Applied Adversarial Reasoning and Risk Modeling: Papers from the 2011 AAAI Workshop (WS-11-06)

Computing Randomized Security Strategies in Networked Domains

Joshua Letchford[*]
Department of Computer Science
Duke University
Durham, NC, 27708
jcl@cs.duke.edu

Yevgeniy Vorobeychik
Sandia National Laboratories
Livermore, CA 94551-0969
eug.vorobey@gmail.com

Abstract

Traditionally, security decisions have been made without explicitly accounting for adaptive, intelligent attackers. Recent game theoretic security models have explicitly included attacker response in computing randomized security policies. Techniques to date, however, generally fail to explicitly account for interdependence between the targets to be secured, which is of vital importance in a variety of domains, including cyber, supply chain, and critical infrastructure security. We introduce a novel framework for computing optimal randomized security policies in networked domains which extends previous approaches in two ways. First, we extend previous linear programming techniques for Stackelberg security games to incorporate benefits and costs of arbitrary security configurations on individual assets. Second, we offer a principled model of failure cascades that allows us to capture both the direct and indirect value of assets. Finally, we use our framework to analyze four models, two based on random graph generation models, a simple model of interdependence between critical infrastructure and key resource sectors, and a model of the Fedwire interbank payment network.

1 Introduction

Game theoretic approaches to security have received much attention in recent years. Most have attempted to distill various aspects of the problem into a model that could then be solved in closed form (see, for example, a recent survey of game theoretic techniques applied to network security (Roy et al. 2010)). Numerous others, however, offer techniques based on mathematical programming to solve actual instances of security problems. Approaches to network interdiction (Cormican, Morton, and Wood 1998), for example, offer (usually) an integer programming formulation solving for an location of sensors that optimally interdict traffic (such as drug traffic) through a network. The point of departure of our work is a different line of work that develops linear and integer programming methods for optimal randomized allocation of security resources among possible attack targets. In this work, the assumption is made that the defender is a Stackelberg leader, that is, he is able to commit to a randomized policy, which is subsequently observed by the attacker who optimally responds to it. Initial work on the subject offered an approach relying on multiple linear programs to compute such an optimal *commitment* strategy in general two-player games (Conitzer and Sandholm 2006). Follow-up work focused on integer programming methods for Bayesian security settings (Paruchuri 2008), and much has attempted to exploit the special structure of security scenarios to build faster algorithms (Kiekintveld et al. 2009), with some of these finding use in actual security applications such as ARMOR for the allocation of canine patrols in LAX and IRIS for the scheduling of federal air marshalls (Jain et al. 2010).

In this paper we introduce a novel framework for computing optimal randomized security policies in networked domains. First, in Section 2 we give some background on how security scenarios have previously been modeled as Stackelberg games. We then propose an extension to this line of work, where rather than having a hard constraint on the number of defense resources we have a soft constraint in the form of costs. In Section 3 we extend previous linear programming techniques for Stackelberg security games to incorporate benefits and costs of arbitrary security configurations on individual assets. In Section 4 we offer a principled model of failure cascades that allows us to capture both the direct and indirect value of assets. In Section 5 we illustrate our model with a simple supply chain example. Finally, in Section 6 we use our model to study security decisions in four types of networks, two based on models of random graph generation, a simple model of interdependence between critical infrastructure and key resource sectors and a model of the Fedwire interbank payment network.

2 Stackelberg Security Games

A Stackelberg security game (Kiekintveld et al. 2009) consists of two players, the leader (defender) and the follower (attacker), and a set of possible targets. The leader can decide upon a randomized policy of defending the targets, possibly with limited defense resources; we say that the leader thereby *commits to a mixed strategy*. The follower (attacker) is assumed to observe the randomized policy of the leader, but not the realized defense actions. Upon observing the leader's strategy, the follower chooses a target so as to maximize its expected utility.

[*]This paper represents work done while at Sandia National Laboratories

In past work, Stackelberg security game formulations focused on defense policies that were costless, but resource bounded. Specifically, it had been assumed that the defender has K fixed resources available with which to cover (subsets of) targets. Additionally, security decisions amounted to covering a set of targets, or not. While in numerous settings to which such work has been applied (e.g., airport security, federal air marshall scheduling) this formulation is very reasonable, in other settings one may choose among many *security configurations* for each valued asset, and, additionally, security resources are only available at some cost. For example, in cybersecurity, protecting computing nodes could involve setting anti-virus and/or firewall configuration settings, with stronger settings carrying a benefit of better protection, but at a cost of added inconvenience, lost productivity, as well as possible licensing costs. Indeed, costs on resources may usefully take place of resource constraints, since such constraints are often not hard, but rather channel an implicit cost of adding further resources.

To formalize, suppose that the defender can choose from a finite set O of security configurations for each target $t \in T$, with $|T| = n$. A configuration $o \in O$ for target $t \in T$ incurs a cost $c_{o,t}$ to the defender. If the attacker happens to attack t while configuration o is in place, the expected value to the defender is denoted by $U_{o,t}$, while the attacker's value is $V_{o,t}$. A key assumption in Stackelberg security games is that the targets are completely independent: that is, a joint defender and attacker decision concerning one target has no impact on the values of others, and total defender and attacker utilities are additive over all targets. We revisit this assumption below when we turn to networked (and general interdependent) settings. We denote by $q_{o,t}$ the probability that the defender chooses o at target t, while a_t denotes the probability that the attacker attacks target t.

3 Computing Optimal Randomized Security Configurations

Kiekintveld et al. (2009) previously introduced the ERASER algorithm, which is a Mixed Integer Programming (MIP) formulation for computing optimal randomized security policies in Stackelberg security games. We first show how to extend this MIP formulation to arbitrary security configurations, as well as to incorporate costs of such configurations. We then proceed to offer an alternative formulation involving multiple linear programs (in the same vein as the initial multiple-LP formulation for general Stackelberg games (Conitzer and Sandholm 2006)) and, finally, formulate this problem as a single linear program.

3.1 Mixed Integer Programming Formulation

The MIP formulation is shown in Equations 1- 7. Equations 2 and 3 force the attack vector to attack a single target with probability 1. This captures the well-known observation that a deterministic choice of the best of n targets to attack is optimal for the attacker (Kiekintveld et al. 2009). Equations 4 and 5 force the configuration decision for each target to be a valid probability distribution over O. In Equations 6 and 7, Z is some constant which is larger than the highest utility achievable in the game. Consequently, Equation 6 will only bind when $a_t = 1$. This, combined with the fact that u is maximized in the objective, implies that in an optimal solution, $u = \sum_o U_{o,t} \cdot q_{o,t}$ for the target that is attacked. Consequently, u corresponds to the optimal expected utility of the defender.

The right-hand-side of Equation 7 will similarly only bind when $a_t = 1$. Since v is forced by the left-hand-side of Equation 7 to be at least the expected attacker utility from attacking any target, and must be exactly equal to the expected utility of the attacked target, it must therefore be the attacker's optimal expected utility given the defender's mixed strategy commitment q.

$$\max \quad u - \sum_t \sum_o c_{o,t} \cdot q_{o,t} \tag{1}$$

s.t.

$$\forall_t \quad a_t \in \{0,1\} \tag{2}$$

$$\sum_t a_t = 1 \tag{3}$$

$$\forall_{o,t} \quad q_{o,t} \in [0,1] \tag{4}$$

$$\forall_t \quad \sum_o q_{o,t} = 1 \tag{5}$$

$$\forall_t \quad u - \sum_o U_{o,t} \cdot q_{o,t} \le (1 - a_t) \cdot Z \tag{6}$$

$$\forall_t \quad 0 \le v - \sum_o V_{o,t} \cdot q_{o,t} \le (1 - a_t) \cdot Z \tag{7}$$

3.2 Multiple-LP Formulation

In reformulating the MIP above as a collection of LPs, we note that the lone integer vector a in the MIP formulation does not have combinatorial structure. Rather, it chooses a single target from n possibilities. Considering each of these possibilities for attack separately then yields n linear programs, and the defender can simply choose the solution with the highest expected value (u) as the optimal mixed strategy commitment. The resulting formulation as n LPs is shown in Equations 8-11.

$$\forall_{\hat{t}} \ \max \quad \sum_o U_{o,\hat{t}} q^{\hat{t}}_{o,\hat{t}} - \sum_t \sum_o c_{o,t} q^{\hat{t}}_{o,t} \tag{8}$$

s.t.

$$\forall_{o,t} \quad q^{\hat{t}}_{o,t} \in [0,1] \tag{9}$$

$$\forall_t \quad \sum_o q^{\hat{t}}_{o,t} = 1 \tag{10}$$

$$\forall_t \quad \sum_o V_{o,t} q^{\hat{t}}_{o,t} \le \sum_o V_{o,\hat{t}} q^{\hat{t}}_{o,\hat{t}} \tag{11}$$

Notice that since the potential target is identified in each of n linear programs, we are able to compress the set of constraints, removing the now-redundant variables u and v. Each program now has a clean interpretation, just as in the original multiple-LP formulation due to Conitzer and Sandholm: for each target, we force the attacker to prefer that target over all others. The intuition behind this is that in an optimal solution, the attacker must (weakly) prefer to attack some target, and consequently, one of these LPs must correspond to an optimal defense policy.

3.3 Single Linear Program Formulation

Starting with the multiple-LP formulation above, it is now not difficult to construct just a single LP that aggregates all of these. We cannot do so immediately, however, because some of the n LPs may actually be infeasible: some targets may not be optimal for the attacker for any defense policy. Consequently, we must prune out all such targets in order to ensure that the combined LP is feasible. Formally, it suffices to check, for each target $\hat{t}$ that

$$\max_o V_{o,\hat{t}} \geq \max_t \min_o V_{o,t}, \tag{12}$$

that is, that $\hat{t}$ is not strictly dominated for the attacker. Let $\hat{T} \subset T$ be the set of targets for which Equation 12 holds. The aggregate single-LP formulation is then shown in Equations 13-16.

$$\max \quad \sum_{\hat{t}\in\hat{T}} \left(\sum_o U_{o,\hat{t}} q^{\hat{t}}_{o,\hat{t}} - \sum_t \sum_o c_{o,t} q^{\hat{t}}_{o,t} \right) \tag{13}$$

s.t.

$$\forall_{\hat{t},o,t} \qquad q^{\hat{t}}_{o,t} \in [0,1] \tag{14}$$

$$\forall_{\hat{t},t} \qquad \sum_o q^{\hat{t}}_{o,t} = 1 \tag{15}$$

$$\forall_{\hat{t},t} \qquad \sum_o V_{o,t} q^{\hat{t}}_{o,t} \leq \sum_o V_{o,\hat{t}} q^{\hat{t}}_{o,\hat{t}} \tag{16}$$

Notice that we can easily incorporate additional linear constraints in any of these formulations. For example, it is often useful to add a budget constraint:

$$\forall_{\hat{t},t} \quad \sum_o c_{o,t} q^{\hat{t}}_{o,t} \leq B.$$

Below we utilize this single-LP formulation, without the budget constraint.

4 Incorporating Network Structure

Thus far, a key assumption has been that the utility of the defender and the attacker for each target depends only on the defense configuration for that target, as well as whether it is attacked or not. In many domains, such as cybersecurity and supply chain security, key assets are fundamentally interdependent, with an attack on one target having potential consequences for others. In this section, we show how to transform certain classes of problems with interdependent assets into a formulation in which targets become effectively independent, for the purposes of our solution techniques.

To begin, let $w_{o,t}(a)$ be an *intrinsic worth* of a target t to the defender when it is protected by a security configuration o and the attacker employs a probability vector a with a'_t specifying the probability of attacking target t'. Suppose that the attacker chooses to attack a target t (and only t). Further, assume that the defender and attacker utilities are additive in target-specific worths. The expected utility $U_{o,t}$ is then

$$U_{o,t} = E[\sum_{t'} w_{o',t'}(a_t = 1)] = \sum_{t'} E[w_{o',t'}(a_t = 1)],$$

where o' denotes the defense configuration at target t'. From this expression, it is apparent that in general, $U_{o,t}$ depends on defense configurations at other targets, and therefore targets cannot be readily decoupled. However, under the following assumption, we recover target independence:

Assumption 1. *For all t and $t' \neq t$, $w_{o',t'}(a_t = 1) = w_{t'}(a_t = 1)$.*

One way to interpret this assumption is that once a particular target is compromised, the fault may spread to others which depend on it even if these assets are very well protected. This hearkens back to the line of research on interdependent security (Kunreuther and Heal 2003) where this assumption was operational. One example of such interdependence given by Kunreuther and Heal (2003) is airline baggage screening: baggage that is transferred between airlines is rarely thoroughly screened, perhaps due to the expense. Thus, even while an airline may have very strong screening policies, it is poorly protected from luggage entering its planes via transfers. Cybersecurity has similar shortcomings: defense is often focused on external threats, with little attention paid to threats coming from computers internal to the network. Thus, once a computer on a network is compromised, the attacker may find it much easier to compromise others on the same network. Use of common operating environments exacerbates that further: once an exploit is found, it can often be reused to compromise other computing resources on a common network.

Under the assumption above,

$$U_{o,t} = E[w_{o,t}(a_t = 1)] + \sum_{t' \neq t} E[w_{t'}(a_t = 1)],$$

and thus $U_{o,t}$ for all t do not depend on defense configurations at other targets t'. By a similar argument and an analogous assumption for the attacker's utility, we recover complete target independence required by the linear programming formulations above.

In general, one may use an arbitrary model to compute or estimate $E[w_{o',t'}(a_t = 1)]$ above. Indeed, often simulation tools are available to perform the analysis of global consequences of attacks on particular pieces of the infrastructure (Dudenhoeffer, Permann, and Manic 2006). Nevertheless, we offer a specific model of interdependence between targets that is rather natural and applies across a wide variety of settings. Suppose that dependencies between targets are represented by a graph (T, E), with T the set of targets (nodes) as above, and E the set of edges (t, t'), where an edge from t to t' (or an undirected edge between them) means that target t' depends on target t (and, thus, a successful attack on t may have impact on t'). Each target has associated with it a worth, w_t as above, although in this context this worth is incurred only if t is affected (compromised, affected by a flaw that spreads from one of its dependencies, etc). The security configuration determines the probability $z_{o,t}$ that target t is compromised (affected) if the attacker attacks it *directly* and the defense configuration is o. We model the interdependencies between the nodes as independent cascade contagion, which has previously been used primarily to model diffusion of product adoption and infectious disease (Kempe, Kleinberg, and Éva Tardos 2003; Dodds and Watts 2005). The contagion proceeds starting at an attacked node t, affecting its network neighbors t' each with probability $p_{t,t'}$. The contagion can only occur once along any network edge (that is, the biased coin is only flipped once), and once a node is affected, it stays affected through the diffusion process. The simple way to conceive of this is to start with the network (T, E) and then remove each edge (t, t') with probability $(1 - p_{t,t'})$. The entire connected component of an attacked node is then deemed affected. The entire framework is illustrated in Figure 1.

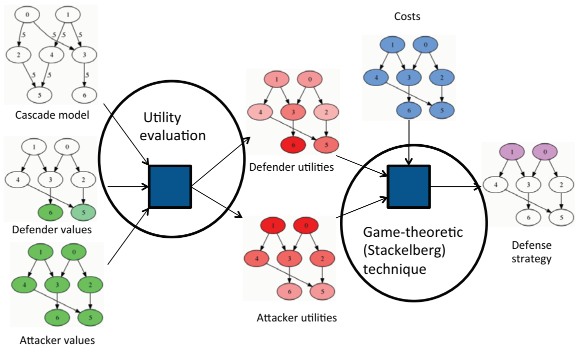

Figure 1: Illustration of the framework for computing optimal randomized defense strategies in networked domains.

4.1 Computing Expected Utilities

Given the independent cascade model of interdependencies between targets, we must compute expected utilities, $U_{o,t}$ and $V_{o,t}$, of the defender and the attacker respectively. In general, we can do so by simulating cascades starting at every node t, with expected utilities estimated as sample average utilities over K simulated cascades (expectation in this case is with respect to random realizations of attack success for specific targets as well as edges that become a part of the failure contagion. In the special case when the network is a tree, however, we can show that expected utilities can be computed exactly in linear time.

Theorem 1. *Given an undirected tree (T, E), an assigned worth for each node (w_t), and a probability of spread over each edge $p_{t,t'}$, we can calculate the expected utility at each target in $O(n)$.*

Proof. Let us define the neighbors of a target t as N_t. By definition, the expected utility of a given node t ($U_{o,t}$) is the direct utility at that node ($z_{o,t}w_t$) plus the expected utility due to the cascading failure. The expected utility due to cascading failure is $z_{o,t} \sum_{t' \neq t} w_{t'} p(failure(t')|a_t = 1)$. Since this is a tree, there is only one path between any pair of nodes, which means we can express $p(failure(t')|a_t = 1)$ as the product of probabilities of the edges on the path between t and t'. Next, let us consider the set of paths generated by each pair of nodes in the tree. If we organize these paths by the edges they contain (and use linearity of expectation), we can express the expected utility of the contagion spreading across an edge (t, t'), $E[U_{(t,t')}]$, as:

$$E[U_{(t,t')}] = p_{t,t'} \left(w_{t'} + \sum_{t'' \in N_{t'}, t'' \neq t} E[U_{(t',t'')}] \right). \quad (17)$$

Thus, we can reason that for each node t:

$$U_{o,t} = z_{o,t} U_t,$$

where

$$U_t = w_t + \sum_{t' \in N_t} E[U_{(t,t')}] \quad (18)$$

Now let us describe a two-pass algorithm for calculating U_t for all t. First, choose an arbitrary node to be the root of the tree. In the first pass, we calculate the expected loss due to each edge from parent to child from the bottom of the tree upward. In the second pass, we calculate the expected loss on each edge from child to parent from the top of the tree downward. We can model this as a message passing algorithm, where calculating $E[U_{(t,t')}]$ is done by passing a message from t' to t. We can see by Equation 17 that the necessary inputs to calculate $E[U_{(t,t')}]$ are the messages from $N_{t'} \setminus t$ to t'. We will now show that at the time that each of these messages is generated, all of the necessary inputs will be available.

Consider the edges between a given node t and its neighbors. Unless t is the root, one of these edges will be between t and its parent P_t, and the rest (possibly 0 in the case where t is a leaf node) will be between t and its children. Since in the first pass we are passing messages from child to parent and a node has only one parent, we will have received messages from $N_t \setminus P_t$ when we generate the message from t to P_t.

For the second pass, when we pass information to a child of t, C_t, we will have received messages from N_t, thus we again have the necessary information to generate the message from t to C_t.

Finally, once a node has received messages from all of its neighbors we can easily calculate the expected loss at each node by Equation 18. However, to achieve a runtime of $O(n)$, we need to be slightly more clever in how we store these values. By combining Equations 17 and 18 we can reason that $E[U_{(t,t')}] = p_{t,t'}(U_{t'} - E[U_{(t',t)}])$. This allows us to give an equivalent definition of U_t:

$$U_t = w_t + \sum_{t \in N_t} p_{t,t'}(U_{t'} - E[U_{(t',t)}]). \quad (19)$$

Now, consider the same two-pass algorithm as before, but rather than storing the expected loss for every edge, we merely store a running total of the expected loss at each node. We argue that by the same reasoning as before that the necessary calculations will have been performed before we need them as inputs. However, we still need to show that

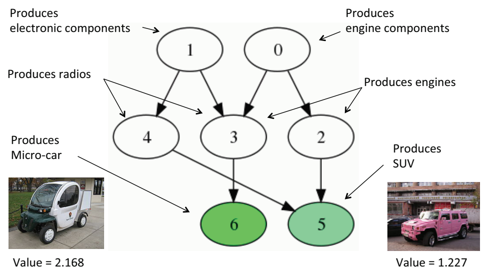

Figure 2: A simple automotive supply chain example.

we can recover the correct value out of the values stored at the two nodes. When we calculate $E[U_{(P,C)}]$ in the first pass, the value stored at C will be $(U_C - E[U_{(C,P)}])$, since we have not yet updated C with $E[U_{(C,P)}]$. However, when we reach this edge on the downward pass to to calculate $E[U_{(C,P)}]$, P will have U_P stored. Since the value stored at C is still $(U_C - E[U_{(C,P)}])$, we can easily calculate $E[U_{(C,P)}] = p_{C,P}(U_P - E[U_{(P,C)}]) = p_{C,P}(U_P - p_{P,C}(U_C - E[U_{(C,P)}]))$ and update C.

Since we visit each edge twice, and perform a constant amount of work each time, we can bound the runtime by $O(|E|)$. Since in a tree $|T| - 1 = |E|$, we can also bound the runtime by $O(n)$. □

5 Supply Chain Example

In this section we illustrate the tools introduced above with a simple example. Consider a seven-node supply chain (directed acyclic graph) shown in Figure 2. We suppose that the entire supply chain (or at least the relevant security decisions) is controlled by a single firm which is primarily concerned with manufacturing two types of cars, one more profitable than the other. The actual components that ultimately comprise the cars are not intrinsically valuable to the manufacturer (or are valued so low relative to the final product as to make them effectively not important in this decision). All parts of the supply chain may be inspected at some cost c, or not (in which case no cost is incurred).

The first step in our framework is to compute (or estimate) the expected utility for each node in the supply chain. To do this, we first specify the probability that an attacked node is affected (in this case, becomes faulty), $z_{o,t}$. We let $z_{o,t} = 1$ when node t is not inspected and $z_{o,t} = 0$ when it is. Next, we must specify the contagion probabilities for each edge. We use $p_{t,t'} = 0.5$ for all edges here. The results are color coded in Figure 3 (left): the darker colors correspond to more valuable nodes. Note that while intrinsic worth is only ascribed to the final products, all components carry some value, due to their indirect impact on the final product (for example, a faulty part will, with some probability, make the component which uses it faulty as well). Supposing now that the game is zero-sum, the expected utilities of the attacker are completely determined by the defender's utilities, and we can use these as an input into the linear programs above. We show the results for two different inspection costs, $c_{high} = 0.1428$ and $c_{low} = 0.0179$ in

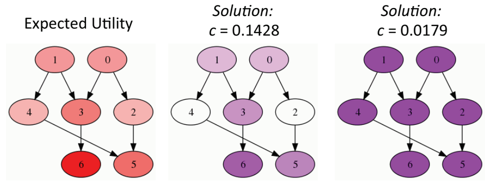

Figure 3: Randomized defense configurations for the simple supply chain example under two defense cost scenarios.

Figure 3. The higher cost setting (Figure 3, middle) yields a security configuration in which five of the seven nodes incur some probability of inspection, with the heavier colors corresponding to high inspection probability. The low-cost setting (Figure 3, right) yields a solution in which every node is defended with probability 1.

6 Experimental results

In this section we apply our framework to several networked domains. First, we consider networks generated from two major generative random graph models: Erdos-Renyi and Preferential Attachment (Newman 2010). In addition, we explore two networks derived from real security settings: one that models dependencies between critical infrastructure and key resource sectors, as inferred from the DHS and FEMA websites, and another that captures payments between banks in the core of the Fedwire network.

For the randomly generated networks, all data presented is averaged over 100 graph samples. Since we generate graphs that may include undirected cycles, we obtain expected utilities for all nodes using 10,000 simulated cascades. (We later revisit this issue and show that this is more than a sufficient number for problems of this scale). In this section we assume that the defender has two strategies at every target, one with a cost of 0 which stops attacks 0% of the time and one with a cost of c which prevents attacks 100% of the time, and, additionally, that we have a zero-sum game between the defender and attacker.

Our first set of results pertain to undirected Erdos-Renyi ($G(n, p_e)$) networks, with 100 nodes (n) and edge probability p_e ranging from 0.0025 to 0.08 (average degree between .25 and 8). We assign each node a worth drawn uniformly at random from $[0, 1]$ and assign each edge a cascade probability of .5. Figure 4 shows how the average cost and expected loss (plus their sum which we will refer to as total loss) varies when we vary c for the case where $p_e = .02$ (average degree of 2).

This plot shows two phase shifts: one at a defense cost of approximately 0.01 and the second near a defense cost of 0.03. For costs below 0.01, the total loss is entirely due to cost, as the optimal defense strategy is to defend all of the targets 100% of the time. For $0.01 \leq c \leq 0.03$, the amount spent on defense is still rising, but the optimal defense policy no longer protected each node with probability 1. As a result, the expected loss becomes non-zero in this cost regime. When $c > 0.03$, the total amount spent on defense starts to fall (even though average costs are still rising), while the

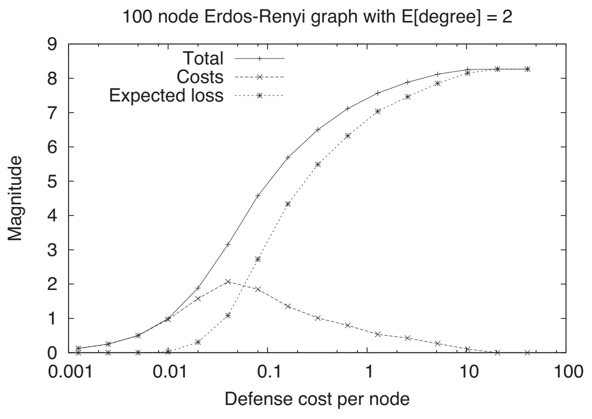

Figure 4: Expected loss, cost, and their sum in 100-node Erdos-Renyi networks as defense cost increases.

expected loss continues to rise. Finally, when c reaches approximately 20, the amount spent on defense becomes 0, and there is no subsequent change to either the expected loss or defense spending.

For the second set of results, we generated undirected Preferential Attachment networks using the Barabási-Albert model, with $n = 100$ and with the degree of new nodes ranging from 1 to 4 (average degree between 2 and 8). We used a 4-node open chain as a seed graph. As before, we assign each node a worth drawn uniformly at random from $[0, 1]$ and each edge has a cascade probability of .5. Figure 5 shows how the average cost, expected loss and total loss vary with c for the case where the initial degree of new nodes is 1 (average degree approximately 2).

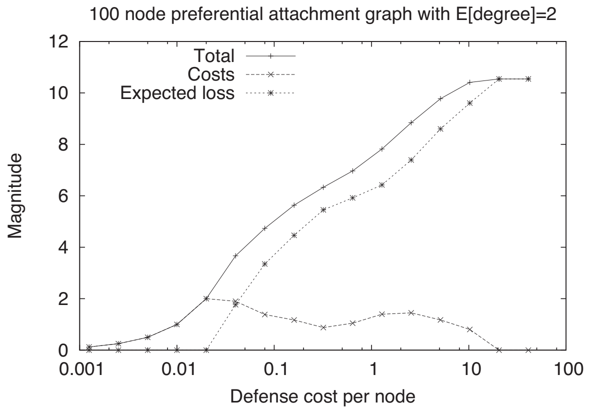

Figure 5: Expected loss, cost, and their sum in 100-node Preferential Attachment networks as defense cost increases.

Unsurprisingly, there again appears to be a point at which 100% defense is no longer reasonable, near $c = 0.03$. Perhaps more interesting is the fact that the amount spent on defense starts to decrease at this point, in contrast with the Erdos-Renyi networks. It is also notable that the amount spent on defense doesn't monotonically decrease for $c > 0.03$, but actually shows a slight increase when $0.32 \leq c \leq 2.56$. If we look at individual instances here, we find that the probability of defending targets is still falling within this range, but not nearly as quickly as the cost of defense is rising, which causes the overall increase in defense spending. Just as in Erdos-Renyi networks, once defense cost reaches 20, there is are no further expenditures on defense.

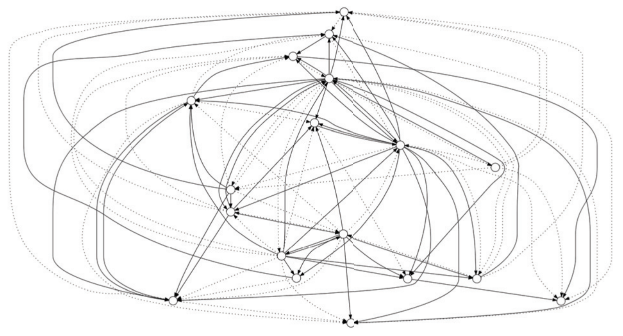

Figure 6: Critical Infrastructure network.

Next, we generated a model of the dependencies of the US critical infrastructure and key resource sectors based on the information on the DHS and FEMA websites. This model (pictured in Figure 6) contains 18 nodes and 94 directed edges. We assigned each node a worth in the $[0, 1]$ range, but only limited actual values to three possibilities: very important ($w_t = 1$), somewhat important ($w_t = 0.5$), and relatively unimportant ($w_t = 0.2$). We also classified importance of each dependency into two categories: very significant (corresponding to the probability of contagion $p_{t,t'} = 0.5$) and less significant (with $p_{t,t'} = 0.1$). Figure 7 shows how the average cost and expected loss (plus their sum) varies when we vary c, giving a now-familiar pattern. Note that while the number of nodes here is only 18, by the time $c = 1$, expected total loss already approaches the value close to 100-node Erdos-Renyi and Preferential Attachment networks. The average (per-node) loss is, thus, much higher, primarily because the critical infrastructure and key resource network is substantially more dense.

The last network model we used is the topology of the core of the Fedwire Interbank Payment Network (Soramaki et al. 2007) shown in Figure 8. This model contains 66 nodes and 181 undirected edges. In this case we simulate a policymaker with an impartial interest in keeping the system afloat. To reflect this, we assigned each node a worth of .5. We were able to infer a rough estimate of the relative magnitude of the worth of the payments that occurred between the various banks, which we used to assign failure probabilities between .05 for low volume edges and .5 for high volume edges. Figure 9 shows how the average cost and expected loss (plus their sum) varies when we vary c. Interestingly, the expected loss and cost follow here a very similar pattern to that we observed for Preferential Attachment networks.

Figure 10 shows a comparison between the total loss per node (scaled total loss) for three of these four models. We omit the critical infrastructure model because edge density and different average worth cause its scaled total loss to be

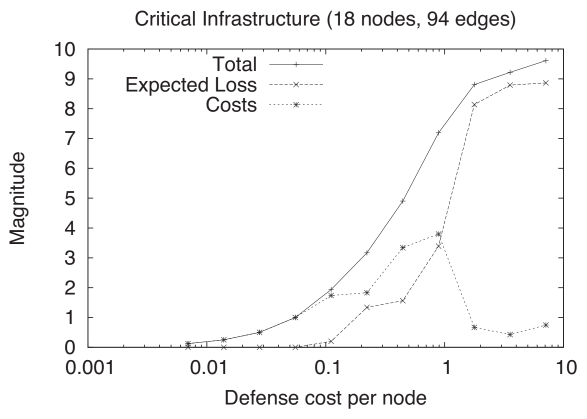

Figure 7: Expected loss, cost, and their sum in the critical infrastructure network as defense cost increases.

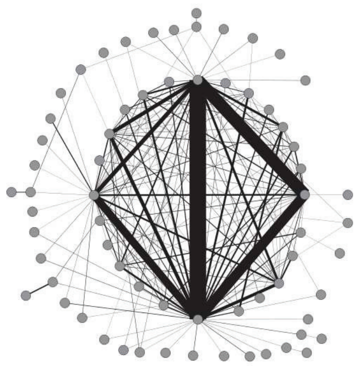

Figure 8: The core of the Fedwire Interbank Payment Network (based on Soramaki et al. (2007)).

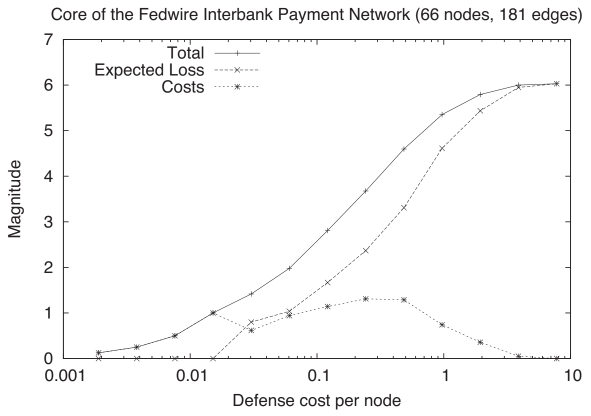

Figure 9: Expected loss, cost, and their sum in the core of the Fedwire network as defense cost increases.

far higher than the other three models. We can observe that the total loss in all of these models not only follows the same general S-curve pattern but actually seems to follow approximately the same path. If we restrict our attention to just the two randomly generated models, we can see that with a few small exceptions, these two models are almost identical until a defense cost reaches 1. However, for $c \geq 10$, all of these models have flattened out at the steady state, and the differences here are entirely due to the differences in maximum expected loss for each model (based in large part on differences in topology).

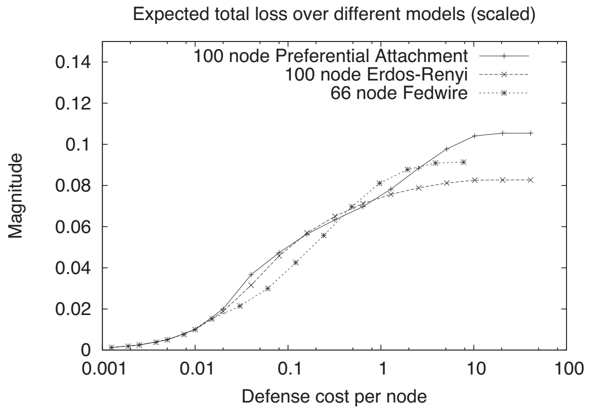

Figure 10: Expected total loss: comparison across different network structures.

So far we assumed that the information about network structure is known. A complementary question is what effect does incomplete information about network structure has on the quality of decisions. We address this question in the context of Erdos-Renyi networks by noting that the ultimate realization of the network over which cascades propagate remains Erdos-Renyi. We can thus study the effect of incomplete information by taking a fixed network, and comparing decisions based on a limited number of contagion samples to "ground truth", which we define to be the result after 100,000 samples. What we obtain can also be used as a guideline for the effect of sampling accuracy in the actual solver. The results are shown in Figure 11. While the solutions found with an extremely small number of samples do rather poorly, as we increase the number of samples we quickly converge upon solutions that do almost exactly as well as the "ground truth". Thus, the decisions are relatively robust to small amounts of noise in estimating the true network structure in Erdos-Renyi networks.

Finally, let us consider how the cost of defense, the expected loss, and total loss change as we increase the average degree from .25 to 8 (increasing network density from 0.005 to 0.16) for a fixed defense cost of $.04$. Intuitively, as the network density rises, so does the threat of large contagion. From the results in Figure 12 we can see two consequences of this. First, the total loss increases as the average degree increases. This follows from the fact that as the degree increases, also does the expected loss from an undefended attack. Furthermore, at lower degrees, a large fraction of the total loss is from the expected loss, as defense is not cost efficient when an attack is unlikely to spread to a significant fraction of the graph. However, as the degree increases,

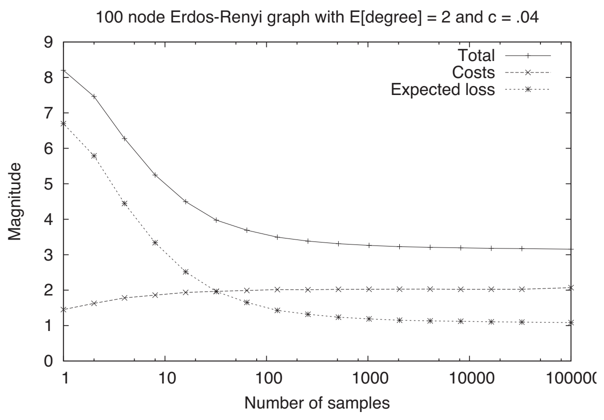

Figure 11: Expected loss, cost, and the sum under incomplete information about network structure.

there is a clear increase in the fraction of the total loss due to defense costs, eventually reaching the point of 100% when average degree is between 2 and 4.

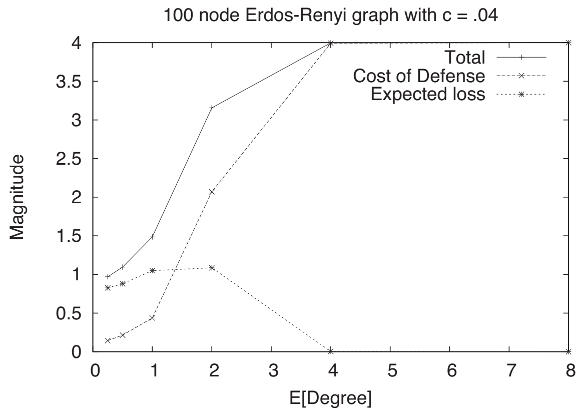

Figure 12: Expected loss, cost, and their sum in 100-node Erdos-Renyi networks as a function of network density (equivalently, expected degree).

7 Conclusion

We present a framework for computing optimal randomized security policies in network domains, extending previous linear programming approaches to Stackelberg security games in two ways. First, we construct a single linear programming formulation which incorporates costs of arbitrary security configurations and may be extended to enforce budget constraints. Second, we demonstrate how to transform a general setting with interdependent assets into a security game with independent targets, allowing us to leverage the compact linear programming formulation for security games. We apply our framework to study four models of interdependent security. Two are based on standard generative models of random graphs, and two others use real networks representing interdependent assets. We show that there is a surprising bi-modal behavior of expected utility in preferential attachment networks as defense costs increase, that such networks lead to greater expected losses than Erdos-Renyi networks when defense costs are high. We also demonstrate that increased network density has substantial deleterious effect on expected losses of targeted attacks and that, as a result, highly interdependent networks such as that representing critical infrastructure sector dependence are extremely difficult to defend. Finally, we show that having some information, even very limited, about the true network structure can be of substantial value in guiding high quality defense decisions.

Acknowledgement

Sandia is a multiprogram laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy under contract DE-AC04-94AL85000.

References

Conitzer, V., and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, EC ’06, 82–90. New York, NY, USA: ACM.

Cormican, K. J.; Morton, D. P.; and Wood, R. K. 1998. Stochastic network interdiction. *Operations Research* 46(2):184–197.

Dodds, P. S., and Watts, D. J. 2005. A generalized model of social and biological contagion. *Journal of Theoretical Biology* 232:587–604.

Dudenhoeffer, D. D.; Permann, M. R.; and Manic, M. 2006. CIMS: A framework for infrastructure interdependency modeling and analysis. In Perrone, L.; Wieland, F.; Liu, J.; Lawson, B.; Nicol, D.; and Fujimoto, R., eds., *Proceedings of the 2006 Winter Simulation Conference*, 478–485.

Jain, M.; Tsai, J.; Pita, J.; Kiekintveld, C.; Rathi, S.; Tambe, M.; and Ordóñez, F. 2010. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces* 40:267–290.

Kempe, D.; Kleinberg, J. M.; and Éva Tardos. 2003. Maximizing the spread of influence in a social network. In *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 137–146.

Kiekintveld, C.; Jain, M.; Tsai, J.; Pita, J.; Ordez, O.; and Tambe, M. 2009. Computing optimal randomized resource allocations for massive security games. In *In AAMAS-09*.

Kunreuther, H., and Heal, G. 2003. Interdependent security. *Journal of Risk and Uncertainty* 26(2-3):231–249.

Newman, M. 2010. *Networks: An Introduction*. Oxford University Press.

Paruchuri, P. 2008. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *In AAMAS*.

Roy, S.; Ellis, C.; Shiva, S.; Dasgupta, D.; Shandilya, V.; and Wu, Q. 2010. A survey of game theory as applied to network security. In *Forty-Third Hawaii International Conference on System Sciences*, 1–10.

Soramaki, K.; Bech, M. L.; Arnold, J.; Glass, R. J.; and Beyeler, W. 2007. The topology of interbank payment flows. *Physica A* 379:317–333.