Audit Games with Multiple Defender Resources*

Jeremiah Blocki[1], **Nicolas Christin**[1], **Anupam Datta**[1], **Ariel D. Procaccia**[1], **Arunesh Sinha**[2]
[1]Carnegie Mellon University, USA; {arielpro@cs., jblocki@cs., danupam@, nicolasc@}cmu.edu
[2]University of Southern California, USA; aruneshs@usc.edu

Abstract

Modern organizations (e.g., hospitals, social networks, government agencies) rely heavily on audit to detect and punish insiders who inappropriately access and disclose confidential information. Recent work on *audit games* models the strategic interaction between an auditor with a single audit resource and auditees as a Stackelberg game, augmenting associated well-studied security games with a configurable punishment parameter. We significantly generalize this audit game model to account for multiple audit resources where each resource is restricted to audit a subset of all potential violations, thus enabling application to practical auditing scenarios. We provide an FPTAS that computes an approximately optimal solution to the resulting non-convex optimization problem. The main technical novelty is in the design and correctness proof of an optimization transformation that enables the construction of this FPTAS. In addition, we experimentally demonstrate that this transformation significantly speeds up computation of solutions for a class of audit games and security games.

1 Introduction

Modern organizations (e.g., hospitals, banks, social networks, search engines) that hold large volumes of personal information rely heavily on auditing for privacy protection. These audit mechanisms combine automated methods with human input to detect and punish violators. Since human audit resources are limited, and often not sufficient to investigate all potential violations, current state-of-the-art audit tools provide heuristics to guide human effort (Fairwarning 2011). However, numerous reports of privacy breaches caused by malicious insiders bring to question the effectiveness of these audit mechanisms (Ponemon Institute 2011; 2012).

Recent work on *audit games* by Blocki et al. (2013) approaches a piece of this problem using game-theoretic techniques. Their thesis is that effective human audit resource allocation and punishment levels can be efficiently computed by modeling the audit process as a game between an auditor and auditees. At a technical level, their audit game model augments a well-studied Stackelberg security games model (Tambe 2011) with a configurable punishment parameter. The auditor (henceforth called the *defender*) can audit *one* of n potential violations (referred to as *targets*). The defender's optimal strategy is a randomized auditing policy—a distribution over targets such that when the attacker best responds the defender's utility is maximized. The novel ingredient of the audit games model is a punishment level, chosen by the defender, which specifies how severely the adversary will be punished if he is caught. The defender may try to set a high punishment level in order to deter the adversary. However, punishment is not free. The defender incurs a cost for punishing, e.g. punishments such as suspension or firing of violators require maintaining resources for hiring and training of replacements. Blocki et al. (2013) provide an efficient algorithm for computing an optimal strategy for the defender to commit to.

While this work distills the essence of the defender's dilemma in auditing situations, it is too restricted to inform real-world audit strategies. In typical audit settings, the defender has *multiple resources* using which she can audit a subset of the targets (not just one target). Furthermore, each resource may be *restricted* in the targets that it can audit. For example, some organizations follow hierarchical audit strategies in which a manager is only required to audit potential violations committed by her direct reports. Similarly, specialized audits are common, for example, employing disjoint audit resources to detect finance-related and privacy-related violations.

Our Contributions. We present a generalized Stackelberg audit game model that accounts for multiple audit resources where each resource is restricted to audit a subset of all potential violations, thus enabling application to the practical auditing scenarios described above. Our main theoretical result is a *fully polynomial time approximation scheme (FPTAS)* to compute an approximate solution to the resulting

*This work was partially supported by the U.S. Army Research Office contract Perpetually Available and Secure Information Systems (DAAD19-02-1-0389) to Carnegie Mellon CyLab, the NSF Science and Technology Center TRUST, the NSF CyberTrust grant Privacy, Compliance and Information Risk in Complex Organizational Processes, the AFOSR MURI Collaborative Policies and Assured Information Sharing,, HHS Grant no. HHS 90TR0003/01 and NSF CCF-1215883. Jeremiah Blocki was also partially supported by a NSF Graduate Fellowship. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

non-convex optimization problem.

To arrive at this FPTAS, we begin with a simple *fixed parameter tractable (FPT) algorithm* that reduces the non-convex optimization problem to a linear programming problem by fixing the punishment parameter at a discrete value. Since we perform a linear search over all possible discrete values of the punishment parameter over a fixed range in increasing intervals of size ϵ, we get an FPT algorithm once the bit precision is fixed.

Next we present an *optimization transformation* that reduces the number of variables in the optimization problem at the cost of generating additional constraints. We also provide sufficient conditions that guarantee that the number of constraints is polynomial in size. Significantly, these conditions are satisfied in important practical auditing scenarios, including the hierarchical and specialized audit settings discussed earlier. The design and correctness proof of this optimization transformation constitutes the central novel technical contribution of this paper. Finally, we present an FPTAS to compute the defender's strategy leveraging the output of the optimization transformation when it generates a polynomial number of constraints.

In addition to its role in enabling the design of an FPTAS, a practical motivation for designing the optimization transformation is its promise of speeding up computing game solutions using the FPT algorithm. We experimentally demonstrate that the transformation produces speedups of up to $3\times$ for audit game instances and over $100\times$ for associated security game instances. In general, the speedups are higher as the problem size increases.

As an additional contribution, we consider audit scenarios where the defender can set a different punishment level for each target instead of setting a single, universal punishment level. We provide an FPT algorithm for this problem by designing a novel reduction of this setting to a second order cone program (SOCP).

Related Work. Our work is most closely related to the paper of Blocki et al (2013). We elaborate on the technical connections in the exposition and analysis of our results.

Our work is also closely related to work on security games (Tambe 2011). The basic approach to solving security games, and Stackelberg games more generally (where a leader chooses an optimal strategy, i.e., one that maximizes its utility assuming that the follower best responds) was introduced by Conitzer and Sandholm (2006); it does not scale well when the number of defender strategies is exponentially large, as is the case in most security games. Typical algorithms rely on formulating a mixed integer linear program, and employing heuristics; these algorithms do not provide provable running-time guarantees (see, e.g., Paruchuri et al. (2009); Kiekintveld et al. (2009)). Often the problem is solved just for the coverage probabilities or marginals (probability of defending a target) with the hope that it would be implementable (i.e., decomposable into a valid distribution over allocations). In contrast, Korzhyk et al. (2010) give a polynomial time algorithm for security games where the attacker has multiple resources, each of which can only protect one target at a time (this restriction is known as *singleton schedules*). Their main tool is the Birkhoff-Von Neumann Theorem, which we also apply.

All our algorithms have theoretical results about their efficiency. Our optimization transformation transforms the problem to one with only coverage probabilities as variables and adds additional constraints for the coverage probabilities that restricts feasible coverage probabilities to be implementable. Thus, in contrast with Korzhyk et al. (2010) our problem has much fewer variables at the cost of additional constraints. Also, the punishment parameter makes our problem non-convex, and so our algorithms must leverage a suite of additional techniques.

2 The Audit Games Model

An audit game features two players: the defender (D), and the adversary (A). The defender wants to audit n targets $t_1, \ldots, t_n$, but has limited resources which allow for auditing only some of the n targets. Concretely, these resources could include time spent by human auditors, computational resources used for audit (e.g., automated log analysis), hardware (e.g., cameras) placed in strategic locations, among many other examples. The exact nature of these resources will depend on the specific audit problem considered. Rather than focusing on a specific type of audit, we denote resources available to the defender for audit as *inspection resources*. The defender has k *inspection resources* $\{s_1, \ldots, s_k\}$ at her disposal, with $k < n$. Each inspection resource can be used to audit at most one target. Inspection resources are further constrained by the set of targets that they can audit: for instance, a human auditor may not have the expertise or authority to audit certain targets. We define a set of tuples R such that a tuple $(j, i) \in R$ indicates that inspection resource s_j cannot be used to audit target t_i.

A pure action of the defender chooses the allocation of inspection resources to targets. A randomized strategy is given by a probability distribution over pure actions. We use a compact form to represent a randomized strategy as a matrix of probabilities, with p_i^j the probability of inspection resource s_j auditing target t_i subject to the following constraints

$$p_i = \textstyle\sum_{j=1}^{k} p_i^j \leq 1, \quad \sum_{i=1}^{n} p_i^j \leq 1 \quad \text{for all } i, j \text{ and}$$
$$p_i^j = 0 \text{ for all } (j,i) \in R \text{ and } \forall (j,i).\ p_i^j \geq 0\ ,$$

where p_i denotes the probability that target t_i is inspected. These constraints can be represented as *grid constraints* as follows:

	t_1	...		t_n	
s_1	p_1^1	...	...	p_n^1	$\sum_i p_i^1 \leq 1$
...	...	...	...	...	...
...	...	...	...	...	...
s_k	p_1^k	...	...	p_n^k	$\sum_i p_i^k \leq 1$
	$\sum_j p_1^j \leq 1$		...	$\sum_j p_n^j \leq 1$	

Such a matrix can be decomposed into pure actions efficiently. (See the full version for the Birkhoff-von Neumann (1946) result presented in Korzhyk et al. (2010), which enables the decomposition.) Furthermore, for every distribution over pure actions we can define an "equivalent" strategy using our compact representation: the distributions

will be equivalent in the sense that for every target t_i the probability p_i that that target is inspected is the same.

Similarly to the basic audit games model of Blocki et al. (2013), the defender also chooses a punishment "rate" $x \in [0, 1]$ such that if auditing detects an attack (i.e., violation), the attacker is fined an amount x. The adversary attacks one target such that given the defender's strategy the adversary's choice of attack is the best response.

We follow the notations of earlier papers on security games: let the utility of the defender be $U_D^a(t_i)$ when audited target t_i was found to be attacked, and $U_D^u(t_i)$ when unaudited target t_i was found to be attacked. We assume that the attacks on unaudited targets are discovered by an external source. Similarly, define the utility of the attacker as $U_A^a(t_i)$ when the attacked target t_i is audited, and $U_A^u(t_i)$ when attacked target t_i is not audited, excluding any punishment imposed by the defender. Attacks discovered externally are costly for the defender, thus, $U_D^a(t_i) \geq U_D^u(t_i)$. Similarly, attacks not discovered by internal audits are more beneficial to the attacker, thus, $U_A^u(t_i) \geq U_A^a(t_i)$.

We can now define the full utility functions. The probabilities $p_1, \ldots, p_n$ of each target being audited are given by $p_i = \sum_{j=1}^{k} p_i^j$. Then, the utility of the defender when target t_* is attacked is $p_* U_D^a(t_*) + (1 - p_*) U_D^u(t_*) - ax$. The defender pays a fixed cost ax regardless of the outcome, where a is a constant. The losses captured by the $-ax$ term includes loss due to creation of a fearful work environment and cost incurred in maintaining a surplus of employees in anticipation of suspension. In the same scenario, the utility of the attacker when target t_* is attacked is $p_*(U_A^a(t_*) - x) + (1 - p_*) U_A^u(t_*)$. The attacker suffers punishment x only when attacking an audited target.

A possible extension of the model above is to account for immediate losses that the defender suffers by imposing a punishment, e.g., firing or suspending an employee requires time and effort to find a replacement. Mathematically, we can account for such losses by including an additional term within the scope of p_* in the payoff of the defender: $p_*(U_D^a(t_*) - a_1 x) + (1 - p_*) U_D^u(t_*) - ax$, where a_1 is a constant. All our results (FPT and FPTAS algorithms) can be readily extended to handle this model extension, which we present in the full version.

Equilibrium. Under the Stackelberg equilibrium solution, the defender commits to a (randomized) strategy, followed by a best response by the adversary; the defender's strategy should maximize her utility. The mathematical problem involves solving multiple optimization problems, one each for the case when attacking t_* is in fact the best response of the adversary. Thus, assuming t_* is the best response of the adversary, the $*^{th}$ optimization problem P_* in our audit games setting is

$$
\begin{array}{ll}
\max\limits_{p_{ij}, x} & p_* U_D^a(t_*) + (1 - p_*) U_D^u(t_*) - ax \ , \\
\text{subject to} & \forall i \neq *.\ p_i(U_A^a(t_i) - x) + (1 - p_i) U_A^u(t_i) \\
& \qquad \leq p_*(U_A^a(t_*) - x) + (1 - p_*) U_A^u(t_*) \ , \\
& \forall j.\ 0 \leq \sum_{i=1}^{n} p_i^j \leq 1 \ , \\
& \forall i.\ 0 \leq p_i = \sum_{j=1}^{k} p_i^j \leq 1 \ , \forall (j, i).\ p_i^j \geq 0 \ , \\
& \forall (j, i) \in R.\ p_i^j = 0 \ , \quad 0 \leq x \leq 1 \ .
\end{array}
$$

The first constraint verifies that attacking t_* is indeed a best response for the adversary.

The auditor solves the n problems $P_1, \ldots, P_n$ (which correspond to the cases where the best response is $t_1, \ldots, t_n$, respectively), and chooses the best among all these solutions to obtain the final strategy to be used for auditing. This is a generalization of the multiple LPs approach of Conitzer and Sandholm (2006).

For ease of notation, let $\Delta_{D,i} = U_D^a(t_i) - U_D^u(t_i)$, $\Delta_i = U_A^u(t_i) - U_A^a(t_i)$ and $\delta_{i,j} = U_A^u(t_i) - U_A^u(t_j)$. Then, $\Delta_{D,i} \geq 0$, $\Delta_i \geq 0$, and the objective can be written as $p_n \Delta_{D,*} - ax$, subject to the quadratic constraint

$$p_i(-x - \Delta_i) + p_n(x + \Delta_*) + \delta_{i,*} \leq 0 \ .$$

Without loss of generality we will focus on the n'th program P_n, that is, we let $*$ be n.

Inputs. The inputs to the above problem are specified in K bit precision. Thus, the total length of all inputs is $O(nK)$.

3 Fixed-Parameter Tractable Algorithms

In this section, we present our FPT algorithm for optimization problem P_n, followed by the optimization transformation that improves the FPT algorithm and enables the FPTAS in the next section. Finally, we briefly describe an extension of our algorithmic results to target-specific punishments.

We start with the FPT for P_n. Our algorithm is based on the following straightforward observation: if we fix the punishment level x then the P_n becomes a linear program that can be solved in polynomial time. We therefore solve the optimization problem P_n for discrete values of x (with interval size ϵ) and take the solution that maximizes the defender's utility. This approach provides the following guarantee:

Theorem 1. *The above approach of solving for discrete values of x is a FPT $\Theta(\epsilon)$-additive approximation algorithm for the problem P_n if either the optimal value of x is greater than a small constant or $\Delta_n \neq 0$; the bit precision is the fixed parameter.*

Proof Sketch. The proof proceeds by arguing how much the objective changes when the value of x is changed by less than ϵ. The exact algebraic steps are in the full version. □

We emphasize that fixing the bit precision is reasonable, because inputs to the game model are never known with certainty, and therefore high-precision inputs are not used in practice (see, e.g., Nguyen et al. (2014), Kiekintveld et al. (2013), Blum et al. (2014)).

A naïve approach to improving our FPT algorithm is to conduct a binary search on the punishment rate x. This approach may fail, though, as the solution quality is not single-peaked in x. We demonstrate this in the full version using an explicit example. Instead, we describe a transformation of the optimization problem, which will enable a FPTAS for our problem under certain restrictions.

3.1 Extracting constraints for p_i's

The transformation eliminates variables p_i^j's and instead extracts inequalities (constraints) for the variables p_i's from the constraints below (referred to as grid constraints)

$$\forall i.\ 0 \le p_i = \sum_{j=1}^{k} p_i^j \le 1\ , \forall j.\ 0 \le \sum_{i=1}^{n} p_i^j \le 1\ ,$$
$$\forall (j,i).\ p_i^j \ge 0\ , \forall (j,i) \in R.\ p_i^j = 0\ .$$

Consider any subset of inspection resources L with the resources in L indexed by $s_1, \ldots, s_{|L|}$ ($|L| \le k$). We let $M = OnlyAuditedBy(L) \subset \{t_1, \ldots, t_n\}$ denote the subset of targets that can only be audited by a resource in L (e.g., for every target $t_i \in M$ and every resource $s_j \notin L$ the resource s_j cannot inspect the target t_i) . For notational convenience we assume that M is indexed by $t_1, \ldots, t_{|M|}$. Then, in case $|L| < |M|$, we obtain a constraint $p_{t_1} + \ldots + p_{t_{|M|}} \le |L|$ because there are only $|L|$ resources that could be used to audit these $|M|$ targets. We call such a constraint $c_{M,L}$. Consider the set of all such constraints C defined as

$$\{c_{M,L} \mid L \in 2^S, M = OnlyAuditedBy(L), |L| < |M|\}$$

where $S = \{s_1, \ldots, s_k\}$ and $T = \{t_1, \ldots, t_n\}$.

Lemma 1. *The optimization problem P_* is equivalent to the optimization problem obtained by replacing the grid constraints by $C \cup \{0 \le p_i \le 1\}$ in P_*.*

Proof Sketch. We present a sketch of the proof with the details in the full version. As the optimization objective depends on variables p_i's only, and quadratic constraints are identical in both problems we just need to show that the regions spanned by the variables p_i's, as specified by the linear constraints, are the same in both problems. As one direction of the inclusion is easy, we show the harder case below.

Let C^+ denote the convex polytope $C \cup \{0 \le p_i \le 1\}$. Given a point $(p_1, ..., p_n) \in C^+$ we want to argue that we can find values p_i^j's satisfying all of the grid constraints. We first note that it suffices to argue that we can find feasible p_i^j's for any extreme point in C^+ because any point in C^+ can be written as a convex combination of its extreme points (Gallier 2008). Thus, we could find feasible p_i^j's for any point in C^+ using this convex combination.

In the full version we prove that each extreme point in C^+ sets the variables $p_1, ..., p_n$ to 0 or 1. Let k' denote the number of ones in an extreme point. Note that $k' \le k$ because one of the inequalities is $p_1 + ... + p_n \le k$. Consider the undirected bipartite graph linking the inspection nodes to the target nodes, with a link indicating that the inspection can audit the linked target. This graph is known from our knowledge of R, and each link in the graph can be labeled by one of the p_i^j variables. Let S' be the set of targets picked by the ones in any extreme points. We claim that there is a perfect matching from S' to the the set of inspection resources (which we prove in next paragraph). Given such a perfect matching, assigning $p_i^j = 1$ for every edge in the matching yields a feasible solution, which completes the proof.

We prove the claim about perfect matching by contradiction. Assume there is no perfect matching, then there must be a set $S'' \subseteq S'$, such that $|N(S'')| < |S''|$ (N is the neighbors function — this result follows from Hall's theorem). As $S'' \subseteq S'$ it must hold that $p_i = 1$ for all $i \in index(S'')$ (function index gives the indices of the set of targets). Also, the set of targets S'' is audited only by inspection resources in $N(S'')$ and, by definition of C, we must have a constraint $\sum_{i \in index(S'')} p_i \le |N(S'')|$. Using $|N(S'')| < |S''|$, we get $\sum_{i \in index(S'')} p_i < |S''|$. But, since $|index(S'')| = |S''|$, we conclude that all p_i for targets in S'' cannot be one, which is a contradiction. □

Observe that obtaining p_i^j's from the p_i's involves solving a linear feasibility problem, which can be done efficiently.

Importantly, the definition of C is constructive and provides an algorithm to compute it. However, the algorithm has a worst-case running time exponential in k. Indeed, consider k resources $s_1, \ldots, s_k$ and $2k$ targets $t_1, \ldots, t_{2k}$. Each resource s_i can inspect targets $t_1, t_2, t_{2i-1}, t_{2i}$. For each set of $k/2$ resources $L \subseteq \{s_2, \ldots, s_k\}$ we have $M = OnlyAuditedBy(L) = \{t_1, t_2\} \cup \left(\bigcup_{s_j \in L} \{t_{2j-1}, t_{2j}\}\right)$. Observe that $|M| = k + 2 > k/2 = |L|$ so for each $L \subseteq \{s_2, \ldots, s_k\}$ we get a new constraint $c_{M,L}$. Thus, we get $\binom{k-1}{k/2}$ constraints.

3.2 Conditions for Poly. Number of Constraints

Motivated by the above observation, we wish to explore an alternative method for computing C. We will also provide sufficient conditions under which $|C|$ is polynomial.

The intuition behind our alternative algorithm is that instead of iterating over sets of inspection resources, we could iterate over sets of targets. As a first step, we identify equivalent targets and merge them. Intuitively, targets that can be audited by the exact same set of inspections are equivalent. Formally, t_i and t_k are equivalent if $F(t_i) = F(t_k)$ where $F(t_\ell) = \{s_j \,|\, (j,\ell) \notin R\}$.

The algorithm is formally given as Algorithm 1. It builds an intersection graph from the merged targets: every merged set of targets is a node, and two nodes are linked if the two sets of inspection resources corresponding to the nodes intersect. The algorithm iterates through every connected induced sub-graph and builds constraints from the targets associated with the nodes in the sub-graphs and the set of inspection resources associated with them. The next lemma proves the correctness of the Algorithm 1.

Lemma 2. CONSTRAINT_FIND *outputs constraints that define the same convex polytope in $p_1, \ldots, p_n$ as the constraints output by the naïve algorithm (iterating over all subsets of resources).*

The proof appears in the full version. The algorithm is clearly not polynomial time in general, because it iterates over all connected subgraphs. The next lemma provides sufficient conditions for polynomial running time.

Lemma 3. CONSTRAINT_FIND *runs in polynomial time if at least one of the following conditions holds:*

- *The intersection graph has $O(\log n)$ nodes.*

Algorithm 1: CONSTRAINT_FIND(T, R)

Compute F, the map from T to $2^{\{s_1,\ldots,s_k\}}$ using R.
Merge targets with same $F(t)$ to get set T' and a map W, where $W(t')$ = #merged targets that yielded t'
Let $PV(t')$ be the set of prob. variables associated with t', one each from the merged targets that yielded t'
Form an intersection graph G with nodes $t' \in T'$ and edge set $E = \{\{t'_i, t'_k\} | F(t'_i) \cap F(t'_k) \neq \emptyset\}$
$L \leftarrow$ CONNECTEDSUBGRAPHS(G)
$C \leftarrow \phi$
for $l \in L$ **do**
  Let V be all the vertices in l
  $P \leftarrow \bigcup_{v \in V} PV(v)$
  $k \leftarrow \sum_{v \in V} W(t')$
  if $|P| > k$ **then**
    $C \leftarrow C \cup \{\sum_{p \in P} p \leq k\}$
return C

- *The intersection graph has constant maximum degree and a constant number of nodes with degree at least* 3*.*

Proof Sketch. The detailed proof is in the full version. It is not hard to observe that we need sufficient conditions for any graph to have polynomially many induced connected sub-graphs. The first case above is obvious as the number of induced connected sub-graphs in the worst case (fully connected graph) is 2^N, where N is number of nodes. The second case can be proved by an induction on the number of nodes in the graphs with degree greater than 3. Removing any such vertex results in a constant number of disconnected components (due to constant max degree). Then, we can argue that the number of connected sub-graphs of the given graph will scale polynomially with the max number of connected sub-graphs of each component. The base case involves graphs of degree less than two, which is a graph with paths and cycles and such a graph has polynomially many connected sub-graphs. □

Why Are These Conditions Realistic? The conditions specified in Lemma 3 capture a wide range of practical audit scenarios.

First, many similar targets can often be grouped together by type. In a hospital case, for instance, rather than considering each individual health record as a unique target worthy of specific audit strategies, it might make more sense to have identical audit policies for a small set of patient types (e.g., celebrities, regular folks...). Likewise, in the context of tax audits, one could envisage that individuals are pooled according to their types (e.g., high income earners, expatriates, ...). In practice, we expect to see only a few different types. Each type corresponds to a single node in the intersection graph, so that a constant number of types corresponds to a constant number of nodes. That is, both of the lemma conditions are satisfied, even though only one is required.

Second, auditing is often localized. For instance, when considering audits performed by corporate managers, one would expect these managers to primarily inspect the activities of their direct subordinates. This means that the inspection resources (inspection actions of a manager) auditing a node (activities of its subordinates) are disjoint from the inspection resources auditing any other node. Thus, our intersection graph has no edges, and the second lemma condition is satisfied. Slightly more complex situations, where, for instance, employees' activities are audited by two different managers, still satisfy the second condition.

3.3 Target-Specific Punishments

We present a brief overview of target-specific punishments with the details in the full version. We extend our model to target-specific punishments by augmenting the program P_n: we use individual punishment levels $x_1, \ldots, x_n$, instead of using the same punishment x for each target. The new optimization problem PX_n differs from P_n only in (1) objective: $\max_{p_i,x} p_n \Delta_{D,n} - \sum_{j \in \{1,\ldots,n\}} a_j x_j$ and (2) quadratic constraints: $p_i(-x_i - \Delta_i) + p_n(x_n + \Delta_n) + \delta_{i,n} \leq 0$.

The naïve way of discretizing each of the variables $x_1, \ldots, x_n$ and solving the resulting sub-problems is not polynomial time. Nevertheless, we show that it is possible to design an FPT approximation algorithm by discretizing only p_n, and casting the resulting sub-problems as second-order cone programs (SOCP), which can be solved in polynomial time (Boyd and Vandenberghe 2004). We first present the following intuitive result:

Lemma 4. *At the optimal point for PX_n, x_n is always 0. Further, discretizing values of p_n with interval size ϵ and solving resulting sub-problems yields a $\Theta(\epsilon)$ approximation.*

The proof is in the full version. Next, we show that for fixed values of x_n and p_n, PX_n reduces to an SOCP. We first describe a general SOCP problem (with variable $y \in \mathbb{R}^n$) that maximizes a linear objective $f^T y$ subject to linear constraints $Fy = g$ and m quadratic constraints of the form

$$\forall i \in \{1, \ldots, m\}.\ ||A_i y + b_i||_2 \leq c_i^T y + d_i$$

In particular, the constraint $k^2/4 \leq y_i(y_j + k')$ is the same as $||[k\ \ (y_i - y_j - k')]^T||_2 \leq y_i + y_j + k$, which is an instance of the quadratic constraint above for appropriate A, b, c, d.

Our problem can be cast as an SOCP by rewriting the quadratic constraints as

$$p_n(x_n + \Delta_n) + \delta_{i,n} \leq p_i(x_i + \Delta_i)$$

Using our approach (discretizing p_n, $x_n = 0$) the LHS of the above inequality is a constant. If the constant is negative we can simply throw out the constraint — it is a tautology since the RHS is always positive. If the constant is positive, we rewrite the constraint as a second-order constraint as described above (e.g., set $k = 2\sqrt{p_n(x_n + \Delta_n) + \delta_{i,n}}$ and set $k' = \Delta_i$). The rest of the constraints are linear. Thus, the problem for each fixed value of p_n, x_n is an SOCP. Putting everything together, we have proved the following result.

Theorem 2. *The method described above is an FPT additive $\Theta(\epsilon)$-approximate algorithm for solving PX_n.*

4 Fully Polynomial Time Approximation

Our goal in this section is to develop an FPTAS for problem P_n, under the condition that the set C returned by CONSTRAINT_FIND has polynomially many constraints. Our algorithm builds on an earlier algorithm (Blocki et al. 2013) for the restricted auditing scenario with just one defender resource. Since we solve the defender's problem after extracting the constraints C, the variables in our problem are just the p_i's and x.

$$\begin{array}{ll} \max\limits_{p_i,x} & p_n\Delta_{D,n} - ax \ , \\ \text{subject to} & \forall i \neq n. \\ & p_i(-x-\Delta_i) + p_n(x+\Delta_n) + \delta_{i,n} \leq 0 \ , \\ & c \in C, \ \ \forall i.\ 0 \leq p_i \leq 1 \ , \quad 0 \leq x \leq 1 \ . \end{array}$$

Property of optimal points. We state the following property of some optimal points p_i^*'s and x^* of the optimization:

Lemma 5. *There exists optimal points p_i^*'s and x^* such that if $p_n^*(x^*+\Delta_n) + \delta_{j,n} \geq 0$ then $p_n^*(x^*+\Delta_n) + \delta_{j,n} = p_j^*(x^*+\Delta_j)$ (i.e., quadratic constraint is tight) else when $p_n^*(x^*+\Delta_n) + \delta_{j,n} < 0$ then $p_j^* = 0$.*

Proof Sketch. The quadratic constraint can be written as $\frac{p_n(x+\Delta_n)+\delta_{j,n}}{x+\Delta_i} \leq p_i$ At the optimal point if $p_n^*(x^*+\Delta_n) + \delta_{j,n} \geq 0$ then if we have $\frac{p_n^*(x^*+\Delta_n)+\delta_{j,n}}{x+\Delta_i} < p_i^*$, we can always reduce p_i^* without affecting the objective value till we get an equality. Also, for the case $p_n^*(x^*+\Delta_n) + \delta_{j,n} < 0$ we can reduce p_i^* to 0. □

We focus on finding one of the optimal points with the property stated above. Next, we sort $\delta_{i,n}$'s to get a sorted array $\delta_{(i),n}$ in ascending order. Then, we split the optimization problem P_n into sub-problems $EQ_{(j)}$, where in each problem $EQ_{(j)}$ it is assumed that p_n, x lies between the hyperbolas $p_n(x+\Delta_n) + \delta_{(j),n}$ (open boundary) and $p_n(x+\Delta_n) + \delta_{(j+1),n}$ (closed boundary) in the plane spanned by p_n, x. Thus, in $EQ_{(j)}$, $p_n(x+\Delta_n) + \delta_{(j),n}$ is non-negative for all $(k) > (j)$ and negative otherwise. Using the property of the optimal point above, for the non-negative case we obtain equalities for the quadratic constraints and for the negative case we claim that for all $(k) \leq (j)$ we can set $p_{(k)} = 0$. The optimal value for P_n can be found by solving each $EQ_{(j)}$ and taking the best solution from these sub-problems.

The optimization problem for $EQ_{(j)}$ is as follows:

$$\begin{array}{ll} \max\limits_{p_i,x} & p_n\Delta_{D,n} - ax \ , \\ \text{subject to} & \forall (i) > (j).\ 0 \leq \frac{p_n(x+\Delta_n)+\delta_{(i),n}}{x+\Delta_{(i)}} = p_{(i)} \leq 1 \\ & p_n(x+\Delta_n) + \delta_{(j),n} < 0 \\ & p_n(x+\Delta_n) + \delta_{(j+1),n} \geq 0 \\ & c \in C,\ \forall (i) \leq (j).\ p_{(i)} = 0 \ , \quad 0 \leq x \leq 1 \ . \end{array}$$

As no p_i (except p_n) appears in the objective, and due to the equality constraints on particular p_i's, we can replace those p_i's by a function of p_n, x. Other p_i's are zero. Next, by a series of simple algebraic manipulations we obtain a two-variable optimization problem:

$$\begin{array}{ll} \max\limits_{p_i,x} & p_n\Delta_{D,n} - ax \ , \\ \text{subject to} & \forall b \in \{1,\ldots,B\}.\ p_n \leq f_b(x) \\ & 0 \leq p_n \leq 1 \ , \quad 0 \leq x \leq 1 \ , \end{array}$$

where B is the total number of constraints, which is of the same order as $|C|$. The details of the algebraic steps are in the full version.

Solving the sub-problem. Our two final lemmas are not hard to prove. Their proofs appear in the full version.

Lemma 6. *Problem $EQ_{(j)}$ can be solved efficiently for a fixed value x or fixed value of p_n.*

Lemma 7. *The optimal point for EQ_j cannot be an interior point of the region defined by the constraints, i.e., at least one of the inequalities is tight for the optimal point.*

Proof Sketch. It is easy to see that if all constraints are non-tight at the optimal point, then p_n can be increased by a small amount without violating any constraint and also increasing the objective value. Thus, some constraint must be tight. □

Algorithm 2: APX_SOLVE$(l, EQ_{(j)})$

```
Solve the problem for p_n = 0, 1 and x = 0, 1
Collect solutions (p_n, x) from the above in M
for b ← 1 to B do
    Replace p_n = f_b(x) in the objective to get
    F(x) = f_b(x)Δ_{D,n} − ax
    Take the derivative to get F'(x) = ∂F(x)/∂x
    R ← ROOTS(F'(x), 2^{-l}, (0, 1))
    R' ← MAKEFEASIBLE(R)
    From R' obtain set M' of potential solutions (p_n, x)
    M ← M ∪ M'
for (b, b') ∈ {(b, b') | b ∈ B, b ∈ B, b' > b} do
    Equate f_b(x) = f_{b'}(x) to get F(x) = 0
    R ← ROOTS(F(x), 2^{-l}, (0, 1))
    R' ← MAKEFEASIBLE(R)
    From R obtain set M' of potential solutions (p_n, x)
    M ← M ∪ M'
(p_n^*, x^*) ← arg max_M {p_n Δ_{D,n} − ax}
return (p_n^*, x^*)
```

We are now ready to present the FPTAS, given as Algorithm 2. The algorithm first searches potential optimal points on the boundaries (in the first loop) and then searches potential optimal points at the intersection of two boundaries (second loop). The roots are found to an additive approximation factor of 2^{-l} in time polynomial in the size of the problem representation and l (Schönhage 1982). As shown in Blocki et al. (2013), the case of roots lying outside the feasible region (due to approximation) is taken care of by the function MAKEFEASIBLE. The first loop iterates a maximum of n times, and the second loop iterates a maximum of $\binom{n}{2}$ times. Thus, we have the following result.

Game	#Target	#Resource (GroupSize)	Time (min) T	Time (min) NT
Audit	100	10 (2)	12	15
Audit	200	100 (10)	81	234
Security	3,000	500 (10)	28	8,500[1]
Security	5,000	1,000 (20)	119	110,000[1]

Table 1: FPT algorithm running times (in min) with our optimization transformation (T) and no transformation (NT).

Theorem 3. *The optimization problem P_n can be solved with an additive approximation factor of $\Theta(2^{-l})$ in time polynomial in the input size and l, i.e., our algorithm to solve P_n is an FPTAS.*

5 Experimental Results

In this section, we empirically demonstrate the speedup gains from our optimization transformation for both audit games and security games. We obtained speedups of up to $3\times$ for audit game instances and over $100\times$ for associated security game instances.

Our experiments were run on a desktop with quad core 3.2 GHz processor and 6GB RAM. Code was written in Matlab using the built-in large scale interior point method implementation of linear programming. We implemented two FPT algorithms—with and without the optimization transformation. For both the algorithms, we used the same problem inputs in which utilities were generated randomly from the range $[0, 1]$, a was fixed to 0.01, x was discretized with interval size of 0.005.

We ran experiments for audit games and security games with varying number of targets and resources. The resources were divided into equal sized groups such that the targets any group of resources could inspect was disjoint from the target set for any other group. Table 1 shows our results with the varying number of targets, resources and size of the group of targets. The results are an average over 5 runs of the optimization with random utilities in each run. Audit games take more time to solve than corresponding security games with a similar number of targets as we run the corresponding LP optimization 200 times (the discrete interval for x is 0.005). Hence we solve for larger security game instances.

Our implementations did not optimize for speed using heuristics because our goal was to only test the speedup gain from our optimization transformation. Thus, we do not scale up to the number of targets that heuristic approaches such as ORIGAMI (Kiekintveld et al. 2009) achieve (1,000,000 targets). ORIGAMI works by iteratively building the optimal attack set, i.e., the set of targets that the adversary finds most attractive at the optimal solution point. However, ORIGAMI considers only coverage probabilities (marginals). Thus, its output may not be implementable with scheduling constraints on resources. In contrast, our approach guarantees that the coverage probabilities output are implementable. Wedding the two approaches to obtain scalable and provably implementable audit and security game solutions remains an interesting direction for future work.

[1]These data points are extrapolations from runs that were allowed to run 12 hours. The extrapolation was based on the number of optimization problems solved vs the total number of optimization problems (3000/5000 total problems for 3000/5000 targets).

References

Birkhoff, G. 1946. Three observations on linear algebra. *Univ. Nac. Tucumán. Revista A.* 5:147–151.

Blocki, J.; Christin, N.; Datta, A.; Procaccia, A. D.; and Sinha, A. 2013. Audit games. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 41–47.

Blum, A.; Haghtalab, N.; and Procaccia, A. D. 2014. Learning optimal commitment to overcome insecurity. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*. Forthcoming.

Boyd, S. P., and Vandenberghe, L. 2004. *Convex optimization*. Cambridge University Press.

Conitzer, V., and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, 82–90.

Dulmage, L., and Halperin, I. 1955. On a theorem of Frobenius-König and J. von Neumann's game of hide and seek. *Trans. Roy. Soc. Canada. Sect. III. (3)* 49:23–29.

Fairwarning. 2011. Industry Best Practices for Patient Privacy in Electronic Health Records.

Kiekintveld, C.; Jain, M.; Tsai, J.; Pita, J.; Ordóñez, F.; and Tambe, M. 2009. Computing optimal randomized resource allocations for massive security games. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 689–696.

Kiekintveld, C.; Islam, T.; and Kreinovich, V. 2013. Security games with interval uncertainty. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 231–238.

Korzhyk, D.; Conitzer, V.; and Parr, R. 2010. Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *Proceedings of the 24th AAAI conference on Artificial Intelligence (AAAI)*, 805–810.

Nguyen, T. H.; Jiang, A. X.; and Tambe, M. 2014. Stop the compartmentalization: Unified robust algorithms for handling uncertainties in security games. In *Proceedings of the 13th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 317–324.

Paruchuri, P.; Pearce, J. P.; Marecki, J.; Tambe, M.; Ordónez, F.; and Kraus, S. 2009. Coordinating randomized policies for increasing security of agent systems. *Information Technology and Management* 10(1):67–79.

Ponemon Institute. 2011. Second Annual Benchmark Study on Patient Privacy and Data Security.

Ponemon Institute. 2012. 2011 Cost of Data Breach Study: United States.

Schönhage, A. 1982. The fundamental theorem of algebra in terms of computational complexity. Technical report, University of Tübingen.

Tambe, M. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

Appendix

A Birkhoff-von Neumann

For the sake of completeness, we state the Birkhoff-von Neumann theorem from Korzhyk et al. (Korzhyk, Conitzer, and Parr 2010), that is used to decompose the probability matrix into pure actions efficiently.

(Birkhoff-von Neumann (Birkhoff 1946)). Consider an $m \times n$ matrix M with real numbers $a_{ij} \in [0,1]$, such that for each $1 \leq i \leq m$, $\sum_{j=1}^{n} a_{ij} \leq 1$, and for each $1 \leq j \leq n$, $\sum_{i=1}^{m} a_{ij} \leq 1$. Then, there exist matrices $M_1, M_2, \ldots, M_q$, and weights $w_1, w_2, \ldots, w_q \in (0,1]$, such that:

1. $\sum_{k=1}^{q} w_k = 1$
2. $\sum_{k=1}^{q} w_k M_k = M$
3. *for each $1 \leq k \leq q$, the elements of M_k are $a_{ij}^k \in \{0,1\}$*
4. *for each $1 \leq k \leq q$, we have: for each $1 \leq i \leq m$, $\sum_{j=1}^{n} a_{ij}^k \leq 1$, and for each $1 \leq j \leq n$, $\sum_{i=1}^{m} a_{ij}^k \leq 1$.*

Moreover, q is $O((m+n)^2)$, and the M_k and w_k can be found in $O((m+n)^{4.5})$ time using Dulmage-Halperin algorithm (Dulmage and Halperin 1955).

Clearly, our variables p_i^j's can be considered as the matrix M in the result above, and hence can be decomposed into pure actions efficiently.

B Obtaining the two variable optimization

The optimization problem for $EQ_{(j)}$ is as follows:

$$\begin{array}{ll} \max\limits_{p_i,x} & p_n\Delta_{D,n} - ax \ , \\ \text{subject to} & \forall (i) > (j).\ 0 \leq \frac{p_n(x+\Delta_n)+\delta_{(i),n}}{x+\Delta_{(i)}} = p_{(i)} \leq 1 \\ & p_n(x+\Delta_n) + \delta_{(j),n} < 0 \\ & p_n(x+\Delta_n) + \delta_{(j+1),n} \geq 0 \\ & c \in C,\ \forall (i) \leq (j).\ p_{(i)} = 0\ , \quad 0 \leq x \leq 1\ . \end{array}$$

As no p_i (except p_n) shows up in the objective, and due to the equality constraints on particular p_i's, we can replace those p_i's by a function of p_n, x. Other p_i's are zero. Denote by $c(p_n, x)$ the inequality obtained after substituting p_i with the function of p_n, x (or zero) in the constraint $c \in C$. Thus, we get the following two variable optimization problem

$$\begin{array}{ll} \max\limits_{p_i,x} & p_n\Delta_{D,n} - ax \ , \\ \text{subject to} & \forall (i) > (j).\ \frac{p_n(x+\Delta_n)+\delta_{(i),n}}{x+\Delta_{(i)}} \leq 1 \\ & p_n(x+\Delta_n) + \delta_{(j),n} < 0 \\ & p_n(x+\Delta_n) + \delta_{(j+1),n} \geq 0 \\ & c(p_n, x) \in C,\ 0 \leq p_n \leq 1\ , \quad 0 \leq x \leq 1\ . \end{array}$$

Observe that we removed the $0 \leq$ condition in the first set of constraints because that is implied by the next two constraints. Next, note that any constraint (indexed by b) with two variables p_n, x can be expressed as $p_n \leq f_b(x)$ (closed boundary) for constraint-specific ratio of polynomials $f_b(x)$ or by $p_n < \frac{-\delta_{(j),n}}{x+\Delta_n}$ (open boundary). However, we close the open boundary, i.e., $p_n \leq \frac{-\delta_{(j),n}}{x+\Delta_n}$, and solve the problem, returning an infeasible solution in case the solution is on the boundary $p_n = \frac{-\delta_{(j),n}}{x+\Delta_n}$. This is justified by the fact that the curve $p_n = \frac{-\delta_{(j),n}}{x+\Delta_n}$ is included in the other optimization problem $EQ_{(j-1)}$, and would be output by that sub-problem if it indeed is the global maximizer. Thus, we represent the optimization problem as

$$\begin{array}{ll} \max\limits_{p_i,x} & p_n\Delta_{D,n} - ax \ , \\ \text{subject to} & \forall b \in \{1, \ldots, B\}.\ p_n \leq f_b(x) \\ & 0 \leq p_n \leq 1\ , \quad 0 \leq x \leq 1\ , \end{array}$$

where B is the total number of constraints described as above.

C Target-Specific Punishments

In this section we extend our FPT result to target-specific punishments. While the formulation of the extended problem is not that different, solving it becomes significantly more challenging.

In more detail, we augment the program P_n by using individual punishment levels $x_1, \ldots, x_n$, instead of using the same punishment x for each target. The new optimization problem PX_n is

$$\begin{array}{ll} \max\limits_{p_i,x} & p_n\Delta_{D,n} - \sum_{j \in \{1,\ldots,n\}} a_j x_j\ , \\ \text{s.t.} & \forall i \neq n.\ p_i(-x_i - \Delta_i) + p_n(x_n + \Delta_n) + \delta_{i,n} \leq 0 \\ & \forall j.\ 0 \leq \sum_{i=1}^{n} p_i^j \leq 1\ , \forall i.\ 0 \leq p_i = \sum_{j=1}^{k} p_i^j \leq 1\ , \\ & \forall (j,i).\ p_i^j \geq 0\ , \forall (j,i) \in R.\ p_i^j = 0\ , \\ & 0 \leq x \leq 1\ . \end{array}$$

Note that the defender's penalty term is now a linear combination of the target-specific punishment level.

The naïve way to approach the above problem is to discretize each of the variables $x_1, \ldots, x_n$ and solve the resulting sub-problems, which are linear programs. However, such a discretization of size ϵ, even if yielding a $\Theta(\epsilon)$ additive approximation, will run in time $O((1/\epsilon)^n)$, which is not polynomial for constant ϵ.

Nevertheless, we show that it is possible to design an algorithm that runs in time polynomial in ϵ and yields a $\Theta(\epsilon)$ additive approximation, by discretizing only p_n, and casting the resulting sub-problems as second-order cone programs (SOCP), which can be solved in polynomial time (Boyd and Vandenberghe 2004). Letting $S(n)$ denote the (polynomial) running time of the SOCP, we obtain a running time of $O(S(n)/\epsilon)$, given fixed bit precision.

We first present the following intuitive result:

Restatement of Lemma 4. At the optimal point for PX_n, x_n is always 0. Further, considering discrete values of p_n with interval size ϵ and solving the resulting sub-problems exactly yields a $\Theta(\epsilon)$ approximation.

The proof is in the Missing Proofs section of the appendix.

Next, we show that for fixed values of x_n and p_n the optimization problem PX_n reduces to a second-order cone program. We first describe a general SOCP problem:

$$\begin{array}{ll} \max\limits_{y} & f^T y\ , \\ \text{subject to} & \forall i \in \{1,\ldots,m\}.\ ||A_i y + b_i||_2 \leq c_i^T y + d_i \\ & Fy = g\ . \end{array}$$

where the optimization variable is $y \in \mathbb{R}^n$, and all constants are of appropriate dimensions. In particular, we observe that the constraint

$$k^2/4 \leq y_i(y_j + k')$$

can always be written as a second order inequality $||Ay + b||_2 \leq c^T y + d$ by selecting A and b such that $Ay + b = [k\ \ (y_i - y_j - k')]^T$ and c, d such that $c^T y + d = y_i + y_j + k$.

Our problem can be cast as a SOCP by rewriting the quadratic constraints as

$$p_n(x_n + \Delta_n) + \delta_{i,n} \leq p_i(x_i + \Delta_i)$$

Using our approach (discretizing p_n, $x_n = 0$) the LHS of the above inequality is a constant. If the constant is negative we can simply throw out the constraint — it is a tautology since the RHS is always positive. If the constant is positive, we rewrite the constraint as a second-order constraint as described above (e.g., set $k = 2\sqrt{p_n(x_n + \Delta_n) + \delta_{i,n}}$ and set $k' = \Delta_i$). The rest of the constraints are linear, and can be rewritten in equality form by introducing slack variables. Thus, the problem for each fixed value of p_n, x_n is an SOCP, and can be solved efficiently. Putting everything together, we get the main Theorem 2 of the target-specific punishment section in the paper.

D Missing Proofs

Proof of Theorem 1. The quadratic constraint can be rewritten as

$$x(p_n - p_i) - p_i\Delta_i + p_n\Delta_n + \delta_{i,n} \leq 0$$

Suppose we have an optimal point x^o, p_i^o's. First, we note that if $x^o = 1$ (resp. $x^o = 0$) then our algorithm will find the optimal solution exactly because $x = 1$ (resp. $x = 0$) is one of the discrete values of x considered by our algorithm. In the rest of the proof we assume that $0 < x^o < 1$. Let $\epsilon' > 0$ denote the smallest value s.t. $x^o + \epsilon'$ is one of the discrete values of x considered by our algorithm (e.g. $x^0 + \epsilon' \in \{\epsilon i \,|\, i \in \mathbb{N}\} \cup \{1\}$). We let

$$\tau = \max\left\{0, \max_{i \leq n} \frac{\epsilon'\,(p_n^o - p_i^o)}{x^o + \epsilon' + \Delta_n}\right\}\ ,$$

and we consider two cases.

Case 1: $p_n^o \geq \tau$. In this case we set $\hat{x} = x^o + \epsilon'$, $\hat{p}_n = p_n^o - \tau$ and $\hat{p}_i = p_i^o$ for each $i \neq n$. We first show that this is a valid solution. For each $i \neq n$ we have

$$\begin{array}{lll} & \hat{x}(\hat{p}_n - \hat{p}_i) - \hat{p}_i\Delta_i + \hat{p}_n\Delta_n + \delta_{i,n} \\ = & (x^o + \epsilon')(p_n^o - \tau - p_i^o) - p_i^o\Delta_i + (p_n^o - \tau)\Delta_n + \delta_{i,n} \\ = & (x^o(p_n^o - p_i^o) - p_i^o\Delta_i + p_n^o\Delta_n + \delta_{i,n}) \\ & + (\epsilon'(p_n^o - p_i^o) - \tau\epsilon' - \tau\Delta_n) - x^o\tau \\ \leq & \epsilon'(p_n^o - p_i^o) - \tau(x^o + \Delta_n + \epsilon') \\ = & \epsilon'(p_n^o - p_i^o) \\ & - (x^o + \Delta_n + \epsilon')\max\left\{0, \max\limits_{k \leq n} \frac{\epsilon'\,(p_n^o - p_k^o)}{x^o + \Delta_n + \epsilon'}\right\} \\ \leq & \epsilon'(p_n^o - p_i^o) - \max\limits_{k \leq n} \epsilon'\,(p_n^o - p_k^o) \\ \leq & 0\ . \end{array}$$

Note that the loss in utility for the defender is $\Delta_{D,n}\tau + a\epsilon'$ which is upper bounded by

$$\epsilon\left(\max_{i \leq n} \frac{|p_n^o - p_i^o|}{x^o + \Delta_n} + a\right)\ .$$

That is ϵb for a constant b when either x^o is greater than a constant or $\Delta_n \neq 0$. Observe that due to the fixed bit precision assumption $\Delta_n \neq 0$ implies Δ_n is greater than a constant. We remark that the solution returned by our algorithm will be at least as good as the solution given by $\hat{x}$ and $\hat{p}_i$'s because $\hat{x}$ is one of the discrete values of x considered. Thus, the defender's utility in the solution given by our algorithm is $\theta(\epsilon)$-close to the defenders utility in the optimal solution.

Case 2: $p_n^o < \tau$. In this case we set $\hat{p_n} = 0$, $\hat{x} = x^o + \epsilon'$ and $\hat{p_i} = p_i^o$ for each $i \neq 0$. We first show that this is a valid solution. For each $i \neq n$ we have

$$\begin{array}{lll} & \hat{x}(\hat{p}_n - \hat{p}_i) - \hat{p}_i\Delta_i + \hat{p}_n\Delta_n + \delta_{i,n} \\ = & (x^o + \epsilon')(p_n^o - p_n^o - p_i^o) - p_i^o\Delta_i + (p_n^o - p_n^o)\Delta_n + \delta_{i,n} \\ = & (x^o(p_n^o - p_i^o) - p_i^o\Delta_i + p_n^o\Delta_n + \delta_{i,n}) \\ & + \epsilon'(-p_i^o) - xp_n^o - p_n^o\Delta_n \\ \leq & 0\ . \end{array}$$

Note that the loss in utility for the defender is at most $\Delta_{D,n}\tau + a\epsilon'$ which is upper bounded by

$$\epsilon\left(\max_{i \leq n} \frac{|p_n^o - p_i^o|}{x^o + \Delta_n} + a\right)\ .$$

That is ϵb for a constant b. We again remark that the solution returned by our algorithm will be at least as good as the solution given by $\hat{x}$ and $\hat{p}_i$'s because $\hat{x}$ is one of the discrete values of x considered. Thus, the defender's utility in the solution given by our algorithm is $\theta(\epsilon)$-close to the defenders utility in the optimal solution.

□

Proof of Lemma 1. Since the optimization objective depends on the variables p_i's only, and the quadratic constraints are same in both problems we just need to show that the regions spanned by the variables p_i's, as specified by the linear constraints, are the same in both problems.

First, let p_i's, p_i^j's belong to region given by the grid constraints. Then, by the definition of C, for any c_{ML} we know M is audited only by L. Thus, for $i \in \{t_1, \ldots, t_{|M|}\}$ the variables p_i^j are non-zero only when $j \in \{s_1, \ldots, s_{|L|}\}$. Therefore,

$$\sum_{i \in \{t_1,\ldots,t_{|M|}\}} p_i \leq \sum_{i \in \{t_1,\ldots,t_{|M|}\}} \sum_{j \in \{s_1,\ldots,s_{|L|}\}} p_i^j \leq |L|$$

Hence, p_i's satisfies all constraints in $C \cup \{0 \leq p_i \leq 1\}$, and therefore p_i's belong to region given by $C \cup \{0 \leq p_i \leq 1\}$.

Next, let p_i's belong to region given by $C \cup \{0 \leq p_i \leq 1\}$. We first show that the extreme points of the convex polytope given by $C \cup \{0 \leq p_i \leq 1\}$ sets the variables $p_1, \ldots, p_n$ to either 0 or 1.

Extreme points 0/1

We prove this result with additional (redundant) constraints in C. These redundant constraints are the ones that were dropped due to $|L| \geq |M|$. As these constraints are redundant, the polytope with or without them is same, and so are the extreme points. Thus, for this part (extreme points) we assume C includes these redundant constraints.

We will do this by induction on the size of the restricted set R. The base case is when $|R| = 0$, then the only constraint in C is $\sum_i p_i \leq k$, i.e., A_C is $\vec{1}$. It can be checked directly the extreme points have k ones and other zeros.

Next assume the result holds for all R with $|R| = w$. Consider a restriction R with $|R| = w + 1$. Choose any particular restriction, say $(j, i) \in R$. Suppose when this restriction does not exist, we have the restriction $R' = R \backslash (j, i)$ of size w and by induction hypothesis with R' the extreme points are 0/1. We proceed to do the induction by checking how the constraint set C and C' differ.. We divide the proof into two cases.

Case 1 First, suppose with restriction R', t_i could be inspected only by k_j, then with R, t_i is not inspected at all. Recall that any constraint $c_{M,L}$ denotes a set of resources L and targets M such that the targets are inspected only by resources. We construct the set of constraints C and C' by iterating through all subsets of resources. If for any subset of resources L, $k_j \notin L$ then the set of targets inspected only by L does not include t_i for both R and R' and is the same set for both R and R'. Thus, $c_{M,L} \in C$ and $c_{M,L} \in C'$ and p_i does not show up in these constraints. On the other hand, if $k_j \in L$, then for R we have $t_i \notin M$ but for R' we have $M' = M \cup t_i$. Thus, for such a subset L we know that

- (differ1) C and C' include these constraint with the difference that $c_{M',L}$ has the additional variable p_i on the LHS.

Thus, the constraints C are formed from constraints C' by setting $p_i = 0$. But, setting $p_i = 0$ is the intersection of $p_i = 0$ and the polytope given by C'. $p_i = 0$ is a $k-1$-face of the polytope given by C'. Thus, the extreme points of this intersection polytope must be a subset of extreme points of polytope given by C' and hence integral.

Case 2 In the second case with restriction R', t_i is inspected by at least one resource other than k_j, then with restriction R, t_i is still inspectable. Let the set of resources that can inspect t_i with restriction R be K_i. We construct the set of constraints C and C' by iterating through all subsets of resources. If for any subset of resources L, $k_j \notin L$ then there are two cases possible:

- $K_i \subseteq L$ which implies $t_i \in M$ and $t_i \notin M'$ and $M = M' \cup t_i$. For this scenario we know that
  - (differ2) C' and C both include these constraint with the difference that $c_{M,L}$ has the additional variable p_i on the LHS.
- $K_i \nsubseteq L$ which implies $t_i \notin M$ and $t_i \notin M'$, i.e., the set of targets inspected only by L is same for both R and R'

On the other hand, if $k_j \in L$, then there are two cases possible:

- $K_i \subseteq L$ which implies $t_i \in M$ and $t_i \in M'$, i.e., the set of targets inspected only by L is same for both R and R'.
- $K_i \nsubseteq L$ which implies $t_i \notin M$ and $t_i \notin M'$, i.e., the set of targets inspected only by L is same for both R and R'

For the cases where the set of targets inspected is same for both R and R', as argued earlier, $c_{M,L} \in C$ and $c_{M,L} \in C'$. Thus, the only scenario to reason about is differ2. We do a step-by-step proof, by modifying constraints in C' one by one to obtain the constraints C, in the process showing for each step that the extreme points are 0/1. Thus, at each step we modify the polytope with 0/1 extreme points (say R_l) given by C_l to obtain the polytope (R_{l+1}) given by C_{l+1} by modifying one constraint c that did not have p_i on the LHS to one c_{+p_i} that has p_i on the LHS. Clearly, $R_{l+1} \subseteq R_l$. First, we take care of some trivial cases. If the constraints c_{+p_i} is redundant (then c_i must also be redundant) then it does not contribute to extreme points. Thus, the relevant case to consider is when c_{+p_i} is not redundant.

Now, we need to only consider those extreme points that lie on the constraint c_{+p_i}, as the other extreme points for R_{l+1} will be same as for R_l and hence integral. Consider the extreme point p^* that lies on the constraint c_{+p_i}. Wlog, let the variables in c_{+p_i} be $p_1, \ldots, p_i$. We must have $\sum_1^i p_j = |L|$. Now, from the polytope R_l remove all extreme points that have $\sum_1^i p_j = |L|+1$, and consider the convex hull R_l^- of remaining extreme points. Note that all extreme points of R_l^- are in R_{l+1}, thus $R_l^- \subseteq R_{l+1}$. There could be two cases: one if there are no extreme points with $\sum_1^i p_j = |L| + 1$ in R_l, then $R_l = R_l^- \subseteq R_{l+1}$, and since $R_{l+1} \subseteq R_l$ we get $R_l = R_{l+1}$. Thus, all extreme points of R_{l+1} will be $0/1$.

The second (more interesting case) is when there are extreme points in R_l with $\sum_1^i p_j = |L| + 1$ that get removed in R_l^-. Since p^* is a point in R_l it can be written as convex combination of extreme points. If all these extreme points satisfy $\sum_1^i p_j \leq |L|$, then p^* can be written as convex combination of extreme points of R_l^- and since $R_l^- \subseteq R_{l+1}$, p^* cannot be a extreme point. Or else, if one of the extreme points (in the convex combination) P has $\sum_1^i p_j = |L| + 1$ with $p_s = 1$ ($1 \leq s \leq i$) then we can write P as sum of two points within R_l^-: one which is same as P except

$p_s = 0$, and other in which only $p_s = 1$ and other components are zero. It is easy to check that these points are in R_l^-, since the inequalities allow for reducing any p_j and clearly $\sum_1^i p_j \leq |L|$. In such a manner, we can ensure that all extreme points involved in the convex combination have $\sum_1^i p_j \leq |L|$, which reduces to the previous case.

Extreme points are pure strategies

Next, it is easy to see that for given p_i's and p_i^*'s, with corresponding feasible p_i^j's and p_i^{*j}'s, any convex combination $p_i^{@}$'s of p_i's and p_i^*'s has a feasible solution $p_i^{@j}$'s which is the convex combination of p_i^j's and p_i^{*j}'s. Since, any point in a convex set can be written as the convex combination of its extreme points (), it is enough to show the existence of feasible p_i^j's for the extreme points in order to prove existence of feasible p_i^j's for any point in the convex polytope under consideration.

The extreme points of the given convex polytope has ones in $k' \leq k$ positions and all other zeros. The $k' \leq k$ arises due to the constraint $p_1 + \ldots + p_n \leq k$. Consider the undirected bipartite graph linking the inspections node to the target nodes, with a link indicating that the inspection can audit the linked target. This graph is known from our knowledge of R, and each link in the graph can be labeled by one of the p_i^j variables. Let S' be the set of targets picked by the ones in any extreme points. We claim that there is a perfect matching from S' to the the set of inspection resources (which we prove in next paragraph). Given such a perfect matching, assigning $p_i^j = 1$ for every edge in the matching yields a feasible solution, which completes the proof.

We prove the claim about perfect matching in the last paragraph. We do so by contradiction. Assume there is no perfect matching, then there must be a set $S'' \subseteq S'$, such that $|N(S'')| < |S''|$ (N is neighbors function, this statement holds by the well known Hall's theorem). As $S'' \subseteq S'$ it must hold that $p_i = 1$ for all $i \in index(S'')$ (function index gives the indices of the set of targets).. Also, the set of targets S'' is audited only by inspection resources in $N(S'')$ and, by definition of C we must have a constraint (the function index gives the set of indices of the input set of targets)

$$\sum_{i \in index(S'')} p_i \leq |N(S'')| \,.$$

Using, $|N(S'')| < |S''|$, we get

$$\sum_{i \in index(S'')} p_i < |S''| \,.$$

But, since $|index(S'')| = |S''|$, we conclude that all p_i for targets in S'' cannot be one, which is a contradiction. □

Proof of Lemma 2. Assume constraint c is in the output of the naive algorithm. Restrict the constraint c to be a non-implied constraint, i.e., if c is given by $LHS(C) \leq RHS(c)$ then it cannot be written as $\sum_{i=1}^n LHS(c_i) \leq \sum_{i=1}^n RHS(c_i)$ for any $c_1, \ldots, c_n$. Note that such a restriction does not change the region defined by the set C. By the description of the naive algorithm, this constraint must correspond to a set of inspection resources, say S, and the set of targets T_S inspected only by inspection resources in S, and there exists no subset of T_S and S such that the subset of targets is inspected only by the subset of inspection resources. The constraint c is of the form $P(T_S) \leq |S|$, where $P(T_S)$ is the sum of the probability variables for targets in T_S. Now, for the intersection graph representation, every node x represents targets that are inspected by a given subset x_s of inspection resources. For our case, we claim that every $t \in T_S$ is either equivalent to or linked to another target in T_S, otherwise we have the subset $\{t\} \subset S$ inspected only by $t_s \subset T_S$. Thus, the targets in T_S form a connected induced sub-graph. Thus, we conclude that the CONSTRAINT_FIND algorithm will consider this set of targets and find the non-implied constraint c.

Next, assume that CONSTRAINT_FIND finds a constraint c. As this corresponds to a connected induced subgraph (with targets as nodes), we obtain a set of targets T and a set of inspection resources $\cup_{t \in T} F(t)$ such that targets in T are audited by $\cup_{t \in T} F(t)$ only. Thus, by definition of the construction of c, this constraint is same as the constraint c' obtained by the naive algorithm when it considers the set $\cup_{t \in T} F(t)$. That is, c will also be found by the naive algorithm. □

Proof of Lemma 3. The proof below lists sufficient conditions under which the number of induced connected subgraphs is polynomial is size. The proofs are constructive also, allowing extracting these sub-graphs in polynomial time.

- Graphs with $O(\log n)$ nodes. The maximum number of connected induced subgraphs in a graph with t nodes is 2^t (take any subset of vertexes). Thus, clearly a graph with $O(\log n)$ nodes will have polynomially many connected sub-graphs.
- Graphs with constant max degree and constant number of nodes with degree ≥ 3. The number of connected induced subgraphs in a tree with max degree d and t vertexes with degree ≥ 3 is bounded from above by $2^{(2(d+1))^{t+1}} n^{(d+1)^{t+1}}$. To prove this result, denote by $T(n, d, t)$ the worst case number of connected induced sub-graphs in a graph with n vertices, and max degree d and t vertices with degree ≥ 3.
  Remove a vertex X with degree ≥ 3 to get $k \leq d$ disconnected components. Each connected sub-graph in any component that was linked to X could be combined with any connected sub-graph of any other component linked to X, yielding a new connected sub-graph. Thus, considering every subset of the k components, we get

  $$T(n, d, t) \leq 2^k \left(T(n-1, d, t-1)\right)^k$$

  Observing that $k < d+1$, and $T(n-1, d, t-1) \leq T(n, d, t-1)$ we get

  $$T(n, d, t) \leq \left(2T(n, d, t-1)\right)^{d+1}$$

  Thus, we see that $T(n, d, t) = 2^{(2(d+1))^{t+1}} (n)^{(d+1)^{t+1}}$ satisfies the above equation.
  As part of the induction, the base case requires reasoning about a graph with max degree either 1 or 2 ($t = 0, d = 1$

or 2). Thus, we need to show that the connected sub-graphs is less than n^{d+1}, which is n^2 for max-degree 1 and n^3 for max-degree 2. The max-degree 1 case is trivial. For the max-degree 2 case, such graphs can be decomposed efficiently into paths and cycles, and the number of connected sub-graphs on cycles and paths is less that n^2.

□

Proof of Lemma 4. First, assume p_i^o's and x_i^o's are an optimal point of PX_n. If $x_n^o > 0$ then reducing the value of x_n^o always yields a feasible point, as the quadratic inequality is still satisfied. Also, clearly reducing x_n^o increases the objective value. Thus, we must have $x_n^o = 0$.

Next, reducing the value of p_n^o by ϵ_p ($\leq \epsilon$) always yields a feasible point, as the quadratic inequality is still satisfied and so are the linear inequalities. The values of ϵ_p are chosen so that the new values of p_n lie on our discrete grid for p_n. Thus, the new feasible point F is p_i^o's for $i = 1$ to $n-1$, x_i^o' for $i = 1$ to $n-1$ and $p_n = p_n^o - \epsilon_p$, $x_n = 0$. The objective at this feasible point F is off from the optimal value by a linear combination of ϵ_p with constant coefficients, which is less than a constant times ϵ. Then, the SOCP with the new values of p_n yields an objective value at least as high as the feasible point F on the grid. Thus, using our approach, we obtain a solution that differs from the optimum only by $\Theta(\epsilon)$. □

Proof of Lemma 5. observe that the quadratic inequality can be rewritten as

$$p_n(x + \Delta_n) + \delta_{i,n} \leq p_i(x + \Delta_i) \ .$$

Suppose is an optimal point, and suppose for some j we have the strict inequality

$$p_n^*(x^* + \Delta_n) + \delta_{j,n} < p_j^*(x^* + \Delta_j) \ .$$

Let $p'_j = \min(0, \frac{p_n^*(x^*+\Delta_n)+\delta_{j,n}}{x^*+\Delta_j})$. Then, we claim that $p_1^*, \ldots, p_{j-1}^*, p'_j, p_{j+1}^*, p_n^*, x^*$ is also an optimal point. This is easy to see since decreasing p_j is not restricted by any inequality other than the quadratic inequality and the objective only depends on p_n. As a result, we can restrict the problem to be equalities for all those quadratic constraints for which $p_n^*(x^* + \Delta_n) + \delta_{j,n} \geq 0$, and restrict $p_j = 0$ in case $p_n^*(x^* + \Delta_n) + \delta_{j,n} < 0$. □

Proof of Lemma 6. The fixed x case is obvious, since the problem is a linear program for fixed x. The fixed p_n case is equivalent to minimizing x with constraints that are of the form $f(x) \leq 0$, where f is a polynomial in x. For any polynomial constraint $f(x) \leq 0$, it is possible to approximate the roots of the polynomial with an additive approximation factor of 2^{-l} (Schönhage 1982) and hence find the intervals of x that satisfies the constraint $f(x) \leq 0$ within additive approximation factor of 2^{-l}. Doing so for the polynomially many constraints and finding the intersection of intervals yields the minimum value of x with additive approximation factor of 2^{-l} that satisfies all constraints. □

$\mathbf{U_D^a}$	$\mathbf{U_D^u}$	$\mathbf{U_A^a}$	$\mathbf{U_A^u}$
0.614	0.598	0.202	0.287
0.719	0.036	0.869	0.999
0.664	0.063	0.597	0.946
0.440	0.322	0.023	0.624
0.154	0.098	0.899	0.902
0.507	0.170	0.452	0.629
0.662	0.371	1.000	0.999

Table 2: Utility values for the counterexample

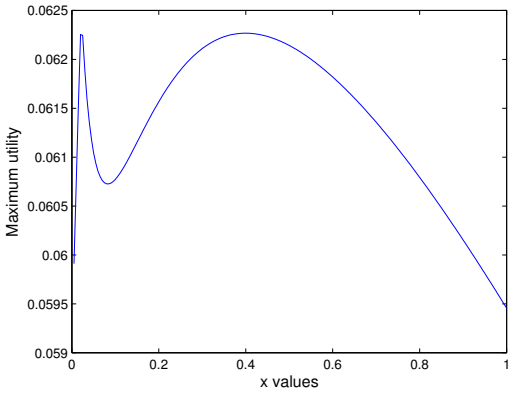

Figure 1: Variation of maximum utility with x showing multiple peaks.

Proof of Lemma 7. We can derive an easy contradiction for this scenario. Assume p_n^*, x is optimal. As f_b is continuous in x, if all inequalities are strict, then it is possible to increase the value of optimal p_n^* by a small amount, without violating the constraints. Clearly, this increased value of p_n results in a higher objective value, contradicting the assumption that p_n^*, x^* is optimal.

□

E Maximum Value of Objective is Not Single-peaked

As stated above, the solution of the optimization problem is not single peaked in punishment x. Here we show the counterexample that proves this fact. We choose a problem instance with just one defender resource and 7 targets, and we consider only the case when the seventh target is the target under attack. The value of a was chosen to be 0.01, and x was discretized with interval size of 0.005. The various values of utilities are shown in Table 2. The variation of maximum utility with x is shown in Figure 1. It can be seen that the maximum value is not single peaked in x.

F Model enhancement.

We state an extension to the model that captures immediate losses that the defender suffers by imposing a punishment, for example, firing or suspending an employee requires time and effort to find a replacement. Mathematically, we can account for such loss by including an additional term in the objective of our optimization:

$$p_n(\Delta_{D,n} - a_1 x) - ax$$

where $-a_1 x$ captures the loss from imposing punishment. It is not hard to check that we still get a FPT with the above objective. The reason for that is as because for any discrete change ϵ in x, the term $-p_n a_1 x$ changes by at max $\theta(\epsilon)$ amount given constant bit precision.

We also still get the same FPTAS result by a minor modification to Lemma 5. In that lemma, we need to consider the case of $\Delta_{D,n} - a_1 x < 0$ separately, and it is not too hard to see that in this case the optimal p_n is 0. We can solve this case of $p_n = 0$ using Lemma 6.

For target-specific punishment the objective would change to

$$p_n(\Delta_{D,n} - a'_n x_n) - \sum_{j \in \{1,\ldots,n\}} a_j x_j$$

Again, using the exact same argument as in Lemma 4, it is not hard to see that at the optimal point x_n should be 0. Once x_n is 0, the objective is linear in the variables and we can apply the same SOCP reduction as earlier.