U_d(i, \mathbf{c})$. For the defender to achieve reward r, we should have $U_d(i, \mathbf{c}) \geq r, \forall i \in \mathbb{T}$ so that $c_i^{max} = \frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}$ (which is bounded by $[0, 1]$).

Given the maximin strategy $\mathbf{c^{max}}$, Lines $5 \sim 15$ iterate through all targets and assign them to either $S_p(\mathbf{c^{max}})$ or $S_i(\mathbf{c^{max}})$ by testing the feasibility of constraints (4). If these constraints are feasible, then $i \in S_p(\mathbf{c})$; otherwise, $i \in S_i(\mathbf{c})$. Next in Lemma 3 we prove that $\exists i \in S_p(\mathbf{c^{max}})$ that satisfies $r > U_d^c(i)$ if and only if reward r is not achievable. In that case, Algorithm 3 returns $flag = false$ (Lines $9 \sim 11$).

Algorithm 3 Maximin

```
function MAXIMIN($r, \mathbf{U}$)
    for $i = 1 \to n$ do
        $c_i^{max} \leftarrow \min\{1, \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}\}$
    end for
    $S_p(\mathbf{c^{max}}), S_i(\mathbf{c^{max}}) \leftarrow \emptyset$
    for $i = 1 \to n$ do
        if Problem (4) is feasible for target $i$ given $\mathbf{c^{max}}$ then
            $S_p(\mathbf{c^{max}}) \leftarrow S_p(\mathbf{c^{max}}) \cup \{i\}$
            if $r > U_d^c(i)$ then
                return $(false, \mathbf{c^{max}}, S_p(\mathbf{c^{max}}), S_i(\mathbf{c^{max}}))$
            end if
        else
            $S_i(\mathbf{c^{max}}) \leftarrow S_i(\mathbf{c^{max}}) \cup \{i\}$
        end if
    end for
    return $(true, \mathbf{c^{max}}, S_p(\mathbf{c^{max}}), S_i(\mathbf{c^{max}}))$
end function
```

LEMMA 2. *Given coverage probability* $\mathbf{c}$*, the defender's reward is* $\min_{i \in S_p(\mathbf{c})} U_d(i, \mathbf{c})$.

PROOF. Follows from the form of problem 5. □

The proof of the following Lemma 3 can be found in the online appendix[4].

LEMMA 3. *Reward* r *is infeasible if and only if Algorithm 3 returns* $flag = false$.

Theorem 2 demonstrates why we compute $\mathbf{c^{max}}$, $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$. We see that $S_p(\mathbf{c^{max}}) = S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c^{max}}) = S_i(\mathbf{c^{opt}})$. Therefore, we get $S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c^{opt}})$ by computing $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$. We introduce supporting lemmas before proving Theorem 2.

The next two lemmas explain how the set $S_p(\mathbf{c})$ changes when the coverage probability for a certain target decreases. The proofs can be found in the online appendix. Lemma 4 shows that if the coverage probability for a target $i \in S_p(\mathbf{c})$ decreases, then the set $S_p(\mathbf{c})$ "shrinks". Lemma 5 shows that if the coverage probability for a target $i \in S_i(\mathbf{c})$ decreases, then the set $S_p(\mathbf{c})$ also "shrinks" but target i might be added to it.

LEMMA 4. *Given coverage probability* $\mathbf{c}$ *and another coverage probability* $\mathbf{c}'$ *which satisfies* $c_i' < c_i$ *for a target* $i \in S_p(\mathbf{c})$ *and* $c_j' = c_j, \forall j \in \mathbb{T}, j \neq i$*, we have* $S_p(\mathbf{c}') \subseteq S_p(\mathbf{c})$.

LEMMA 5. *Given coverage probability* $\mathbf{c}$ *and another coverage probability* $\mathbf{c}'$ *which satisfies* $c_i' < c_i$ *for a target* $i \in S_i(\mathbf{c})$ *and* $c_j' = c_j, \forall j \in \mathbb{T}, j \neq i$*, we have* $S_p(\mathbf{c}') \subseteq S_p(\mathbf{c}) \bigcup \{i\}$.

The next two lemmas discuss key properties of $\mathbf{c^{opt}}$, and the proofs are in the online appendix. Lemma 6 shows that the coverage probability for a target $i \in S_p(\mathbf{c^{opt}})$ must be $\max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$; Lemma 7 shows that the coverage probability for a target $i \in S_i(\mathbf{c^{opt}})$ is at most $\min\{1, \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}\}$. This property is used in the Reduce function that derives $\mathbf{c^{opt}}$ from $\mathbf{c^{max}}$, as well as in the proof of Theorem 2.

LEMMA 6. *Given a feasible reward* r*, all* $i \in S_p(\mathbf{c^{opt}})$ *must satisfy* $U_d^c(i) \geq r$ *and have expected reward* $\max\{U_d^u(i), r\}$ *for the defender, i.e.,* $c_i^{opt} = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}, \forall i \in S_p(\mathbf{c^{opt}})$.

LEMMA 7. *Given a feasible reward* r*,* $\forall i \in S_i(\mathbf{c^{opt}})$*,* i *has expected reward at most* $\min\{U_d^c(i), \max\{U_d^u(i), r\}\}$ *for the defender, i.e.,* $c_i^{opt} \leq \min\{1, \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}\}, \forall i \in S_i(\mathbf{c^{opt}})$.

We are now ready to combine these preliminary lemmas to prove Theorem 2.

THEOREM 2. *Given a feasible reward* r*,* $S_p(\mathbf{c^{max}}) = S_p(\mathbf{c^{opt}})$ *and* $S_i(\mathbf{c^{max}}) = S_i(\mathbf{c^{opt}})$.

PROOF. First we present two results that will be used in the proof: (i) $\forall i \in S_p(\mathbf{c^{max}})$, since the reward r is feasible, Lemma 3 and Algorithm 3 imply that $U_d^c(i) \geq r$ so that $c_i^{max} = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$; (ii) $\forall i \in S_p(\mathbf{c^{opt}})$, according to Lemma 6, $U_d^c(i) \geq r$ and $c_i^{opt} = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$. Furthermore, $U_d^c(i) \geq r$ implies that $c_i^{max} = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$ according to Algorithm 3. Thus, $c_i^{max} = c_i^{opt}, \forall i \in S_p(\mathbf{c^{opt}})$.

We will prove by contradiction that $S_p(\mathbf{c^{opt}}) \subseteq S_p(\mathbf{c^{max}})$. Suppose there exists a target $i \in S_p(\mathbf{c^{opt}})$ with $i \in S_i(\mathbf{c^{max}})$, we have $c_i^{max} = c_i^{opt}$ according to result (ii). Since $i \in S_p(\mathbf{c^{opt}})$, $\widehat{U}_a(i, \mathbf{c^{opt}}) \geq \widehat{U}_a(j, \mathbf{c^{opt}}), \forall j \in \mathbb{T}, j \neq i$ under some mapping $\widehat{U} \in \mathcal{U}$ by definition. Since $i \in S_i(\mathbf{c^{max}})$, for this mapping $\widehat{U}$, $\exists j \neq i, j \in S_p(\mathbf{c^{max}})$ such that $\widehat{U}_a(j, \mathbf{c^{max}}) > \widehat{U}_a(i, \mathbf{c^{max}})$ where $c_j^{max} = \max\{\frac{r - U_d^u(j)}{U_d^c(j) - U_d^u(j)}, 0\}$ according to result (i). So $\widehat{U}_a(j, \mathbf{c^{max}}) > \widehat{U}_a(i, \mathbf{c^{max}}) = \widehat{U}_a(i, \mathbf{c^{opt}}) \geq \widehat{U}_a(j, \mathbf{c^{opt}})$, which leads to $\widehat{U}_a(j, \mathbf{c^{max}}) > \widehat{U}_a(j, \mathbf{c^{opt}})$. Thus, we have $c_j^{opt} > c_j^{max} = \max\{\frac{r - U_d^u(j)}{U_d^c(j) - U_d^u(j)}, 0\}$, which contradicts Lemmas 6 and 7. So, it must be that $i \in S_p(\mathbf{c^{max}})$ which implies $S_p(\mathbf{c^{opt}}) \subseteq S_p(\mathbf{c^{max}})$.

We will prove by contradiction that $S_i(\mathbf{c^{opt}}) \subseteq S_i(\mathbf{c^{max}})$. Suppose there exists a target $i \in S_i(\mathbf{c^{opt}})$ with $i \in S_p(\mathbf{c^{max}})$. We have $c_i^{max} = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$ according to result (i). Since $i \in S_p(\mathbf{c^{max}})$ there exists a mapping $\widehat{U} \in \mathcal{U}$ such that $\widehat{U}_a(i, \mathbf{c^{max}}) \geq \widehat{U}_a(j, \mathbf{c^{max}}), \forall j \in \mathbb{T}, j \neq i$. Since $i \in S_i(\mathbf{c^{opt}})$, for this mapping $\widehat{U}$, $\exists j \neq i, j \in S_p(\mathbf{c^{opt}})$ such that $\widehat{U}_a(j, \mathbf{c^{opt}}) > \widehat{U}_a(i, \mathbf{c^{opt}})$ by definition. We have $c_j^{max} = c_j^{opt}$ according to result (ii). Thus $\widehat{U}_a(i, \mathbf{c^{max}}) \geq \widehat{U}_a(j, \mathbf{c^{max}}) = \widehat{U}_a(j, \mathbf{c^{opt}}) > \widehat{U}_a(i, \mathbf{c^{opt}})$, which yields $\widehat{U}_a(i, \mathbf{c^{max}}) > \widehat{U}_a(i, \mathbf{c^{opt}})$. Then $c_i^{opt} > c_i^{max} = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$, which contradicts Lemma 7. It follows that $i \in S_i(\mathbf{c^{max}})$ which implies $S_i(\mathbf{c^{opt}}) \subseteq S_i(\mathbf{c^{max}})$.

To conclude, $S_p(\mathbf{c^{opt}}) = S_p(\mathbf{c^{max}}), S_i(\mathbf{c^{opt}}) = S_i(\mathbf{c^{max}})$. □

7.2 Reduce Function: Derive $\mathbf{c^{opt}}$ from $\mathbf{c^{max}}$

Section 7.1 demonstrated that Algorithm 2 returns $(\infty, \infty, \ldots, \infty)^\top$ if the reward r is infeasible; if the reward r is feasible, then $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$ are the same as $S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c^{opt}})$. It follows that Algorithm 4 correctly derives the optimal strategy $\mathbf{c^{opt}}$ from $\mathbf{c^{max}}$.

Given $\mathbf{c^{max}}$, $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$, Algorithm 4 returns $c_i^{opt} = c_i^{max}$ for $i \in S_p(\mathbf{c^{max}})$. For $i \in S_i(\mathbf{c^{max}})$, Algorithm 4 uses binary search to find the minimum coverage probability c_i such that any further decrease[5] in coverage probability would add target i to

[4] http://teamcore.usc.edu/people/yundiqia/web%20page/papers/AAMAS2015Appendix.pdf

[5] δ can be arbitrarily small

Algorithm 4 Computing $\mathbf{c^{opt}}$ by reducing $\mathbf{c^{max}}$

```
function REDUCE($\mathbf{U}, \mathbf{c^{max}}, S_p(\mathbf{c^{max}}), S_i(\mathbf{c^{max}})$)
    $\mathbf{c^{opt}} = \mathbf{c^{max}}$
    for every $i \in S_i(\mathbf{c^{max}})$ do
        $lb \leftarrow 0, ub \leftarrow c_i^{opt}$
        while $ub - lb \geq \delta$ do
            $c_i^{opt} \leftarrow \frac{lb+ub}{2}$
            if Problem (4) is feasible for target $i$ given $\mathbf{c^{opt}}$ then
                $lb \leftarrow \frac{lb+ub}{2}$
            else
                $ub \leftarrow \frac{lb+ub}{2}$
            end if
        end while
        $c_i^{opt} \leftarrow ub$
    end for
    return $\mathbf{c^{opt}}$
end function
```

the set $S_p(\mathbf{c^{opt}})$. The next lemma shows that this mechanism leads to the optimal strategy $\mathbf{c^{opt}}$, and the proof can be found in the online appendix.

LEMMA 8. *Given a feasible reward r, Algorithm 4 returns the optimal strategy* $\mathbf{c^{opt}}$.

THEOREM 3. *Given reward r, Algorithm 2 either detects its infeasibility or provides the optimal strategy* $\mathbf{c^{opt}}$.

PROOF. Follows from Lemmas 3 and 8. □

8. DISCUSSIONS

8.1 Computational Cost of BeRRA

The main computational cost of our BeRRA algorithm comes from evaluating the feasibility of the linear constraints (4), which is a linear feasibility problem and can be solved in polynomial time. Algorithm 2 is called $\mathcal{O}(\log(\frac{1}{\epsilon}))$ times, and every call to Algorithm 2 involves solving Problem (4) $\mathcal{O}(n + |S_i(\mathbf{c^{max}})| \log(\frac{1}{\delta}))$ times, which is bounded by $\mathcal{O}(n \log(\frac{1}{\delta}))$. Thus Problem (4) is solved $\mathcal{O}(n \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta}))$ times in total for our BeRRA algorithm.

8.2 Extensions of BeRRA to General Risk Awareness

Notice that we only require $\mathcal{U}$ to be increasing in the preceding proofs and algorithms. Thus, our BeRRA algorithm can also be used to compute the optimal robust strategy against other kinds of risk-aware attackers, e.g., risk-seeking criminals [4, 5, 8]. If the attacker is risk-seeking, $\widehat{U}$ should be a strictly increasing, convex function and satisfies $\widehat{U}(0) = 0$. Therefore, when adapting our BeRRA algorithm to deal with risk-seeking attackers, the only difference is in testing feasibility of constraints (4), where the condition $\widehat{U} \in \mathcal{U}$ in constraints (4) should be written as:

$$\begin{aligned} \epsilon_u \leq \frac{\widehat{U}(\theta_2) - \widehat{U}(\theta_1)}{\theta_2 - \theta_1} &\leq \frac{\widehat{U}(\theta_3) - \widehat{U}(\theta_2)}{\theta_3 - \theta_2} \\ &\leq \ldots \leq \frac{\widehat{U}(\theta_I) - \widehat{U}(\theta_{I-1})}{\theta_I - \theta_{I-1}} \qquad (10) \\ \widehat{U}(0) &= 0 \end{aligned}$$

8.3 Zero-sum Game

When the game is a zero-sum game, the utilities for the defender and the attacker are strongly correlated with correlation coefficient -1, i.e., $U_a^c(i) = -U_d^c(i)$ and $U_a^u(i) = -U_d^u(i), \forall i \in \mathbb{T}$. Based on this property, we obtain the following theorem.

THEOREM 4. *For zero-sum games, the defender's Robust Stackelberg Equilibrium (RSE) strategy and Maximin strategy are the same.*

PROOF. We first prove that given a defender's strategy $\mathbf{c}$, the defender's reward is the same in both settings.

Given the defender's strategy $\mathbf{c}$, the defender's reward in the Maximin setting is $\min_{j \in \mathbb{T}} U_d(j, \mathbf{c})$, which is a lower bound on the defender's reward in the Robust Stackelberg game setting since $\min_{j \in \mathbb{T}} U_d(j, \mathbf{c})$ is the minimum reward the defender can possibly achieve with $\mathbf{c}$. Meanwhile, since the risk-neutral case is a special case of the risk-averse case, $i = \arg\min_{j \in \mathbb{T}} U_d(j, \mathbf{c}) = \arg\max_{j \in \mathbb{T}} U_a(j, \mathbf{c}) \in S_p(\mathbf{c})$. Thus, the attacker might attack target i so that the expected reward the defender achieves if the attacker attacks target i — $\min_{j \in \mathbb{T}} U_d(j, \mathbf{c})$ is also an upper bound on the defender's reward in the Robust Stackelberg game setting. Therefore, the defender's reward in the Robust Stackelberg game setting is $\min_{j \in \mathbb{T}} U_d(j, \mathbf{c})$.

Since the defender's reward given the defender's strategy $\mathbf{c}$ is the same in both settings, the strategy $\mathbf{c}$ that maximizes the defender's reward in both settings is also the same. □

It is known that the solution concepts of Nash Equilibrium, minimax, maximin, and SSE all give the same answer for finite two-person zero-sum games. Therefore, Theorem 4 adds RSE to this equivalence list.

9. EXPERIMENTAL EVALUATION

We will evaluate the performance of our algorithms in this section through extensive numerical experiments. Unless otherwise stated, all of the experiment results are averaged over 20 instances. $U_d^c(i)$ and $U_a^u(i)$ are generated as random variables between 11 and 40; $U_d^u(i)$ and $U_a^c(i)$ are generated as random variables between -11 and -40. To generate payoff matrixes with correlation between the defender's and the attacker's utilities, we set $U_a^c(i) \leftarrow -\alpha U_d^c(i) + \sqrt{1-\alpha^2} U_a^c(i)$ and $U_a^u(i) \leftarrow -\alpha U_d^u(i) + \sqrt{1-\alpha^2} U_a^u(i)$, where $-\alpha$ is the correlation coefficient between $U_a^c(i)(U_a^u(u))$ and $U_d^c(i)(U_d^u(i))$. $\alpha = 1$ corresponds to zero-sum games. n is the number of targets in the game and m is the number of resources the defender has.

9.1 MIBLP vs BeRRA

We first compare the performance of the MIBLP formulation and our BeRRA algorithm. Due to the scalability of the MIBLP, we only compare the case when $n = 2$ and $n = 3$. m is set to be 1. The KNITRO solver is used to solve the MIBLP formulation.

MIBLP vs BeRRA: Solution Quality The solution quality of the MIBLP formulation and our BeRRA algorithm is shown in Table 2. We can see from the table that BeRRA algorithm has a higher average reward compared with MIBLP, and the difference becomes larger as the number of targets n increases. This is because KNITRO can only find the locally optimal solution while our BeRRA algorithm finds the globally ϵ-optimal solution, and larger game scale leads to worse solution quality of the local optimum.

MIBLP vs BeRRA: Runtime The runtime of the MIBLP formulation and our BeRRA algorithm is shown in Table 3. We can see from the table that BeRRA is much faster than MIBLP, and the difference becomes larger as the number of targets n increases.

Table 2: MIBLP vs BeRRA in Solution Quality

(a) $n = 2$

	MIBLP	BeRRA
$\alpha = 0$	3.18	3.41
$\alpha = 0.2$	2.78	2.99
$\alpha = 0.4$	2.45	2.62
$\alpha = 0.6$	1.72	1.82
$\alpha = 0.8$	0.75	0.81
$\alpha = 1.0$	0.53	0.53

(b) $n = 3$

	MIBLP	BeRRA
$\alpha = 0$	-5.69	-4.60
$\alpha = 0.2$	-5.71	-5.32
$\alpha = 0.4$	-6.75	-5.84
$\alpha = 0.6$	-6.92	-6.31
$\alpha = 0.8$	-7.24	-6.96
$\alpha = 1.0$	-7.64	-7.64

This is because solving MIBLP is difficult and the computational complexity increases exponentially with the problem size, while BeRRA only requires solving $\mathcal{O}(n \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta}))$ linear feasibility problems. For MIBLP, it takes about 15 minutes for the very trivial case $n = 3$, which means it cannot scale up at all.

Table 3: MIBLP vs BeRRA in Runtime (s)

(a) $n = 2$

	MIBLP	BeRRA
$\alpha = 0$	70.4	0.95
$\alpha = 0.2$	71.2	0.95
$\alpha = 0.4$	72.0	0.94
$\alpha = 0.6$	68.9	0.77
$\alpha = 0.8$	73.4	0.48
$\alpha = 1.0$	64.4	0.20

(b) $n = 3$

	MIBLP	BeRRA
$\alpha = 0$	863.1	1.75
$\alpha = 0.2$	1004.4	1.63
$\alpha = 0.4$	958.8	1.53
$\alpha = 0.6$	886.9	1.36
$\alpha = 0.8$	1119.8	1.10
$\alpha = 1.0$	859.3	0.27

Runtime of BeRRA Figure 2 further analyzes the runtime of our BeRRA algorithm. m is set to be $n/2$ and all results are averaged over 100 instances. We observe that the runtime increases almost linearly with the number of targets n, and the game with 50 targets only takes about 2 minutes to solve, which demonstrates BeRRA's ability to scale up to larger problems. The figure also shows that the runtime does not change significantly with different α, but it decreases significantly when α is increased from 0.9999 to 1. This is because $|S_i(\mathbf{c}^{\mathbf{max}})|$ is almost 0 in zero-sum games.

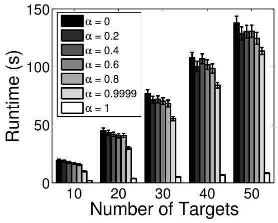


Figure 2: Runtime of BeRRA

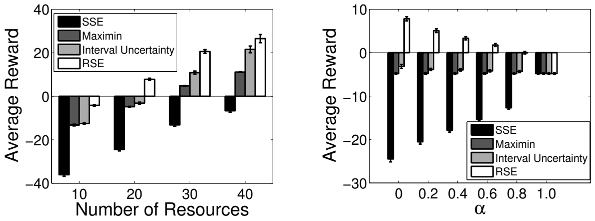
9.2 Performance Evaluation of RSE strategy

In this section, we will evaluate solution quality of the RSE strategy in detail. Since BeRRA shows advantages in both solution quality and runtime compared with the MIBLP formulation, we use BeRRA to evaluate the performance of RSE strategy.

Solution Quality in Worst Case Figures 3(a) and 3(b) show the solution quality of RSE strategy in the worst case — the attacker attacks target $i = \arg\min_{j \in S_p(\mathbf{c})} U_d(j, \mathbf{c})$. We compare its performance with the SSE strategy, Maximin strategy and the robust strategy against interval uncertainty of $U_a^c(i)$ and $U_a^u(i)$ [9]. For values of the intervals, we tried different intervals ranging from 1 to 20 and pick the best one among them.

Figure 3(a) shows how the performance comparison changes with different number of resources m. The RSE strategy significantly outperforms all of the other strategies. Since the robust strategy against interval uncertainty considers some type of "robustness", it outperforms the SSE strategy and the Maximin strategy. However, since the interval uncertainty does not fully capture the risk aversion of the attacker, it is worse than the RSE strategy. The Maximin strategy is a more conservative strategy compared with the SSE strategy, leading to better performance when compared with SSE.

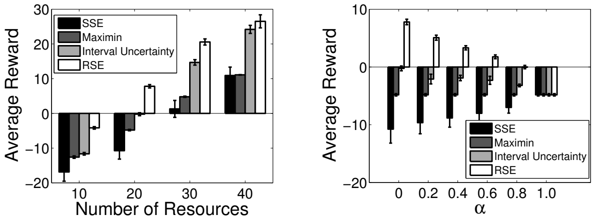
Figure 3(b) shows the performance comparison with different α. It shows the similar patten that RSE > Interval Uncertainty > Maximin > SSE as in Figure 3(a). Another observation is that the difference between these four strategies becomes less as α increases, and when $\alpha = 1$, these four strategies perform the same, as is proved in Theorem 4[6].



(a) $\alpha = 0, n = 50$ (b) $m = 20, n = 50$

Figure 3: Solution Quality of RSE in Worst Case

Solution Quality in Average Case Figures 4(a) and 4(b) show the solution quality of the RSE strategy in the average case — the attacker randomly attacks a target i in $S_p(\mathbf{c})$. We explore this case since unknown risk-averse attackers in the real world would not necessarily minimize the defender's reward. The performance comparison of these four strategies in the average case shows similar patterns compared with that in the worst case. Thus even in the average case, the RSE strategy still performs the best among them.



(a) $\alpha = 0, n = 50$ (b) $m = 20, n = 50$

Figure 4: Solution Quality of RSE in Average Case

10. CONCLUSION

This paper presents a model and algorithm to compute robust defender strategy in security games against risk-averse attackers with uncertainty in the degree of risk aversion. We find that the intuitive MIBLP formulation only finds locally optimal solutions and is unable to scale up. Inspired by the MIBLP formulation, we develop our BeRRA algorithm which finds globally ϵ-optimal solutions by solving $\mathcal{O}(n \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta}))$ linear feasibility problems. The key idea of our BeRRA algorithm is to reduce the problem to minimizing the number of resources needed for a given reward, which can be solved efficiently using special properties of the problem. Although the BeRRA algorithm is designed for risk-averse attackers, it also applies to other risk-aware attackers, e.g., risk-seeking attackers. In addition, we also show that we do not need to consider the attacker's risk attitude in zero-sum games.

11. ACKNOWLEDGEMENT

This research is supported by the United States Department of Homeland Security through the National Center for Risk and Economic Analysis of Terrorism Events (CREATE) under award number 2010-ST-061-RE0001 and MURI grant W911NF-11-1-0332.

[6] Interval Uncertainty = Maximin can be proved with similar techniques in the proof of Theorem 4.

REFERENCES

[1] M. Aghassi and D. Bertsimas. Robust game theory. *Mathematical Programming*, 2006.

[2] J. F. Bard. *Practical bilevel optimization: algorithms and applications*. Springer, 1998.

[3] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, 2009.

[4] G. S. Becker. Crime and punishment: an economic approach. *Journal of political economy*, 1968.

[5] M. K. Block and V. E. Gerety. Some experimental evidence on differences between student and prisoner reactions to monetary penalties and risk. *The Journal of Legal Studies*, 1995.

[6] C. Camerer. *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press, 2003.

[7] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *EC*, 2006.

[8] J. Grogger. Certainty vs. severity of punishment. *Economic Inquiry*, 1991.

[9] C. Kiekintveld, T. Islam, and V. Kreinovich. Security games with interval uncertainty. In *AAMAS*, 2013.

[10] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, 2009.

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

[12] J. Letchford and Y. Vorobeychik. Computing randomized security strategies in networked domains. In *Applied Adversarial Reasoning and Risk Modeling*, 2011.

[13] R. D. McKelvey and T. R. Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 1995.

[14] R. D. McKelvey and T. R. Palfrey. Quantal response equilibria for extensive form games. *Experimental economics*, 1998.

[15] T. Nguyen, A. Jiang, and M. Tambe. Stop the compartmentalization: Unified robust algorithms for handling uncertainties in security games. In *AAMAS*, 2014.

[16] T. H. Nguyen, R. Yang, A. Azaria, S. Kraus, and M. Tambe. Analyzing the effectiveness of adversary modeling in security games. In *AAAI*, 2013.

[17] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games with security: An efficient exact algorithm for bayesian stackelberg games. In *AAMAS*, 2008.

[18] P. Phillips. The preferred risk habitat of al-qa'ida terrorists. *European Journal of Economics, Finance and Administrative Sciences*, 2010.

[19] P. J. Phillips. Applying modern portfolio theory to the analysis of terrorism. computing the set of attack method combinations from which the rational terrorist group will choose in order to maximise injuries and fatalities. *Defence and Peace Economics*, 2009.

[20] P. J. Phillips. The end of al-qa'ida: rationality, survivability and risk aversion. *International Journal of Economic Sciences*, 2013.

[21] B. Von Stengel and S. Zamir. Leadership with commitment to mixed strategies. 2004.

[22] R. Yang, C. Kiekintveld, F. Ordonez, M. Tambe, and R. John. Improving resource allocation strategy against human adversaries in security games. In *IJCAI*, 2011.

[23] Z. Yin, M. Jain, M. Tambe, and F. Ordonez. Risk-averse strategies for security games with execution and observational uncertainty. In *AAAI*, 2011.

[24] Z. Yin and M. Tambe. A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *AAMAS*, 2012.