Autonomous Agents and Multi-Agent Systems

A Game-Theoretic Approach for Selecting Optimal Time-Dependent Thresholds for Anomaly Detection

Amin Ghafouri · Aron Laszka · Waseem Abbas · Yevgeniy Vorobeychik · Xenofon Koutsoukos

Received: July 26, 2018 / Accepted: May 8, 2019

Abstract Adversaries may cause significant damage to smart infrastructure using malicious attacks. To detect and mitigate these attacks before they can cause physical damage, operators can deploy anomaly detection systems (ADS), which can alarm operators to suspicious activities. However, detection thresholds of ADS need to be configured properly, as an oversensitive detector raises a prohibitively large number of false alarms, while an undersensitive detector may miss actual attacks. This is an especially challenging problem in dynamical environments, where the impact of attacks may significantly vary over time. Using a game-theoretic approach, we formulate the problem of computing optimal detection thresholds which minimize both the number of false alarms and the probability of missing actual attacks as a two-player Stackelberg security game. We provide an efficient dynamic programming-based algorithm for solving the game, thereby finding optimal detection thresholds. We analyze the performance of the proposed algorithm and show that its running time scales polynomially as the length of the time horizon of interest increases. In addition, we study the problem of finding optimal thresholds in the presence of both random faults and attacks. Finally, we evaluate our result using a case study of contamination attacks in water networks, and show that

A. Ghafouri, Y. Vorobeychik, and X. Koutsoukos
Institute for Software Integrated Systems,
Vanderbilt University, Nashville, TN, USA.
E-mail: {amin.ghafouri, yevgeniy.vorobeychik, xenofon.koutsoukos}@vanderbilt.edu

A. Laszka
Department of Computer Science,
University of Houston, TX, USA.
E-mail: alaszka@uh.edu

W. Abbas
Department of Electrical Engineering,
Information Technology University, Lahore, Pakistan.
E-mail: w.abbas@itu.edu.pk

our optimal thresholds significantly outperform fixed thresholds that do not consider that the environment is dynamical.

Keywords Anomaly detection systems · Game theory · Smart infrastructure · Stackelberg security game · Random faults

1 Introduction

Smart infrastructures equipped with data-gathering devices and computational capabilities for data-intensive analysis lead to efficient monitoring and management of cyber-physical systems including transportation, electrical, and water distribution systems. The ability to collect diverse data at low-cost allows for intelligent system monitoring, automation, and efficient resource management. Continuous monitoring of modern infrastructure networks to detect anomalies and malicious intruders is a prominent requirement for smart operations. Inability to early detect a malicious attack on some system component might not only cause disruption of services but could lead to complete system failure, excessive physical and financial losses. For instance, in water networks, water pipes are exposed to the risk of intentional contamination with toxic chemicals. If not detected early, such a malicious attack may have detrimental consequences, including poisoning and propagation of infectious diseases.

Efficient intrusion and attack detection mechanisms need to be employed to *quickly* and *accurately* detect attacks. Attackers, on the other hand, strive to maximize the damage inflicted to the system while remaining covert and not getting detected for an extended duration of time. An anomaly detection system (ADS) can monitor the system for signatures of known attacks or for anomalies. When an ADS detects suspicious activity, it raises an alarm, which can then be investigated by system operators and experts. For instance, in the case of water networks, water quality sensors continuously monitor parameters such as chlorine, pH, and turbidity. The collected data is then analyzed by detection systems such as CANARY [15] to detect anomalous events and provide an indication of potential contamination.

A well-known method that can be used for detecting anomalies is sequential change detection [5]. This method considers a sequence of measurements that starts under the normal hypothesis and then, at some point in time, changes to the anomaly hypothesis. In sequential change detection, the *detection delay* is the time difference between when an anomaly occurs and when an alarm is raised. Detection algorithms may induce *false positives* that are alarms raised for normal system behavior. In general, it is desirable to reduce detection delay as much as possible while maintaining an acceptable false-positive rate. There exists a trade-off between the detection delay and the rate of false positives, which can be controlled by changing the sensitivity of the detector. A typical way to control the sensitivity is by changing the detection threshold. By decreasing (increasing) the detection threshold, a defender can decrease (increase) the detection delay and increase (decrease) the false-positive rate.

Consequently, the detection threshold must be carefully selected, since a large value may result in large detection delays, while a small value may result in wasting resources on investigating false alarms.

Finding an *optimal threshold*, which optimally balances the trade-off between detection delay and rate of false positives is a challenging problem. The problem is exacerbated when detectors are deployed in systems with dynamic behavior and when the expected damage incurred from undetected attacks depends on the system state and time. For example, in water distribution networks, contamination attacks at a high-demand time are more calamitous than attacks at a low-demand time. Hence, defenders need to incorporate time-dependent information in computing optimal detection thresholds when facing strategic attackers. In dynamic systems, potential damage from attacks changes over time, which implies that optimal thresholds must also change with time. However, if we have to select a different threshold for each time period, then the number of possible solutions grows exponentially with the time-horizon.

An adversary can attack a system in multiple ways, and each of these may cause a different amount of damage or may be detected with a different delay. To account for these differences, attack types available to the adversary must be explicitly modeled. For instance, in water-distribution networks, potassium ferricyanide and arsenic trioxide are both chemicals that can be used to contaminate water. In this case, addition of a specific toxic chemical constitutes an attack type as each chemical affects water quality in different ways and hence may cause different damage or may be detected with different delay [14].

Contamination events may also occur due to non-malicious incidents or equipment failures. For instance, pipe bursts and leakages can become a source of water contamination. Therefore, it is desirable to design ADS that are able to quickly and accurately detect either incidental contaminations or malicious attacks.

We study the problem of finding optimal thresholds for anomaly-based detection in dynamical systems in the face of strategic attacks.[1] Our main contributions are the following:

– We formulate a two-player Stackelberg game between a defender and an adversary. We assume that the adversary attacks the system, choosing the time and type of the attack (e.g., type of harmful chemical introduced into a water-distribution network) to maximize the inflicted damage. On

[1] This work is a significant extension of our conference paper [11], which appeared at the 7th Conference on Decision and Game Theory for Security (GameSec 2016). The novel contributions are the following: (1) extended model that considers multiple attack types, which can be used to represent, for example, multiple targets within a system that an adversary may attack or multiple choices for the magnitude of the attack; (2) novel polynomial-time algorithms and theoretical analysis for finding optimal detector configurations against multiple attack types; (3) extended model that considers both intentional attacks and random faults (e.g., reliability failures that occur at random) and novel algorithms for finding optimal detection thresholds in the presence of both attacks and random faults; (4) comprehensive numerical results based on real-wold data and simulations, which study multiple attack types, random faults, sensitivity analysis, etc.

the other hand, the defender selects detection thresholds to minimize both damage from best-response attacks and the cost of false alarms.
- We present a dynamic-programming based algorithm to solve the game, thereby computing optimal time-dependent thresholds. We call this approach the *time-dependent threshold* strategy. We analyze the performance of the proposed algorithm and show that its running time scales polynomially as the length of the time horizon of interest increases, which is important in practice from the perspective of scalability.
- We also provide and study a polynomial-time algorithm for the problem of computing optimal *fixed thresholds*, which do not change with time.
- In addition, we study the problem of finding optimal thresholds in the presence of random faults and attacks, and present an algorithm that computes the optimal thresholds. The running time of the algorithm scales polynomially as the length of the time horizon of interest increases.
- Finally, we evaluate and apply our results to the detection of contamination attacks in a water-distribution system as a case study. Since expected damage to the system by an attack is time-dependent as water demand changes throughout the day, the time-dependent threshold strategy can achieve much lower losses than a fixed-threshold strategy. Our simulation results confirm this, showing that time-dependent thresholds significantly outperform fixed ones.

2 Related Work

The problem of optimal design of anomaly detection systems has been studied in a variety of different ways in the academic literature [31, 7]. Nevertheless, to the best of our knowledge, prior work has not specifically addressed the optimal threshold selection problem in the face of strategic attacks when the damage corresponding to an attack depends on time-varying properties of the underlying system.

Change detection methods with adaptive thresholds have been previously used. An extension of CUSUM test that can be configured at run-time is proposed in [1]. The paper discusses methods to configure the detector's parameters, and shows how the detector performs when the correct configuration is not known a priori. Further, a procedure to obtain adaptive thresholds for CUSUM-type detectors is presented that takes into account non-stationary nature of the stochastic systems under supervision [33]. The proposed method outperforms fixed threshold in obtaining desired rate of false alarms. Finally, an adaptive CUSUM control chart is presented that uses variable sampling intervals [19]. The method is shown to perform better than the fixed sampling interval approach. Nonetheless, unlike our work, these studies fail to address dependencies between the detector's performance and dynamic properties of a system that can be maliciously exploited by strategic adversaries.

There have been several distinct efforts involving game-theoretic modeling of anomaly detection. The first is signaling games used to model intrusion de-

tection [25, 10]. For example, an intrusion detection game based on a signaling game is proposed in order to select the optimal detection strategy that lowers resource consumption [29]. Further, distributed intrusion detection is studied as a game between an IDS and an attacker using a model that represents the flow of information from the attacker to the IDS [2, 3]. The work investigates the existence of a unique Nash equilibrium and best-response strategies. Nevertheless, the IDS models used in these works are significantly different from the ones used in our work (i.e., anomaly-based change detection). Another related game-theoretic setting is FlipIt game [32, 18]. FlipIt is an attacker-defender game that studies the problem of stealthy takeover of control over a critical resource, in which the players receive benefits proportional to the total time that they control the resource. A framework for the interaction between an attacker, defender, and a cloud-connected device is presented in [26]. The interactions are described using a combination of a FlipIt game and a signaling game. What distinguishes our work from FlipIt is using an anomaly detector that has detection delay and false alarms. Finally, our work is related to the broad literature on Stackelberg security games [24, 17, 30], although our particular problem and model are novel in that context.

Contaminant intrusion in water distribution network has been considered in water security literature [8, 9]. In particular, data-driven water monitoring approaches have received considerable attention due to the advances in smart monitoring technologies [22, 16]. Bayesian sequential analysis is integrated with neural network models to detect possible quality threats in water distribution systems [28]. Further, a dynamic thresholds scheme for contamination event detection is presented by defining optimal detection thresholds as the ones that maximize detection rate [4]. While the mentioned work also uses detection thresholds that change in time, the method of threshold selection does not consider losses obtained by detection delay and false alarms. In addition, unlike our work, it does not consider malicious adversaries that exploit time-varying aspects of WDS.

Sequential change detection methods such as CUSUM have been used to detect changes in water quality. Combined Shewhart-CUSUM control charts are used for ground water monitoring in [12]. The study uses Shewhart control chart for identifying large changes at a single timestep in addition to CUSUM chart for detecting small continuous changes. The method is evaluated by presenting false-positive rate, false-negative rate, and detection delay. Further, CUSUM methods for water quality monitoring are implemented in [20]. Considering six kinds of quality trends, the performance of CUSUM is studied by measuring detection delay and false-negative error. It is concluded that CUSUM performs well when used for monitoring water quality. While such studies effectively use sequential change detection for water quality monitoring, they simply use fixed thresholds and do not consider time-dependent thresholds. In this work, we show that time-dependent thresholds significantly outperform the fixed threshold in terms of minimizing the losses.

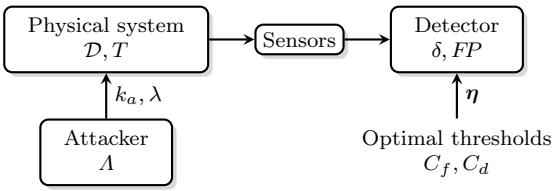

Fig. 1: System description.

3 System Model

We consider a system which may be attacked by an adversary. We assume a discrete-time system model with a finite time horizon of interest denoted by $\{1, ..., T\}$. The system provides some utility in its normal state and this utility is substantially reduced when the system is under attack. Further, the system and hence these utilities may be be time-dependent. Instead of explicitly considering these quantities, we take a general, security-focused approach and model the impact of attacks using a time-dependent damage function $\mathcal{D}$. Finally, we assume that the system is monitored by a set of sensors and an operator can use anomaly detection based on sensor data for detecting attacks.

For example, consider a water distribution network that is monitored by sensors that measure water quality using pH or choline levels. The system is subject to attacks such as intrusive contamination with toxic chemicals [13]. The utility from supplying clean water for residential consumption depends on the water demand, which fluctuates significantly over time. The damage caused by a contamination attack depends on both the lack of clean water supply as well as the impact on public health of the population exposed to contaminated water. Water quality sensors may be used to detect anomalies, such as changes in chemical concentrations that could be attributed to the introduction of harmful chemicals.

Our primary goal is to address the problem of *finding optimal time-dependent configurations* for anomaly detection algorithms. Table 1 shows a list of symbols used in this paper. In addition, Figure 1 shows a high level overview of the system model, whose elements will be detailed in the following subsections.

Attack Model. Adversaries may compromise the system through an attack of type $\lambda \in \Lambda$ (e.g., type of harmful chemical introduced into a water-distribution network). The attack starts at time k_a and ends at k_e, thus spanning the interval $[k_a, k_e]$. If an attack remains undetected, it will enable the attacker to cause physical or financial damage. In order to represent the tight relation between the system's dynamic behavior and the expected loss incurred from undetected attacks, we model the potential damage of an attack as a function of time.

Definition 1 (Expected Damage Function) The damage function of a system is a function $\mathcal{D}: \{1, ..., T\} \times \Lambda \to \mathbb{R}_+$ which represents the expected

Table 1: List of Symbols

Symbol	Description
T	cardinality of time horizon of interest
$\boldsymbol{\eta}$	vector of time-dependent threshold $\boldsymbol{\eta} = \langle \eta_k \rangle_{k=1}^{T}$
Λ	set of attack types
$\mathcal{D}(k, \lambda)$	expected damage caused by an attack of type $\lambda \in \Lambda$ at timestep k
$\delta(\eta_k, \lambda)$	detection delay given detection threshold η_k and attack type λ
$FP(\eta_k)$	false alarm probability given detection threshold η_k
C_f	cost of false alarms
C_d	cost of changing the detection threshold
$\mathcal{P}(\boldsymbol{\eta}, k_a, \lambda)$	attacker's payoff for time-dependent threshold $\boldsymbol{\eta} = \langle \eta_k \rangle_{k=1}^{T}$ and attack (k_a, λ)
$\mathcal{L}(\boldsymbol{\eta}, k_a, \lambda)$	defender's loss for threshold $\boldsymbol{\eta} = \langle \eta_k \rangle_{k=1}^{T}$ and attack (k_a, λ)
E	set of thresholds corresponding to set of possible detection delays Δ
$\mathcal{P}_F(\boldsymbol{\eta})$	defender's loss for time-dependent threshold $\boldsymbol{\eta} = \langle \eta_k \rangle_{k=1}^{T}$ due to random faults
$\mathcal{L}_C(\boldsymbol{\eta}, k_a, \lambda)$	defender's loss for threshold $\boldsymbol{\eta} = \langle \eta_k \rangle_{k=1}^{T}$ due to random faults and attack (k_a, λ)

damage $\mathcal{D}(k, \lambda)$ incurred by the system from an undetected attack of type $\lambda \in \Lambda$ at time $k \in \{1, ..., T\}$.

Detector. We consider a defender whose objective is to protect the system using anomaly detection based on the sensor measurements. The detector's goal is to determine whether a sequence of received measurements corresponds to normal behavior or an attack. Although the proposed approach can be used for various detection algorithms, we consider a widely used method known as sequential change detection [5]. This method assumes a sequence of measurements that starts under the normal hypothesis, and then, at some point in time, changes to the attack hypothesis. Change detection attempts to detect this change as soon as possible. Examples of change detection algorithms are geometric moving average, generalized likelihood ratio (GLR), and cumulative sum (CUSUM) [5].

The performance of change detectors is characterized by the *detection delay*, which is the time between the beginning of an attack and the time when an alarm is raised, and the *false-positive probability*, which is the probability of raising an alarm when there has been no attack. In general, it is desirable to reduce detection delay while maintaining an acceptable false-positive probability. However, there exists a trade-off between the detection delay and the probability of false positives, which can be controlled by changing the detection threshold. In particular, by decreasing (increasing) the detection threshold, a defender can decrease (increase) the detection delay and increase (decrease) the false-positive probability. Finding the optimal trade-off and its corresponding *optimal threshold* is an important problem since the damage from an attack depends on the performance of the detector.

The time-dependent threshold is denoted by $\boldsymbol{\eta} = \langle \eta_k \rangle_{k=1}^{T}$ and the detection delay by $\delta : \mathbb{R}_+ \times \Lambda \to \mathbb{N} \cup \{0\}$, where $\delta(\eta, \lambda)$ is the detection delay (in timesteps) when the threshold is $\eta \in \mathbb{R}_+$ and the type of the attack is $\lambda \in \Lambda$. The rationale behind this model of delay is that while a certain threshold might not detect an attack immediately after it happens, the same threshold might detect the attack later. For example, to detect an attack of type λ using a CUSUM-based detector, enough error has to accumulate to reach a threshold η, which takes a certain number of timesteps $\delta(\eta, \lambda)$. We assume that for each $\lambda \in \Lambda$, $\delta(\eta, \lambda)$ is a left-continuous function of η. [2] Further, we denote the false-positive probability (i.e., probability of raising a false alarm during a single timestep) by $FP : \mathbb{R}_+ \to [0, 1]$, where $FP(\eta)$ is the false-positive probability when the detection threshold is η. We assume that FP is decreasing and δ is non-decreasing with respect to η, which is true for most typical detectors, including sequential change detectors. For example, in Section 7, we obtain detection delay and false-positive probability for a CUSUM detector.

4 Problem Statement

In this section, we present the optimal threshold selection problem. We consider the case in which the defender selects time-dependent thresholds for the anomaly detection. We model this problem as a conflict between a defender and an attacker, which is formulated as a two-player Stackelberg security game.

The idea of time-dependent threshold is to reduce the detector's sensitivity during less critical periods (via increasing the threshold) and increase the sensitivity during more critical periods (via decreasing the threshold). As we will show, this significantly decreases the loss corresponding to false alarms. However, the defender may not want to continuously change the threshold, since a threshold change requires a reconfiguration of the detector that has a cost. Hence, the defender needs to find an *optimal threshold*, which is a balance between continuously changing the threshold and keeping it fixed.

Defender's Loss and Attacker's Payoff. The defender's strategic choice is to select the threshold $\boldsymbol{\eta} = \langle \eta_k \rangle_{k=1}^{T}$ for each timestep. We consider a worst-case attacker who will not stop the attack before detection in order to maximize the damage. Consequently, the attacker's strategic choice becomes to select an attack type λ and a time k_a to start the attack.

Since our work focuses on optimizing detection delay, we consider damage arising from attacks only during the time they remain undetected. In other words, we consider the impact of an attack from its beginning until its detection. We define the detection time $\sigma(\boldsymbol{\eta}, k_a, \lambda)$ of an attack of type λ that starts

[2] We assume that $\delta(\eta, \lambda)$ is left continuous to ensure that the optimal thresholds exist (see Definition 3). Without this assumption, the loss $\mathcal{L}(\boldsymbol{\eta}, k_a, \lambda)$ would have an infimum but not necessarily a minimum. Similarly, the maximum thresholds in Equation (4) would not necessarily exist. Since these phenomena have virtually no practical relevance (in practice, values will typically be represented as floating-point numbers with limited precision), we introduce the mild assumption of left continuity for ease of presentation.

at k_a as the first timestep in which an alarm is raised due to the attack. Since an alarm is raised in timestep k for an attack of type λ that started at k_a if and only if $\delta(\eta_k, \lambda) \leq k - k_a$, the detection time of an attack is

$$\sigma(\boldsymbol{\eta}, k_a, \lambda) = \{\min k \,|\, \delta(\eta_k, \lambda) \leq k - k_a\} \; .$$

Note that the equation above represents the timestep at which the attack is first detected, and not the detection delay.

For the strategies $(\boldsymbol{\eta}, k_a, \lambda)$, the attacker's payoff is the total damage until the expected detection time,

$$\mathcal{P}(\boldsymbol{\eta}, k_a, \lambda) = \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda) \; , \tag{1}$$

that is, the total damage incurred by the system until the expected detection time. This payoff function assumes a worst-case attacker that has the goal of maximizing the damage.

If an alarm is raised, the defender needs to investigate the system to determine whether an attack has actually occurred or not, which will cost C_f. Further, let C_d be the cost associated with each threshold change. The number of threshold changes is described by $N(\boldsymbol{\eta}) = |\Gamma|$, where $\Gamma = \{k \,|\, \eta_k \neq \eta_{k+1}, k \in \{1, ..., T-1\}\}$. When the defender selects a time-dependent threshold $\boldsymbol{\eta}$, and the attacker starts the attack at a timestep k_a, the defender's loss (i.e., inverse payoff) is

$$\mathcal{L}(\boldsymbol{\eta}, k_a, \lambda) = N(\boldsymbol{\eta}) \cdot C_d + \sum_{k=1}^{T} C_f \cdot FP(\eta_k) + \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda) \; , \tag{2}$$

that is, the amount of resources spent on changing the threshold, operational costs of manually investigating false alarms, and the expected amount of damage caused by the attack before its detection.

Best-Response Attack and Optimal Threshold. We assume that the attacker has complete and perfect information, and will play a *best-response* attack to the defender's strategy as defined below.

Definition 2 (Best-Response Attack) Assuming a defender's strategy, the attacker's strategy is a best-response if it maximizes the attacker's payoff. Formally, an attack (k_a, λ) is a best-response given a defense strategy $\boldsymbol{\eta}$ if it maximizes $\mathcal{P}(\boldsymbol{\eta}, k_a, \lambda)$ as defined in (1).

Further, the defender must choose his strategy expecting that the attacker will play a best-response or uniformly random attack. We formulate the defender's optimal strategy as a strong Stackelberg equilibrium (SSE), which is commonly used in the security literature for solving Stackelberg games [17].

Definition 3 (Optimal Thresholds) We call a defense strategy *optimal* if it minimizes the defender's loss given that the attacker always plays a best-response with tie-breaking in favor of the defender. Formally, an optimal defense is

$$\arg\min_{\substack{\boldsymbol{\eta},\\ (k_a,\lambda)\in \text{bestResponses}(\boldsymbol{\eta})}} \mathcal{L}(\boldsymbol{\eta}, k_a, \lambda), \tag{3}$$

where bestResponses($\boldsymbol{\eta}$) are the best-response attacks against $\boldsymbol{\eta}$.

5 Selection of Optimal Thresholds

In this section, we present an approach for computing optimal thresholds for any instance of the attacker-defender game, based on the SSE. The approach consists of two steps: 1) a dynamic-programming algorithm (Algorithm 1) for finding minimum-cost thresholds subject to the constraint that the damage caused by a best-response attack is lower than or equal to a given damage bound and 2) an exhaustive-search algorithm (Algorithm 2) that finds an optimal damage bound and thereby optimal thresholds.

Let Δ denote the set of all possible detection delay values:

$$\Delta = \left\{ m \in \{1, \dots, T\} \;\middle|\; \exists \lambda \in \Lambda, \eta \in \mathcal{R}_+ \left[m = \delta(\eta, \lambda) \right] \right\}.$$

In other words, Δ is the set of all delay values between 1 and T that can be attained by some threshold η for some attack type λ.

Next, let E be the set of maximal threshold values that attain the delay values Δ:

$$E = \left\{ \eta^* \;\middle|\; \exists \lambda \in \Lambda, m \in \Delta \left[\eta^* = \max_{\eta \,:\, \delta(\eta,\lambda) \le m} \eta \right] \right\}. \tag{4}$$

Introducing the set E enables us to restrict the strategy set of the defender to a discrete set. The following lemma shows that the defender can always find optimal thresholds by considering only threshold values from the set E.

Theorem 1 *Given an instance of our game, there exist optimal thresholds* $\boldsymbol{\eta}$ *such that*

$$\forall k \in \{1, \dots, T\} : \; \eta_k \in E.$$

The intuition behind Theorem 1 is that any threshold value $\eta_k \notin E$ can be replaced with $\eta_k^* \in E$ such that $\delta(\eta_k^*, \lambda) = \delta(\eta_k, \lambda)$, and this replacement cannot increase attack damage (since detection delay δ remains the same), but it may decrease false-alert losses (since threshold η_k may increase). We provide a formal proof in Appendix A.1.

Consequently, for the remainder of this paper, we will consider only strategies in which every threshold η_k is chosen from the set E.

Next, we present the algorithm for computing the optimal thresholds. The dynamic-programming algorithm (Algorithm 1) finds minimum-cost thresholds subject to the constraint that the damage caused by a best-response attack is lower than or equal to a given damage bound P. The exhaustive

Algorithm 1 MinimumCostThresholds(P)

$\forall\, \boldsymbol{m} \in \Delta^{|\Lambda|},\ \eta \in E:\ \textsc{Cost}(T+1, \boldsymbol{m}, \eta) \leftarrow 0$
for $n = T, \ldots, 1$ **do**
  for all $\boldsymbol{m} \in \Delta^{|\Lambda|}$ **do**
    for all $\eta_{\text{prev}} \in E$ **do**
      if $\bigvee_{\lambda \in \Lambda} \left(\sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P \right)$ **then**
        $\textsc{Cost}(n, \boldsymbol{m}, \eta_{\text{prev}}) \leftarrow \infty$
      else
        for all $\eta \in E$ **do**
          if $\eta_{\text{prev}} = \eta \vee n = 1$ **then**
            $S(n, \boldsymbol{m}, \eta_{\text{prev}}, \eta)$
            $\leftarrow \textsc{Cost}(n+1, \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}, \eta) + C_f \cdot FP(\eta)$
          else
            $S(n, \boldsymbol{m}, \eta_{\text{prev}}, \eta)$
            $\leftarrow \textsc{Cost}(n+1, \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}, \eta) + C_f \cdot FP(\eta) + C_d$
        $\eta^*(n, \boldsymbol{m}, \eta_{prev}) \leftarrow \arg\min_\eta S(n, \boldsymbol{m}, \eta_{prev}, \eta)$
        $\textsc{Cost}(n, \boldsymbol{m}, \eta_{prev}) \leftarrow \min_\eta S(n, \boldsymbol{m}, \eta_{prev}, \eta)$
$\boldsymbol{m} \leftarrow \langle 0, \ldots, 0 \rangle,\ \eta_0^* \leftarrow$ arbitrary
for all $n = 1, \ldots T$ **do**
  $\eta_n^* \leftarrow \eta^*(n, \boldsymbol{m}, \eta_{n-1}^*)$
  $\boldsymbol{m} \leftarrow \langle \min\{\delta(\eta_n^*, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}$
return $(\textsc{Cost}(1, \langle 0, \ldots, 0 \rangle, \text{arbitrary}), \boldsymbol{\eta}^*)$

Algorithm 2 OptimalThresholds

$\mathit{SearchSpace} \leftarrow \left\{ \sum_{k=k_a}^{k_a+\delta} \mathcal{D}(k, \lambda) \;\middle|\; \exists\, k_a \in \{1, \ldots, T-1\},\ \delta \in \Delta,\ \lambda \in \Lambda \right\}$
for all $P \in \mathit{SearchSpace}$ **do**
  $(TC(P), \boldsymbol{\eta}^*(P)) \leftarrow \textsc{MinimumCostThresholds}(P)$
$P^* \leftarrow \arg\min_{P \in \mathit{SearchSpace}} TC(P) + P$
return $\boldsymbol{\eta}^*(P^*)$

search (Algorithm 2) computes the optimal thresholds by finding an optimal damage bound P and using Algorithm 1.

In the first algorithm (Algorithm 1), we use a dynamic-programming approach, iterating backwards through the timesteps. The key idea of our approach is that we can compute optimal thresholds for timesteps $n, n+1, \ldots, T$ without knowing the preceding thresholds $1, 2, \ldots, n-1$: we need to know only what undetected attacks may be in progress at the beginning of timestep n. We can describe the state of these attacks by specifying for each attack type $\lambda \in \Lambda$ when in the last Δ timesteps the attacker may have started a yet undetected attack. Consequently, the optimal thresholds for the remaining timesteps depend only on the attack state $\boldsymbol{m} \in \Delta^{|\Lambda|}$, which enables us to formulate the following dynamic-programming algorithm. For each timestep $n \in \{T, \ldots, 1\}$ and attack state $\boldsymbol{m} \in \Delta^{|\Lambda|}$, we assume that the optimal thresholds for the remaining timesteps (for each possible following state) have already been computed, and we compute the optimal threshold η_k for timestep n and attack state $\boldsymbol{m}$ in polynomial-time.

When computing the optimal threshold η_k, we must consider the cost of false alerts in the current timestep n as well as the possible costs in the remaining timesteps $n+1, n+2, \ldots, T$. To keep track of possible costs during backwards induction, we define $\textsc{Cost}(n, \boldsymbol{m})$ to be the minimum attainable cost of false alerts from timesteps n to T subject to the damage bound P, given that attacks of type λ can start at $k_a \in \{n - m_\lambda, \ldots, T\}$ and they are not detected prior to n. Note that for certain values of n and $\boldsymbol{m}$, there exist no thresholds that could satisfy the damage bound P. For such values, we let $\textsc{Cost}(n, \boldsymbol{m})$ be equal to ∞. We will assign ∞ to $\textsc{Cost}(n, \boldsymbol{m})$ directly if values n and $\boldsymbol{m}$ allow an attack to cause more than P damage by timestep n. For values that lead to a violation of the damage bound in a later timestep, we let the backward propagation assign ∞. We can compute cost $\textsc{Cost}(n, \boldsymbol{m})$ recursively as

$$\textsc{Cost}(n, \boldsymbol{m}) = \begin{cases} \infty & \text{if } \bigvee_{\lambda \in \Lambda} \sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P, \\ \min_\eta \; C_f \cdot FP(\eta) + \textsc{Cost}(n+1, & \\ \qquad \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}) & \text{otherwise.} \end{cases}$$

Since cost $\textsc{Cost}(n, \boldsymbol{m})$ depends only on costs $\textsc{Cost}(n+1, \ldots)$, we can compute all cost using backwards induction. Further, by also keeping track of the minimizing threshold η_n^*, our dynamic programming algorithm will readily have the optimal thresholds.

In addition to the cost of false alerts, we also need to minimize the cost of threshold changes. To account for these, we extend the state that is used by our dynamic programming algorithm to also include the threshold of the previous timestep. Formally, we define $\textsc{Cost}(n, \boldsymbol{m}, \eta_{\text{prev}})$ to be the minimum attainable cost for timesteps starting from n subject to the same constraints as before but also given that the threshold value in timestep $n-1$ (i.e., the previous timestep) is η_{prev}. Again, we can compute this cost recursively as

$$\textsc{Cost}(n, \boldsymbol{m}, \eta_{\text{prev}}) = \begin{cases} \infty & \text{if } \bigvee_{\lambda \in \Lambda} \sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P, \\ \min_\eta S(n, \boldsymbol{m}, \eta_{\text{prev}}, \eta) & \text{otherwise.} \end{cases}$$

where

$$S(n, \boldsymbol{m}, \eta_{\text{prev}}, \eta) = \begin{cases} \textsc{Cost}(n+1, \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}) + C_f \cdot FP(\eta) & \text{if } \eta = \eta_{\text{prev}} \vee n = 1, \\ \textsc{Cost}(n+1, \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}) + C_f \cdot FP(\eta) + C_d & \text{otherwise.} \end{cases} \tag{5}$$

Lemma 1 *For any given damage bound $P \in \mathbb{R}$, Algorithm 1 computes thresholds $\boldsymbol{\eta} = \langle \eta_k \rangle_{k=1}^{T}$ that minimize*

$$N \cdot C_d + \sum_{k=1}^{T} C_f \cdot FP(\eta_k)$$

subject to

$$\forall k_a \in \{1, \ldots, T\}, \lambda \in \Lambda : \ \mathcal{P}(\boldsymbol{\eta}, k_a, \lambda) \leq P \,. \tag{6}$$

The algorithm returns the minimum cost attained, or if no thresholds exist satisfying (6)*, it returns infinity as the cost.*

We provide a proof of Lemma 1 and a more detailed discussion of Algorithm 1 in Appendix A.2.

By building on Algorithm 1, our second algorithm (Algorithm 2) finds the optimal damage bound P^* and thereby the optimal thresholds (Definition 3). Recall that Algorithm 1 finds the minimum cost of false alerts and threshold changes subject to the constraint that a best-response attack may cause at most damage P. For a given damage bound P, we let this minimum cost be denoted by $TC(P)$. Then, we can express the problem of finding an optimal damage bound P^* and optimal thresholds as

$$\min_{P} \ TC(P) + P. \tag{7}$$

To find the minimizing damage bound P^*, observe that the amount of damage that an attack may cause belongs to a limited set of attainable damage values. In fact, the amount of damage resulting from any attack (regardless of the thresholds) is necessarily from the following set:

$$\left\{ \sum_{k=k_a}^{k_a+\delta} \mathcal{D}(\lambda, k) \,\middle|\, \exists\, k_a \in \{1, \ldots, T\},\, \delta \in \Delta,\, \lambda \in \Lambda \right\}. \tag{8}$$

Then, it is easy to see that we can find the optimal damage bound P^* by searching over the above set, whose cardinality is polynomial in the size of the input. The following theorem establishes that Algorithm 2 indeed finds optimal thresholds.

Theorem 2 *Algorithm 2 computes optimal thresholds that minimize the defender's loss (see Definition 3).*

We provide a proof of Theorem 2 and a more detailed discussion of Algorithm 2 in Appendix A.3.

Proposition 1 *The running time of Algorithm 2 is* $\mathcal{O}(T^2 \cdot |\Delta|^{|\Lambda|+2} \cdot |\Lambda|^2 \cdot |E|)$.

We provide a proof of Proposition 1 in Appendix A.4. Note that since detection delay values can be upper-bounded by T, the running time of Algorithm 2 is also $\mathcal{O}(T^{|\Lambda|+4} \cdot |\Lambda|^2 \cdot |E|)$.

Finally, note that the running time of the algorithm can be substantially reduced in practice by computing Cost in a lazy manner. Starting from $n = 1$ and $\boldsymbol{m} = \langle 0, \ldots, 0 \rangle$, we can compute and store the value of each $\textsc{Cost}(n, \boldsymbol{m}, \delta_{\text{prev}})$ only when it is referenced, and then reuse it when it is referenced again.

Fixed Detection Thresholds We also present an efficient polynomial-time algorithm to compute the optimal threshold for the special case when the threshold is fixed for the time horizon $\{1, \ldots, T\}$. In this case, a detection threshold is chosen and is kept fixed. Detectors with fixed threshold are widely used in practice and are advantageous when it is not possible to change the threshold due to operational restrictions. To compute an optimal fixed threshold, we use an exhaustive search algorithm, which we formally present as Algorithm 3 in Appendix A.5. The algorithm iterates over all possible threshold values $\eta \in E$ and selects one that minimizes the defender's loss considering a best-response attack. Given a threshold η, to find a best-response attack (k_a, λ), the algorithm iterates over all possible pairs of (k_a, λ), and selects one that maximizes the payoff.

Proposition 2 *Algorithm 3 computes an optimal fixed threshold in* $\mathcal{O}(T \cdot |E| \cdot |\Lambda|)$ *steps.*

We provide a proof of Proposition 2 in Appendix A.5.

6 Optimal Thresholds in the Presence of Faults and Attacks

In this section, we modify our game to take into account random faults and attacks. This is motivated by the fact that contamination may also occur due to non-malicious incidents such as pipe bursts and leakages. Therefore, it is desirable to design anomaly detectors that are able to quickly and accurately detect either random faults or attacks. We formally define random faults as follows.

Definition 4 (Random Fault) A random fault is represented by (k_a, λ) where k_a and λ are randomly selected from uniform distributions over $\{1, \ldots, T\}$ and Λ.

The expected loss from random faults, denoted by $\mathcal{P}_F(\boldsymbol{\eta})$ is the mean of the losses, that is

$$\mathcal{P}_F(\boldsymbol{\eta}) = \frac{1}{T \cdot |\Lambda|} \sum_{k_a=1}^{T} \sum_{\lambda \in \Lambda} \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda). \tag{9}$$

Then, the combined loss due to faults and attacks can be represented as the average of the loss (9) due to random faults and the loss (1) due to attacks:[3]

$$\mathcal{P}_C(\boldsymbol{\eta}, k_a, \lambda) = \frac{1}{2} \left(\mathcal{P}_F(\boldsymbol{\eta}) + \mathcal{P}(\boldsymbol{\eta}, k_a, \lambda) \right). \tag{10}$$

[3] Note that combined loss could be defined as a general linear combination of faults and attacks, i.e., $\mathcal{P}_C = \alpha_F \cdot \mathcal{P}_F + \alpha \cdot \mathcal{P}$, where α_F and α are arbitrary constants. Our results can be extended trivially to cover this more general formulation by simply scaling the constants in our model up or down. For ease of presentation, we consider combined loss to be the average of faults and attacks.

Therefore, the defender's total loss with both random faults and best-response attacks is

$$\mathcal{L}_C(\boldsymbol{\eta}, k_a, \lambda) = N(\boldsymbol{\eta}) \cdot C_d + \sum_{k=1}^{T} C_f \cdot FP(\eta_k) + \mathcal{P}_C(\boldsymbol{\eta}, k_a, \lambda) \tag{11}$$

$$= N(\boldsymbol{\eta}) \cdot C_d + \sum_{k=1}^{T} C_f \cdot FP(\eta_k) + \frac{1}{2}(\mathcal{P}_F(\boldsymbol{\eta}) + \mathcal{P}(\boldsymbol{\eta}, k_a, \lambda)) \tag{12}$$

As before, the defender's problem is to find the thresholds that minimize the loss, that is

$$\underset{\substack{\boldsymbol{\eta}, \\ (k_a,\lambda)\in \text{bestResponses}(\boldsymbol{\eta})}}{\arg\min} \mathcal{L}_C(\boldsymbol{\eta}, k_a, \lambda), \tag{13}$$

Algorithm Here, we show how to generalize Algorithms 1 and 2 to account for random faults. First, we introduce the problem of finding thresholds that minimize costs and damage from random faults subject to the constraint that an attack may cause at most P damage. Formally, we define the following subproblem given a damage bound P:

$$TC_C(P) = \min_{\boldsymbol{\eta}} N(\boldsymbol{\eta}) \cdot C_d + \sum_{k=1}^{T} C_f \cdot FP(\eta_k) + \frac{1}{2} \cdot \frac{1}{T \cdot |\Lambda|} \sum_{k'_a=1}^{T} \sum_{\lambda' \in \Lambda} \sum_{k=k'_a}^{\sigma(\boldsymbol{\eta}, k'_a, \lambda')} \mathcal{D}(k, \lambda')$$

subject to

$$\forall k_a, \lambda : \ \frac{1}{2} \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda) \leq P,$$

We let $TC_C(P) = \infty$ if there exist no k_a and λ that would satisfy the constraint of $TC_C(P)$. Then, it follows from the argument that we presented for Theorem 2 that we can find an optimal damage bound P^* by solving

$$\min_P TC_C(P) + P. \tag{14}$$

Further, it also follows from the same argument that for an optimal damage bound P^*, an optimal solution $\boldsymbol{\eta}^*$ to $TC_C(P^*)$ is also an optimal solution to (13).

Next, to compute $TC_C(P)$, we generalize Algorithm 1 by defining the following sub-subproblem:

$$\begin{aligned}\text{Cost}(P, n, \boldsymbol{m}, \eta_{n-1}) = &\min_{\eta_n, \eta_{n+1}, \ldots, \eta_T} N(\langle \eta_{n-1}, \eta_n, \ldots, \eta_T \rangle) \cdot C_d \\ &+ \sum_{k=n}^{T} C_f \cdot FP(\eta_k) \\ &+ \frac{1}{2} \frac{1}{T \cdot |\Lambda|} \sum_{\lambda' \in \Lambda} \sum_{k'_a = n - m_{\lambda'}}^{T} \sum_{k=n}^{\sigma(\boldsymbol{\eta}, k'_a, \lambda')} \mathcal{D}(k, \lambda')\end{aligned}$$

subject to

$$\forall \lambda, k_a \in \{n - m_\lambda, \ldots, T\} : \sum_{k=k_a}^{\min\{i \mid i \geq n \wedge \delta(\eta_i, \lambda) \leq i - k_a\}} \frac{1}{2} \mathcal{D}(k, \lambda) \leq P,$$

where P is a real number, $n \in \{1, \ldots, T\}$, $\boldsymbol{m}$ is a $|\Lambda|$-element vector of natural numbers, and $\eta_{n-1} \in E$. Clearly, we have $TC_C(P) = \text{Cost}(P, 1, (0, \ldots, 0), \eta_0)$ for any η_0, and an optimal solution to Cost is also an optimal solution to TC_C.

Finally, we show that we can solve Cost using dynamic programming. We let $\text{Cost}(P, n, \boldsymbol{m}, \eta_{n-1}) = \infty$ if there exist no $\eta_n, \eta_{n+1}, \ldots, \eta_T$ that would satisfy the constraint of $\text{Cost}(P, n, \boldsymbol{m}, \eta_{n-1})$. Then, we can break down the computation of Cost as Equation (15), where 1_x is equal to 1 if x is true, and 0 otherwise. The correctness of the reduction follows from the same argument that was presented in Lemma 1.

$$\text{Cost}(P, n, \boldsymbol{m}, \eta_{n-1}) = \tag{15}$$

$$\begin{cases} \infty & \text{if } \bigvee_\lambda \frac{1}{2} \sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P, \\ \min_{\eta_n} \text{Cost}(P, n+1, \langle \min\{\delta(\eta_n, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}, \eta_n) & \\ \quad + 1_{\{\eta_{n-1} \neq \eta_n\}} C_d + C_f \cdot FP(\eta_n) & \\ \quad + \frac{1}{2} \frac{1}{T \cdot |\Lambda|} \sum_{\lambda' \in \Lambda} \sum_{k'_a = n - m_{\lambda'}}^{T} 1_{\{n \leq \sigma(\boldsymbol{\eta}, k'_a, \lambda')\}} \mathcal{D}(n, \lambda') & \text{otherwise.} \end{cases}$$

7 Evaluation

In this section, we evaluate our approach numerically using a case study of detecting contamination attacks in water distribution systems. Ensuring the supply of clean and safe drinking water is mandatory for any water infrastructure. This requires continuous monitoring of water quality parameters and assessing the sensor measurements for any intrusive (or non-intrusive) contamination.

7.1 System Model

We consider a water distribution system (WDS) and a malicious adversary who attempts to penetrate the system through one of many entry points, such as hydrant and connections, and contaminate the water with toxic chemicals [13]. To model normal behavior, we use data collected by a utility in the United States available at [6]. The data contains water quality measurements at a resolution of ten minutes spanning six weeks (i.e., 6048 time steps). All measurements are taken under normal conditions and include the following water quality parameters: Total chlorine, electrical conductivity (EC), pH, total organic carbon (TOC), and turbidity[4]. We divide the data into two subsets, 67% for training and 33% for testing. The training subset is used to construct an estimator used in the detector. The testing subset is used to imitate real-time operation and to evaluate the detector by considering contamination attacks.

Contamination Attack A contamination attack consists of adding harmful contaminants in the water-distribution network to decrease the quality of drinking water below safe levels. As a result of a contamination attack, there are abrupt changes and spikes in the water quality parameters that are measured by sensors at various locations within the network. An optimal attack, which results in maximum damage, is characterized by its *magnitude* (i.e., magnitude of abrupt changes in the water quality parameters) as well as its *starting time*. To model the magnitude of a contamination attack, we use a standard approach that is employed in the water networks literature to simulate anomalous events disrupting water quality data [16, 22, 14, 28].

The first step of this approach is the standardization of data to a common scale since various water quality parameters are measured in different units. The standardization of data is performed as follows: For each water quality parameter i, the data collected from the water quality sensors is normalized by subtracting the mean μ_i of the parameter value from each water quality measurement $x_i(k)$ and dividing this difference by the standard deviation σ_i of the parameter; that is, we compute $z_i(k) = \frac{x_i(k)-\mu_i}{\sigma_i}$. For our evaluation, we estimated the mean μ_i and standard deviation σ_i of each parameter i based on the six weeks of data collected under normal conditions (see Section 7.1). In the standardized data, all quality parameters have zero mean and unit standard deviation. To simulate a contamination attack at a particular time k, we multiply the normal water quality value at that time by a certain factor λ and then superimpose it on the normal data; that is, the attacked value is computed as $z_i'(k) = (\lambda + 1) \cdot z_i(k)$. For instance, if the standardized pH value is $z_{\mathrm{pH}}(k) = 0.5$ at time k, then an attack with factor $\lambda = 3$ results in attacked pH value $z'_{\mathrm{pH}}(k) = (3 + 1) \cdot 0.5 = 2$. Once we have superimposed the anomalous event on the normal standardized data, we de-normalize the data

[4] Studies on the response of water quality sensors to chemical and biological loads have shown that free chlorine, total organic carbon (TOC), electrical conductivity, and chloride are among the most reactive parameters to water contaminants [14].

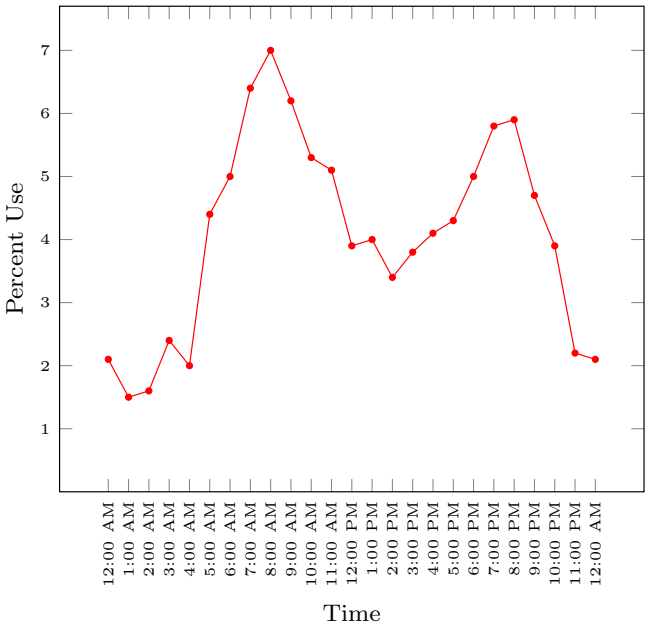

Fig. 2: Hourly water demand during a day [21].

by transforming it to the original units. We de-normalize the data by multiplying $z'_i(k)$ with standard deviation σ_i and adding mean μ_i to the result. We refer readers to [22] for a detailed account of this approach for simulating contamination events. Finally, in our evaluation, we consider six attack magnitudes $\lambda \in \{1.5, 2, 2.5, 3, 4, 5\}$, and we consider best-response attacks that select magnitude and starting time to maximize damage. Note that for given thresholds, finding a best-response attack is computationally easy since an exhaustive search needs to consider only $T \cdot \Lambda$ possible attacks.

Damage Function Figure 2 presents a typical water demand during a day [21]. Since demand is time-dependent, expected damage caused by contamination attacks, e.g., exposed population and volume of contaminated water, is also time-dependent. That is, expected disruptions at a high-demand time would cause higher damage than disruptions at a low-demand time. To model the damage function, we consider the finite horizon to be a single day divided into 10 min intervals (i.e., $T = \{1, ..., 144\}$). Then, for each timestep $k \in T$, we define the expected damage as $\mathcal{D}(k, \lambda) = (\lambda - 1) \cdot d(k)$, where $d(k) \in [0, 1]$ is the demand ratio at time k and $\lambda - 1$ is the added attack magnitude.

Table 2: Model Assessment on Test Data

	Chl.	EC	pH	Temp.	TOC	Turb.
R^2	0.939	0.980	0.967	0.344	0.920	0.538
MSE	0.003	14.639	0.001	10.3	0.002	0.000

7.2 Detector Model

The detector comprises two parts: 1) An estimator, which estimates a relation between the water quality parameters during normal operation, and 2) a detection algorithm, which identifies whether an attack has occurred in the system.

Estimator We construct an estimator using an artificial neural network (ANN) for each water quality parameter [28]. For each parameter, the inputs to its corresponding ANN are the parameter's lagged measurements and current measurements of all the other quality parameters. Formally, we have $\hat{z}_i(k) = f(z_i(k-1), \boldsymbol{z}_{-i}(k))$, where $\hat{z}_i(k)$ and $z_i(k)$ are, respectively, the estimated and measured values of water parameter i at timestep k, and f is a function attained by the artificial neural network. The estimated values are used to calculate the residuals, which are defined as the difference between the measured and estimated values, denoted by $r_i(k) = z_i(k) - \hat{z}_i(k)$, where $r_i(k)$ is the residual signal for parameter i at timestep k.

Six neural networks, one for each water quality parameter, are trained. A feed-forward back-propagation network with twenty neurons in the hidden layer is used, and the network is trained using scikit-learn 0.18.1 library with tan-sigmoid transfer function in the hidden layer and linear transfer function in the output layer [27]. Table 2 shows the estimator's performance using mean squared error (MSE) and coefficient of determination (R^2) as performance criteria.

Detection Algorithm We use the CUSUM method as the detection algorithm. CUSUM is a sequential algorithm frequently used for change detection [23, 33]. The CUSUM statistic $S(k)$ is described by $S(k) = (S(k-1) + r(k) - b)^+$, where $S(0) = 0$, $(a)^+ = a$ if $a \geq 0$ and zero otherwise, $r(k)$ is a residual difference between expected and measured sensor values generated by an estimator such that under normal behavior it has expected value of zero, and $b \in \mathbb{R}_+$ is a small constant. Assigning η_k as the detection threshold selected based on a desired false-alarm probability, the decision rule is defined as

$$d(S(k)) = \begin{cases} \text{Attack} & \text{if } S(k) > \eta_k \\ \text{Normal} & \text{otherwise.} \end{cases}$$

As discussed in Section 3, for each attack type (characterized by attack magnitude in this case), there exists a trade-off between the false-positive probability and detection delay, which depends on the detection threshold. To obtain the trade-off curve for an attack magnitude, we simulate attacks for various threshold values with randomly chosen start times, and then measure the detection

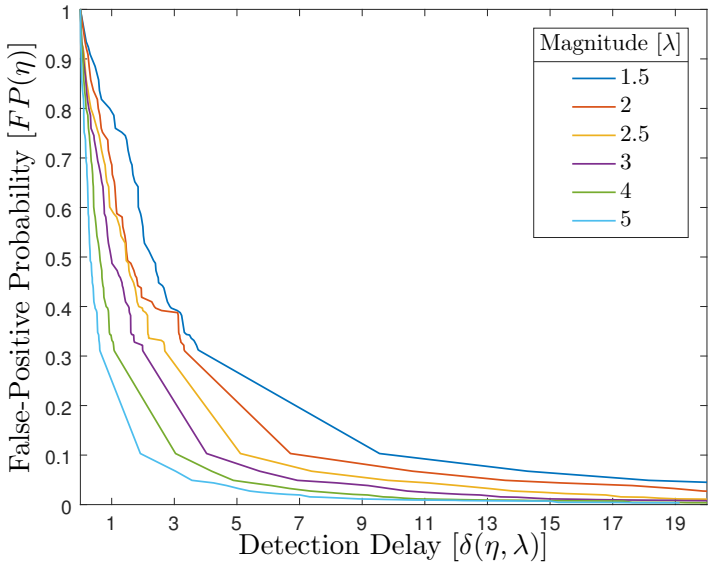

Fig. 3: Trade-off between detection delay and false-positive probability (total chlorine).

delay values. For each threshold value, we perform 1,000 simulations and compute the average detection delay. Next, using the same threshold, we simulate the system under normal operation and measure the false-positive probability. By varying the threshold and repeating these steps for all attack magnitudes, we derive the attainable detection delays and false alarm probabilities.

We consider six attack magnitudes $\lambda \in \{1.5, 2, 2.5, 3, 4, 5\}$. We select $b = 0.01$ for the CUSUM detector in order to allow small displacements to be detected quickly. Our results for a water quality parameter (total chlorine) are demonstrated in the trade-off curve shown in Figure 3, which defines the false-positive probability that can be obtained as a function of the corresponding detection delay. The results confirm that the detection delay is proportional to the threshold, and the false positive rate is inversely proportional to the threshold. Further, it can be observed that as the absolute value of attack magnitude increases, the detection delay decreases.

7.3 Optimal Thresholds

The objective is to select the strategy that minimizes the defender's loss while assuming that the attacker responds using a best-response attack, which is characterized by its magnitude and start time. We let $C_f = 10$ and $C_d = 1$, and use Algorithm 2 to compute the optimal time-dependent threshold. Figure 4 shows the obtained thresholds for each timestep. The resulting opti-

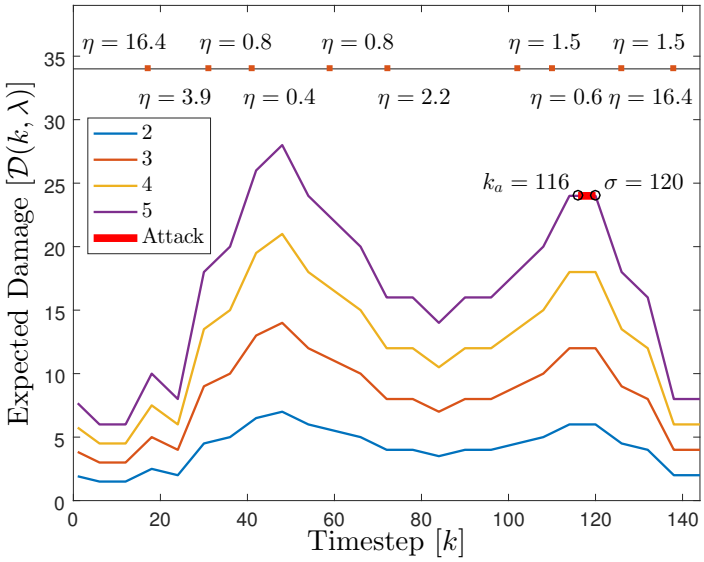

Fig. 4: Best-response attack against the optimal time-dependent threshold has the magnitude $\lambda = 5$ and starts at $k_a = 116$.

mal loss is $L^* = 187.72$. Figure 4 shows the corresponding best-response attack. The best-response attack has the magnitude $\lambda = 5$ and starts at $k_a = 116$. The attack is detected 4 timesteps later and attains the payoff $P^* = \sum_{k=116}^{120} \mathcal{D}(k, \lambda) = 120.00$. The figure also demonstrates that the detection threshold decreases as the system experiences high-demand, so that the attacks can be detected early enough. On the other hand, as the system experiences low-demand, the threshold increases to have fewer false alarms.

We also compute the optimal fixed threshold in order to compare with the time-dependent thresholds. In this case, we obtain the optimal fixed threshold $\eta^* = 0.90$ and the optimal loss $L^* = 222.45$. Figure 5 shows the best-response attack corresponding to this threshold. The best-response attack has the magnitude $\lambda = 4$ and starts at $k_a^* = 44$. The attack is detected 6 timesteps later and attains the payoff $P^* = \sum_{k=44}^{44+6} \mathcal{D}(k, \lambda) = 144.00$. Note that if the attacker starts the attack at any other timestep, the damage caused before detection is less than P^*. We observe that the optimal loss obtained by the time-dependent threshold is significantly smaller than the loss obtained by the fixed threshold.

Simulation Results We test the optimal thresholds by performing simulations that imitate realistic operation. Using our dataset, we run 42 simulations, with each of them representing a single day. We consider scenarios where the defender selects the optimal thresholds for the detector, and then the adversary attacks the system using a best-response attack. In each simulation, we record

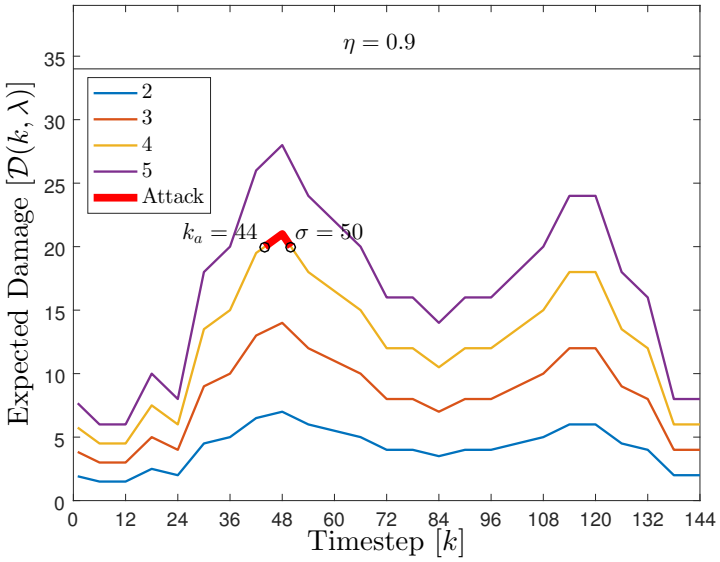

Fig. 5: Best-response attack against the optimal fixed threshold has the magnitude $\lambda = 4$ and starts at $k_a = 44$.

Table 3: Simulation Results

	Loss	Payoff	Delay	Number of FPs
Mean	195.83	110.29	3.71	5.60
STD	4.66	8.87	0.31	0.25
MSE	87.04	127.99	0.12	0.43

the payoff attained by the attacker and the loss incurred by the defender. Table 3 summarizes the simulation results. The results show that the defender's actual loss is very close to the optimal loss computed by the algorithm. In particular, the relative error between the optimal loss and the mean loss is 4.26% for the time-dependent threshold and 2.45% for the fixed threshold.

Sensitivity Analysis Figure 6a shows the optimal loss as a function of cost of threshold change C_d, when keeping cost of false positive fixed at $C_f = 10$. For small values of C_d, the optimal losses obtained by the time-dependent threshold strategy are significantly lower than the loss obtained by the fixed threshold strategy. As the cost of threshold change C_d increases, the solutions of time-dependent and fixed threshold problems become more similar. The time-dependent threshold solution converges to a fixed threshold when $C_d \geq 13.50$. Figure 6b shows the optimal loss as a function of cost of false positives for fixed and time-dependent threshold strategies when the cost of threshold change is fixed at $C_d = 1$. It can be seen that in both cases, the optimal loss

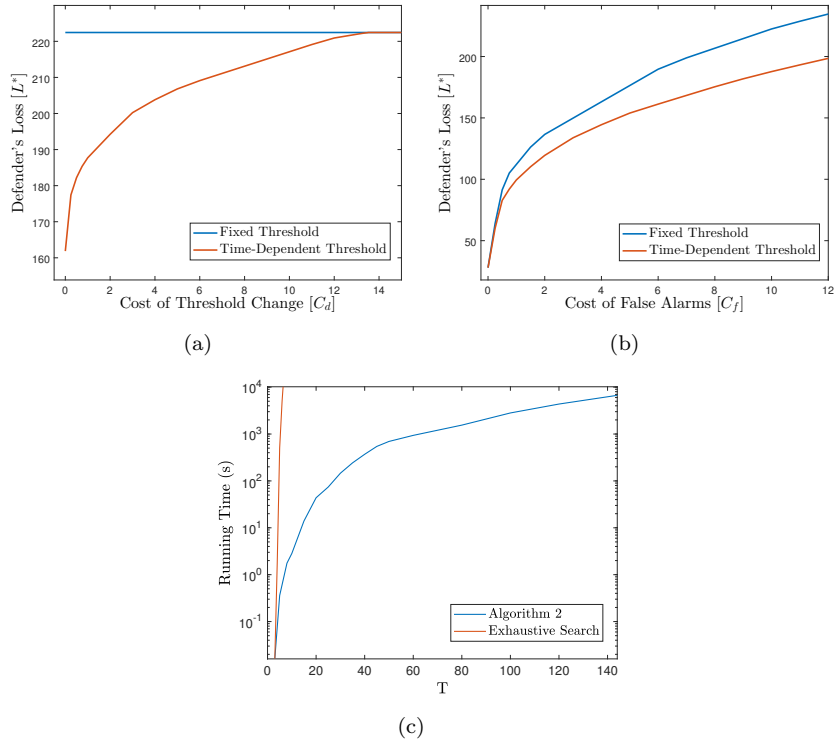

Fig. 6: (a) The defender's loss as a function of cost of threshold change. (b) The defender's loss as a function of cost of false alarms. (c) Running time of Algorithm 2 compared to exhaustive search.

increases as the cost of false alarms increases. However, in the case of time-dependent threshold, the change in loss is relatively smaller than the fixed threshold.

Running Time We now compare the running time of Algorithm 2 with an algorithm that finds the optimal thresholds using an exhaustive search. Figure 6c plots the running times as a function of T (i.e., time horizon). It can be seen that the exhaustive search algorithm has an exponential running time with respect to T, and its running time becomes significantly high even for small values of T. This is expected as the exhaustive search algorithm has the running time $\mathcal{O}(\Delta^{T+|\Lambda|})$. In contrast, Algorithm 2 performs considerably better, and the running time is reasonable for all values of T.

7.4 Random Faults

Figure 7 shows a comparison between thresholds chosen based on only attacks and combination of either faults or attacks. For each set of thresholds, we compute two different losses, loss due to only attacks (i.e., Equation (2)) and loss due to combination of either faults or attacks (i.e., Equation (11)). In the figure, we denote the thresholds obtained by considering faults and attacks as η_C^* and the thresholds obtained by considering only attacks as η_A^*. We also let $\mathcal{L}_C^*(\eta)$ be the combination, i.e., (11), when thresholds η are selected. Similarly, we let $\mathcal{L}_A^*(\eta)$ be the loss considering only attacks, i.e., Equation(2), when thresholds η are selected.

We observe that $\mathcal{L}_C^*(\eta_C^*)$ outperforms $\mathcal{L}_C^*(\eta_A^*)$ and $\mathcal{L}_A(\eta_A^*)$ outperforms $\mathcal{L}_A(\eta_C^*)$. This was clearly expected as η_C^* are the optimal thresholds with respect to $\mathcal{L}_C^*$ and η_A^* are the optimal thresholds with respect to $\mathcal{L}_A^*$. However, we notice that the difference between $\mathcal{L}_C^*(\eta_A^*)$ and $\mathcal{L}_C^*(\eta_C^*)$ is extremely small, whereas the difference between $\mathcal{L}_A^*(\eta_A^*)$ and $\mathcal{L}_A^*(\eta_C^*)$ is very large. In other words, the thresholds η_C^* perform well only when combination of faults and attacks is considered and perform very poorly when only attacks is considered, whereas the thresholds η_A^* perform very well in both cases. This highlights the difference between minimizing losses due to attacks and due to random faults. In the former case, the defender must focus on the timesteps in which the system is most vulnerable (i.e., highest expected damage), and it can be less vigilant at other times since the attacker will not choose to attack when the system is less vulnerable. Therefore, thresholds η_A^* may be "negligent" at times when the expected damage $\mathcal{D}$ is low; however, even if a random fault occurs at such time, it will cause minor damage. When minimizing losses due to random faults, the defender must be vigilant at all times, which means that it cannot afford to focus on the timesteps in which the system is most vulnerable. Therefore, if thresholds minimize losses due to random faults, an attack can cause catastrophic losses by targeting one of these vulnerable timesteps. Since thresholds η_C^* consider both attacks and random faults, they do not perform catastrophically against attacks, but they do perform poorly in comparison with the other cases.

8 Conclusion

In this paper, we studied the problem of finding optimal detection thresholds for anomaly-based detectors implemented in dynamical systems in the face of strategic attacks. We formulated the problem as an attacker-defender security game that determined thresholds for the detector to achieve an optimal trade-off between the detection delay and the false-positive probabilities. To this end, we presented a dynamic-programming based algorithm that computes optimal time-dependent thresholds. We analyzed the performance of the time-dependent threshold strategy, showing that the running time of our algorithm is polynomial in the time dimension. As a special case, we also

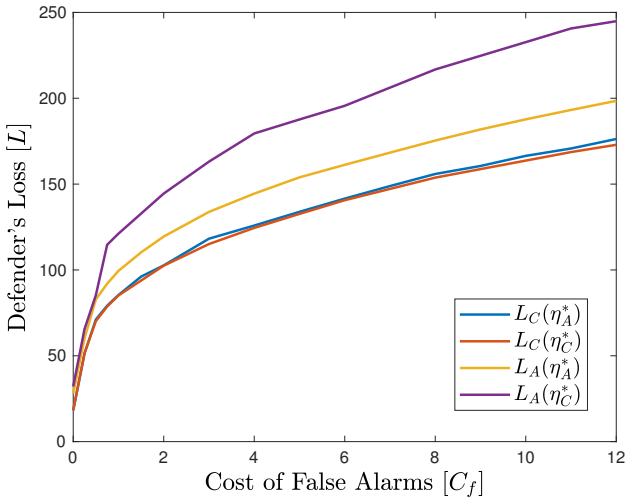

Fig. 7: The defender's loss as a function of cost of false alarms for time-dependent thresholds. η_A^* is the optimal threshold for attacks and η_C^* is the optimal threshold for combination of faults and attacks.

studied and provided a polynomial-time algorithm for the problem of computing optimal fixed thresholds, which do not change with time. In addition, we studied the problem of finding optimal thresholds in the presence of random faults and attacks, and presented an efficient algorithm that computes the optimal thresholds. Finally, we evaluated our results using a case study of detecting contamination attacks in a water distribution system. We showed that the optimal time-dependent thresholds found using our algorithm significantly outperform fixed thresholds.

References

1. Alippi C, Roveri M (2006) An adaptive CUSUM-based test for signal change detection. In: Proceedings of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS), pp 5752–5755
2. Alpcan T, Basar T (2003) A game theoretic approach to decision and analysis in network intrusion detection. In: Proceedings of the 42nd IEEE Conference on Decision and Control (CDC), IEEE, vol 3, pp 2595–2600
3. Alpcan T, Başar T (2004) A game theoretic analysis of intrusion detection in access control systems. In: Proceedings of the 43rd IEEE Conference on Decision and Control (CDC), IEEE, vol 2, pp 1568–1573
4. Arad J, et al (2013) A dynamic thresholds scheme for contaminant event detection in water distribution systems. Water Research

5. Basseville M, Nikiforov IV (1993) Detection of abrupt changes: Theory and application, vol 104. Prentice Hall, Englewood Cliffs
6. CANARY (2010) Canary: a water quality event detection tool. `http://waterdata.usgs.gov/nwis/`, [Online; accessed October 20, 2016]
7. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: A survey. ACM Computing Surveys 41(3):15
8. Deng Y, Jiang W, Sadiq R (2011) Modeling contaminant intrusion in water distribution networks: A new similarity-based dst method. Expert Systems with Applications 38(1):571–578
9. Di Nardo A, et al (2013) Water network protection from intentional contamination by sectorization. Water Resources Management 27(6):1837–1850
10. Estiri M, Khademzadeh A (2010) A theoretical signaling game model for intrusion detection in wireless sensor networks. In: Proceedings of the 14th International Telecommunications Network Strategy and Planning Symposium (NETWORKS), IEEE, pp 1–6
11. Ghafouri A, Abbas W, Laszka A, Vorobeychik Y, Koutsoukos X (2016) Optimal thresholds for anomaly-based intrusion detection in dynamical environments. In: Proceedings of the 7th Conference on Decision and Game Theory for Security (GameSec), pp 415–434
12. Gibbons RD (1999) Use of combined Shewhart-CUSUM control charts for ground water monitoring applications. Ground Water 37(5):682–691
13. Gleick PH (2006) Water and terrorism. Water Policy 8(6):481–503
14. Hall J, et al (2007) On-line water quality parameters as indicators of distribution system contamination. Journal – American Water Works Association 99(1):66–77
15. Hart D, et al (2007) CANARY: A water quality event detection algorithm development tool. In: Proceedings of the World Environmental and Water Resources Congress
16. Klise KA, McKenna SA (2006) Water quality change detection: multivariate algorithms. In: Procedings of the International Society for Optical Engineering, Defense and Security Symposium, International Society for Optics and Photonics
17. Korzhyk D, Yin Z, Kiekintveld C, Conitzer V, Tambe M (2011) Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. Journal of Artificial Intelligence Research 41:297–327
18. Laszka A, Johnson B, Grossklags J (2013) Mitigating covert compromises: A game-theoretic model of targeted and non-targeted covert attacks. In: Proceedings of the 9th Conference on Web and Internet Economics (WINE), pp 319–332
19. Luo Y, Li Z, Wang Z (2009) Adaptive CUSUM control chart with variable sampling intervals. Computational Statistics & Data Analysis
20. Mac Nally R, Hart B (1997) Use of CUSUM methods for water-quality monitoring in storages. Environmental Science & Technology 31(7):2114–2119

21. Mayer PW, et al (1999) Residential end uses of water
22. McKenna SA, Wilson M, Klise KA (2008) Detecting changes in water quality data. Journal – American Water Works Association 100(1):74
23. Page E (1954) Continuous inspection schemes. Biometrika 41(1/2):100–115
24. Paruchuri P, Pearce JP, Marecki J, Tambe M, Ordonez F, Kraus S (2008) Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS), International Foundation for Autonomous Agents and Multiagent Systems, pp 895–902
25. Patcha A, Park JM (2004) A game theoretic approach to modeling intrusion detection in mobile ad hoc networks. In: Proceedings of the 5th Annual IEEE SMC Information Assurance Workshop, IEEE, pp 280–284
26. Pawlick J, Farhang S, Zhu Q (2015) Flip the cloud: Cyber-physical signaling games in the presence of advanced persistent threats. In: Proceedings of the 6th International Conference on Decision and Game Theory for Security (GameSec), Springer, pp 289–308
27. Pedregosa F, et al (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830
28. Perelman L, et al (2012) Event detection in water distribution systems from multivariate water quality time series. Environmental Science & Technology 46
29. Shen S, Li Y, Xu H, Cao Q (2011) Signaling game based strategy of intrusion detection in wireless sensor networks. Computers & Mathematics with Applications 62(6):2404–2416
30. Tambe M (ed) (2011) Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned. Cambridge University Press
31. Urbina DI, et al (2016) Limiting the impact of stealthy attacks on industrial control systems. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, pp 1092–1105
32. Van Dijk M, et al (2013) Flipit: The game of stealthy takeover. Journal of Cryptology 26(4):655–713
33. Verdier G, et al (2008) Adaptive threshold computation for CUSUM-type procedures in change detection and isolation problems. Computational Statistics & Data Analysis 52(9):4161–4174

A Supplementary Material

A.1 Proof of Theorem 1

Proof Given an instance of the Stackelberg game, let $\boldsymbol{\eta}$ be optimal thresholds that do not necessarily satisfy the constraint of the lemma. Then, construct thresholds $\boldsymbol{\eta}^*$ that satisfy the constraint by replacing each η_k with $\eta_k^* = \max_{\eta\,:\,\delta(\eta,\lambda) \leq \delta(\eta_k,\lambda)} \eta$. For any attack (k_a, λ), the detection delay and hence the expected damage are the same for $\boldsymbol{\eta}$ and $\boldsymbol{\eta}^*$. Consequently, the damage caused by best-response attacks must also be the same for $\boldsymbol{\eta}$ and $\boldsymbol{\eta}^*$. Further, the defender's costs for $\boldsymbol{\eta}$ are greater than or equal to those for $\boldsymbol{\eta}^*$ since 1) for every k,

$\eta_k \leq \eta^*_k$ and FP is decreasing, and 2) the number of threshold changes in $\boldsymbol{\eta}$ is greater than or equal to that in $\boldsymbol{\eta}^*$. Therefore, $\boldsymbol{\eta}^*$ is optimal, which concludes our proof.

A.2 Proof of Lemma 1

Proof We assume that we are given a damage bound P, and we have to find thresholds that minimize the total cost of false positives and threshold changes, subject to the constraint that any attack against these thresholds will result in at most P damage. In order to solve this problem, we use a dynamic-programming algorithm. We will first discuss the algorithm without a cost for changing thresholds, and then show how to extend it to consider costly threshold changes.

We let $\Delta^{|\Lambda|}$ denote the Cartesian power $\underbrace{\Delta \times \Delta \times \ldots \times \Delta}_{|\Lambda|}$ of the set Δ. For any two variables $n \in \{1, \ldots, T\}$ and $\boldsymbol{m} \in \Delta^{|\Lambda|}$ such that $\forall \lambda \in \Lambda : 0 \leq m_\lambda < n$, we define $\textsc{Cost}(n, \boldsymbol{m})$ to be the minimum cost of false positives from n to T subject to the damage bound P, given that attacks of type λ can start at $k_a \in \{n - m_\lambda, \ldots, T\}$ and they are not detected prior to n. Formally, we can define $\textsc{Cost}(n, \boldsymbol{m})$ as

$$\min_{(\eta_n, \ldots, \eta_T)} \sum_{k=n}^{T} C_f \cdot FP(\eta_k) \tag{16}$$

subject to

$$\forall \lambda \in \Lambda, k_a \in \{n - m_\lambda, \ldots, T\} : \min_{i \,:\, i \geq n \,\wedge\, \delta(\eta_i, \lambda) \leq i - k_a} \sum_{k=k_a}^{i} \mathcal{D}(k, \lambda) \leq P. \tag{17}$$

If there are no thresholds that satisfy the damage bound P under these conditions, we let $\textsc{Cost}(n, \boldsymbol{m})$ be ∞.[5]

We can recursively compute $\textsc{Cost}(n, \boldsymbol{m})$ as follows. Firstly, for any n and $\boldsymbol{m}$, if there exists an attack type λ such that $\sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P$, then an attack of type λ starting at time $n - m_\lambda$ will cause greater than P damage, regardless of the thresholds $\eta_n, \ldots, \eta_T$. Consequently, in this case, we can immediately set $\textsc{Cost}(n, \boldsymbol{m})$ to ∞.

Otherwise, we iterate over all possible threshold values $\eta \in E$, and choose the one that minimizes the cost $\textsc{Cost}(n, \boldsymbol{m})$. For any threshold η, we can compute the resulting cost as follows. If $\delta(\eta, \lambda) > m_\lambda$, then no attack of type λ would be detected at time n, so we would have to increase m_λ for the next timestep $n + 1$. On the other hand, if $\delta(\eta, \lambda) \leq m_\lambda$, then attacks starting at time $n - \delta(\eta, \lambda)$ or earlier would be detected at time n, so we would have to decrease m_λ to $\delta(\eta, \lambda)$ for the next timestep $n + 1$. Hence, if we selected threshold η for timestep n, then we would have to update $\boldsymbol{m}$ to $\langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}$ for the next timestep. Therefore, if we selected threshold η for timestep n, then the attained cost would be the sum of the cost $C_f \cdot FP(\eta)$ for timestep n and the best possible cost $\textsc{Cost}(n + 1, \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda})$ for the remaining timesteps. By combining this formula with the rule for assigning infinite cost, we can compute $\textsc{Cost}(n, \boldsymbol{m})$ as

$$\textsc{Cost}(n, \boldsymbol{m}) = \begin{cases} \infty & \text{if } \bigvee_{\lambda \in \Lambda} \sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P, \\ \min_\eta \textsc{Cost}(n + 1, \langle \min\{\delta(\eta, \lambda), & \\ \quad m_\lambda + 1\} \rangle_{\lambda \in \Lambda}) + C_f \cdot FP(\eta) & \text{otherwise.} \end{cases} \tag{18}$$

Note that in the equation above, $\textsc{Cost}(n, \boldsymbol{m})$ does not depend on $\eta_1, \ldots, \eta_{n-1}$, it depends only on the feasible thresholds for the subsequent timesteps. Therefore, starting from

[5] Note that in practice, ∞ can be represented by a sufficiently high natural number.

the last timestep T and iterating backwards, we are able to compute $\textsc{Cost}(n, \boldsymbol{m})$ for all timesteps n and all values $\boldsymbol{m}$. Finally, for $n = T$ and any $\boldsymbol{m}$, computing $\textsc{Cost}(T, \boldsymbol{m})$ is straightforward: if the damage from $\boldsymbol{m}$ does not exceed the threshold P for any attack type λ, then $\textsc{Cost}(T, \boldsymbol{m}) = \min_{\eta \in E} C_f \cdot FP(\eta)$; otherwise, $\textsc{Cost}(T, \boldsymbol{m}) = \infty$.

Having found $\textsc{Cost}(n, \boldsymbol{m})$ for all n and $\boldsymbol{m}$, by definition, $\textsc{Cost}(1, \langle 0, \dots, 0 \rangle)$ is the minimum cost of false positives subject to the damage bound P. The minimizing threshold values can be recovered by iterating forward from $n = 1$ to T and again using Equation (18). That is, for every n, we select the threshold value η_n^* that attains the minimum cost $\textsc{Cost}(n, \boldsymbol{m})$, where $\boldsymbol{m}$ can easily be computed from the preceding threshold values $\eta_1^*, \dots, \eta_{n-1}^*$.[6]

Costly Threshold Changes. Now, we show how to extend the computation of $\textsc{Cost}$ to consider the cost C_d of changing the threshold. Let $\textsc{Cost}(n, \boldsymbol{m}, \eta_{\text{prev}})$ be the minimum cost for timesteps starting from n subject to the same constraints as before but also given that the threshold value in timestep $n-1$ (i.e., the previous timestep) is η_{prev}. Then, $\textsc{Cost}(n, \boldsymbol{m}, \eta_{\text{prev}})$ can be computed similarly to $\textsc{Cost}(n, \boldsymbol{m})$: for any $n < T$, iterate over all possible threshold values η, and choose the one that results in the lowest cost $\textsc{Cost}(n, \boldsymbol{m}, \eta_{\text{prev}})$. If $\eta_{\text{prev}} = \eta$ or if $n = 1$, then the cost is computed the same way as in the previous case (i.e., similar to Equation (18)). Otherwise, the cost also has to include the cost C_d of changing the threshold. Consequently, we first define

$$S(n, \boldsymbol{m}, \eta_{\text{prev}}, \eta) = \begin{cases} \textsc{Cost}(n+1, \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}) + C_f \cdot FP(\eta) & \text{if } \eta \in \{\eta_{\text{prev}}, 1\}, \\ \textsc{Cost}(n+1, \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}) + C_f \cdot FP(\eta) + C_d & \text{otherwise.} \end{cases} \tag{19}$$

Then, similar to Equation (18), we can express the optimal cost as

$$\textsc{Cost}(n, \boldsymbol{m}, \eta_{\text{prev}}) = \begin{cases} \infty & \text{if } \bigvee_{\lambda \in \Lambda} \sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P, \\ \min_\eta S(n, \boldsymbol{m}, \eta_{\text{prev}}, \eta) & \text{otherwise.} \end{cases}$$

Note that for $n = 1$, we do not add the cost C_d of changing the threshold. Similarly to the previous case, $\textsc{Cost}(1, 0, \text{arbitrary})$ is the minimum cost subject to the damage bound P, and the minimizing thresholds can be recovered by iterating forward.

A.3 Proof of Theorem 2

Proof For any damage bound P, using the algorithm MinimumCostThresholds (Algorithm 1), we can find thresholds that minimize the total cost of false positives and threshold changes, which we will denote by $TC(P)$, subject to the constraint that an attack can cause at most P damage. Since the defender's loss is the sum of its total cost and the damage resulting from a best-response attack, we can find optimal thresholds by solving

$$\min_P \; TC(P) + P \tag{20}$$

and computing the optimal thresholds $\boldsymbol{\eta}^*$ for the minimizing P^* using our dynamic-programming algorithm.

To show that this formulation does indeed solve the problem of finding optimal thresholds, we use indirect proof. For the sake of contradiction, suppose that there exist thresholds $\boldsymbol{\eta}'$ for which the defender's loss L' is lower than the loss L^* for the solution $\boldsymbol{\eta}^*$ of the above formulation. Let P' be the damage resulting from the attacker's best-response

[6] Note that in Algorithm 1, we store the minimizing values $\eta^*(n, \boldsymbol{m})$ for every n and $\boldsymbol{m}$ when iterating backwards, thereby decreasing running time and simplifying the presentation of our algorithm.

against $\boldsymbol{\eta}'$, and let TC' be the defender's total cost for $\boldsymbol{\eta}'$. Since the best-response attack against $\boldsymbol{\eta}'$ achieves at most P' damage, we have from the definition of $TC(P)$ that $TC' \geq TC(P')$. It also follows from the definition of $TC(P)$ that $L^* \leq TC(P^*) + P^*$. Combining the above with our supposition $L^* > L'$, we get

$$TC(P^*) + P^* \geq L^* > L' = TC' + P' \geq TC(P') + P'.$$

However, this is a contradiction since P^* minimizes $TC(P) + P$ by definition. Therefore, thresholds $\boldsymbol{\eta}^*$ must be optimal.

It remains to show that Algorithm 2 finds an optimal damage bound P^*. To this end, we show that P^* can be found using an exhaustive search over a set, whose cardinality is polynomial in the size of the problem instance. Consider the set of damage values resulting from all possible attack scenarios $k_a \in T$, $\delta \in \Delta$, $\lambda \in \Lambda$, that is, the set

$$\left\{ \sum_{k=k_a}^{k_a+\delta} \mathcal{D}(\lambda, k) \;\middle|\; \exists\, k_a \in \{1, \ldots, T\},\, \delta \in \Delta,\, \lambda \in \Lambda \right\}. \tag{21}$$

Let the elements of this set be denoted by $P_1, P_2, \ldots$ in increasing order. It is easy to see that for any i, the set of thresholds that satisfy the damage constraint is the same for every damage value $P \in [P_i, P_{i+1})$. Hence, for any i, the cost $TC(P)$ is the same for every $P \in [P_i, P_{i+1})$. Therefore, the optimal P^* must be a damage value P_i from the above set, which we can find by simply iterating over the set.

A.4 Proof of Proposition 1

Proof In the dynamic-programming algorithm (Algorithm 1), we first compute $\textsc{Cost}(n, \boldsymbol{m}, \delta_{n-1})$ for every $n \in \{1, \ldots, T\}$, $\boldsymbol{m} \in \Delta^{|\Lambda|}$, and $\eta_{\text{prev}} \in E$, and each computation takes $\mathcal{O}(|E| \cdot |\Lambda|)$ time. Then, we recover the optimal detection delay for all timesteps $\{1, \ldots, T\}$, and the computation for each timestep takes a constant time. Consequently, the running time of the dynamic-programming algorithm is $\mathcal{O}(T \cdot |\Delta|^{|\Lambda|+1} \cdot |\Lambda| \cdot |E|)$.

In the exhaustive search, we first enumerate all possible damage values by iterating over all possible attacks (k_a, δ, λ), where $k_a \in \{1, \ldots, T\}$, $\delta \in \Delta$, and $\lambda \in \Lambda$. Then, for each possible damage value, we execute the dynamic-programming algorithm, which takes $\mathcal{O}(T \cdot |\Delta|^{|\Lambda|+1} \cdot |\Lambda| \cdot |E|)$ time. Consequently, the running time of Algorithm 2 is $\mathcal{O}(T^2 \cdot |\Delta|^{|\Lambda|+2} \cdot |\Lambda|^2 \cdot |E|)$.

A.5 Algorithm 3 and Proof of Proposition 2

Proof The obtained threshold is optimal since the algorithm evaluates all possible solutions through exhaustive search. Given a tuple (η, k_a, λ), when computing the attacker's payoff $\mathcal{P}(\eta, k_a, \lambda)$, we use the payoff computed in previous iteration, which takes constant time. We repeat these steps for each attack type $\lambda \in \Lambda$. Therefore, the running time of the algorithm is $\mathcal{O}(T \cdot |E| \cdot |\Lambda|)$.

Algorithm 3 Optimal Fixed Threshold

Input: $\mathcal{D}(k,\lambda)$, T, C_f
Initialize: $L^* \leftarrow \infty$
for all $\eta \in E$ **do**
  $P' \leftarrow 0$
  for all $\lambda \in \Lambda$ **do**
    for all $k_a \in \{1,\ldots,T\}$ **do**
      $P(\eta,k_a,\lambda) \leftarrow \sum_{k_a}^{k_a+\delta(\eta,\lambda)} D(k,\lambda)$
      if $\mathcal{P}(\eta,k_a,\lambda) > P'$ **then**
        $P' \leftarrow \mathcal{P}(\eta,k_a,\lambda)$
        $L' \leftarrow P' + C_f \cdot FP(\eta) \cdot T$
  if $L' < L^*$ **then**
    $L^* \leftarrow L'$
    $\eta^* \leftarrow \eta$