Resilient Anomaly Detection in Cyber-Physical Systems

By

Amin Ghafouri

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

February 28, 2018

Nashville, Tennessee

Approved:

Xenofon Koutsoukos, Ph.D.

Yevgeniy Vorobeychik, Ph.D.

Gautam Biswas, Ph.D.

Abhishek Dubey, Ph.D.

Gabor Karsai, Ph.D.

To my parents.

Acknowledgments

I would like to thank my Ph.D. advisor, Dr. Xenofon Koutsoukos for his help and guidance. I would also like to thank Dr. Yevgeniy Voroboyechik, Dr. Gautam Biswas, Dr. Abhishek Dubey, and Dr. Gabor Karsai for being members of my thesis committee.

In addition, I would like to thank other colleagues at Vanderbilt Univeristy who assisted me during my studies. In particular, I would like to thank Aron Laszka, Waseem Abbas, Zhenkai Zhang, Siyuan Dai, Hamzah Abdel-aziz, Bradley Potteiger, Saqib Hasan, Hamid Zare, Hamed Khorasgani, and others at the Institute for Software Integrated Systems.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The complexity of cyber-physical systems (CPS), the roles and responsibilities of the humans that interact with them, and the cyber-security of these highly interconnected systems have led to a new research paradigm known as resilient CPS. By considering the elements and disciplines that contribute to a more effective design, resilient CPS provide interdisciplinary solutions for problems such as how to tailor the control system to enable it to respond to disturbances quickly and efficiently, how to better integrate widely distributed CPS to prevent faults that result in disruptions to operations of critical infrastructure, and how to design cyber-security protection mechanisms so that the system defends itself from cyber-attacks by changing its behaviors.

A resilient CPS is one that should maintain state awareness and an accepted level of operational normalcy in response to any disturbances, including threats of unexpected and malicious nature. In particular, a resilient CPS maintains its operational goals in the presence of:

- System Faults: Advanced control algorithms deployed in CPS are dependent upon data from multiple sensors to predict the behaviors of the system and make corrective responses. However, such systems can become brittle to the extent that any unrecognized fault or degradation in the sensors can lead to incorrect responses by the control algorithm and potentially compromise the desired operation. Therefore, advanced control algorithms in resilient CPS require the implementation of detection and diagnosis architectures to recognize sensor faults and degradations.

- Cyber-Attacks: Computer security and sensor network security have focused on prevention mechanisms but do not address how a CPS can continue to function under attacks. Control theory, on the other hand, has strong results on robust and fault-tolerant algorithms against well-defined uncertainties or faults, but there is little work accounting for attacks caused by malicious adversaries.

1.2 Challenges

A resilient CPS must be able to detect failures and cyber-attacks quickly and accurately in order to minimize their adverse effects. While a large number of anomaly detection methods are designed in other domains, they are often built with Information Technology (IT) systems in mind. Thus, typical anomaly detectors do not take into account the properties of the physical components of CPS and are merely based on the cyber components. Also, anomaly detectors designed specifically for CPS are often faced with limitations such as inapplicability to general CPS domains and shortcomings such as high overhead that make them undesirable solutions. Hence, a significant challenge is the design of anomaly detectors that effectively detect failures and cyber-attacks in CPS.

To design anomaly detectors and analyze their performance, a model of the physical system is needed. The model can be used, for example, to predict the future state of the CPS or to define utility and loss functions. Models of the physical system are typically developed using either physical laws or data-driven system identification methods. While models based on physical laws provide high fidelity, they are often very hard to obtain due to the highly nonlinear and complex nature of the system. In addition, although data-driven models are relatively simple to obtain, they often do not provide the precision and accuracy of models based on the physical laws. Therefore, another challenge is the design of accurate and precise models of the physical system.

In design and evaluation of anomaly detectors, realistic attack models that represent the harmful effects of cyber-attacks on CPS are needed. However, unlike fault detection where assumptions are made about the properties of faults, making restrictive assumptions about strategic cyber-attacks is not permitted since it results in unrealistic models. Thus, there is a challenge in obtaining attack models that adequately represent the attack while not being too restrictive.

In anomaly detectors, while it is desirable to simultaneously improve detection performance and security overhead, this is often impossible due to the trade-offs between them. These trade-offs can typically be changed through selecting different configurations for anomaly detectors, where each configuration results in a different detection performance and security overhead. Finding the right configuration that optimally balances the trade-off between detection performance and security overhead is another challenge.

The final challenge is the design of resilient control algorithms that guarantee some notion of correct behavior at a minimum level of performance even when the system is under attack. For example, in traffic signal control, it is desirable that when traffic flow sensors are under attack, traffic signal controllers still maintain a minimum level of performance to avoid disastrous congestions.

1.3 Contributions

In this section, we present an overview of the contributions towards addressing the outlined challenges. In particular, the focus of our contributions is on improving resilience of CPS through design and evaluation of anomaly detection methods that incorporate domain-specific properties of the physical system. The contributions of this document are described below.

Chapter 3

We study the problem of finding optimal thresholds for anomaly-based detection in dynamical systems in the face of strategic attacks. Specifically, we have the following contributions:

- We formulate a two-player Stackelberg game between a defender and an adversary. 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 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 and compute optimal time-dependent 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.
- We provide and study a polynomial-time algorithm for the problem of computing optimal *fixed thresholds*, which do not change with time.
- 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.
- We evaluate our results by applying it to the detection of contamination attacks in a water-distribution system. 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.

Chapter 4

We study the problem of finding optimal thresholds for anomaly detection of faulty traffic sensors, considering route planning as the application of interest. The objective is to select the optimal

thresholds of anomaly detectors in order to optimize the performance of the route planning application in the presence of faulty sensors. In particular, we make the following contributions:

- We devise an effective anomaly detector for identifying faulty traffic sensors using a prediction model based on Gaussian Processes.
- We present an approach for computing the optimal parameters of the detector which minimize losses due to false-positive and false-negative errors.
- We characterize critical sensors, whose failure can have high impact on the traffic application.
- We implement our method and evaluate it numerically using a real-world dataset and the route planning platform OpenTripPlanner [93].

Chapter 5

We propose a framework for *application-aware* anomaly detection in sensors measurements in CPS. The objective is to optimally configure all anomaly detectors of a CPS such that the performance loss in the presence of detection errors is minimized. Specifically, we make the following contributions:

- We present an approach to recover from anomalies in order to maintain operation when detection alerts are triggered.
- We propose the application-aware anomaly detector, in which the anomaly detector is optimally configured such that the performance loss in the presence of detection errors is minimized. In particular, the thresholds are selected such that the performance of the system in the presence of detection errors is as close as possible to the performance that could have been obtained if there were no detection errors.
- We prove that the application-aware detection problem is, in general, NP-hard. We then present an algorithm to efficiently find near-optimal solutions.
- We study two special variations of the application-aware detection problem, that is, single detector and detectors with equal threshold. We optimally solve both special cases, which can be used in resource-constrained environments.
- We perform experiments on a case study of real-time control of traffic signals. We evaluate our approach numerically and show its benefits compared to standard anomaly detection practices.

Chapter 6

We study the *adversarial regression* problem, in which an omniscient adversary that is capable of perturbing the values of a subset of sensors attempts to drive a CPS to an unsafe state (e.g., raising the pressure of a reactor beyond its safety limit in a process control system) while remaining undetected. We make the following contributions:

- We formulate the adversarial regression problem. In the problem, a safety-critical CPS that is monitored by regression-based anomaly detectors is considered. Then, an omniscient adversary that is capable of perturbing the values of a subset of sensors attempts to drive the system to an unsafe state (e.g., raising the pressure of a reactor beyond its safety limit in a process control system) without being detected.
- We solve the adversarial regression problem considering different types of regression-based detectors. In particular, we solve the problem for detectors that use linear regression, neural network regression, and an ensemble of the two as their regression algorithms.
- We present a resilient detector that mitigates the impact of stealthy attacks without increasing the number of false alarms. The resilient detector achieves this by resilient selection of detection thresholds.
- We numerically evaluate the adversarial regression problem, and demonstrate the effectiveness of the resilient detector through applying our methods to a case study of a process control system.

1.4 Organization

The technical contributions of this thesis are in Chapters 3-6. The organization is as follows.

- Chapter 2 reviews the related work in the research of resilient CPS.
- Chapter 3 presents the problem of selecting optimal thresholds for attack detection in dynamical environments.
- Chapter 4 presents the problem of optimal detection of faulty traffic sensors used in route planning.
- Chapter 5 proposes the application-aware anomaly detection framework.
- Chapter 6 studies the problem of adversarial regression in CPS.
- Chapter 7 concludes this thesis.

Chapter 2

Related Work

The breadth of material in the resilient CPS literature is exorbitant since it spans several areas including control theory, information security, game theory, and artificial intelligence. To be germane, this chapter only reviews works that are most related to the contributions of this document. We begin with an overview of resilience in CPS in Section 2.1. Then, in Section 2.2, we discuss attack detection in CPS, where we also provide relevant background on anomaly detection. In Section 2.3, we review existing work on game-theoretical approaches for anomaly detection in CPS. We discuss different research directions on anomaly detection within the scope of machine learning in Section 2.4. Finally, in Section 2.5, we discuss how the contributions of this dissertation fit within the scope of the literature.

2.1 Resilience in Cyber-Physical Systems (CPS)

Resilience in engineering is defined as "the intrinsic ability of a system to adjust its functioning prior to, during, or following changes and disturbances, so that it can sustain required operations under both expected and unexpected conditions" [59]. Similarly, resilience in CPS is defined as protecting the operational goals (e.g., stability) as well as other non-operational goals (e.g., privacy) in the presence of both expected events (e.g., failures) and unexpected events (e.g., cyber-attacks) [26].

Resilience in CPS must attain three goals: (1) Integrity which represents the trustworthiness of data or resources, (2) Availability which is the ability to access and use information on demand as specified, and (3) Confidentiality, which is the ability to keep information secret or private from unauthorized users. To satisfy these goals, merely applying well-known IT-resilience measures to CPS is not enough. This is because unlike IT systems, where resilience essentially involves encryption and protection of data, resilience in CPS considers unwanted behavior that can potentially influence the physical system's behavior. Therefore, there exist fundamental vulnerabilities, threats and challenges, which are a result of CPS's complex and typically distributed architectures and the interconnection of IT and control systems, that need to be addressed [26], [27].

Malicious attacks against CPS can be categorized into deception attacks (or integrity attacks) and denial-of-service (DoS) attacks. Deception attacks refer to the possibility of compromising the

integrity of control packets or measurements, and they are cast by altering the behavior of sensors and actuators. DoS attacks, instead, compromise the availability of resources by, for instance, jamming the communication channel. Deception attacks result in the lack of the security goal of integrity, and DoS attacks result in the lack of the security goal of availability.

Analysis of CPS-resilience against malicious attacks and design of algorithms and architectures to survive such attacks has received increasing attention [26], [27]. Elements from information security, sensor network security, and control theory that can be used for solving the problem of CPS-resilience are discussed in [26]. While existing approaches provide many useful mechanisms for improving security of CPS, they are not sufficient when used against CPS. A detailed three-dimensional attack space is proposed that links different types of attacks to the attacker's resources and knowledge of the system is presented in [128]. This attack space gives insight to both the attacker and the defender on how an attack can be perpetuated, as well as the potential impact of the attack on CPS. To demonstrate different attack scenarios captured by the attack space, a testbed of a quadruple tank process under cyber-attacks is analyzed.

DoS attacks are studied for discrete-time linear dynamical systems in [6]. The attack model is defined by generalizing traditional uncertainty classes and safety constraints are defined as security requirements of the control system. From the defender's perspective, the objective is to design control laws that are robust against the attacker's actions, whereas from the attacker's viewpoint, the goal is to determine the optimal attack plan that degrades the performance of the system as much as possible. Following this scenario, an optimal attack plan for discrete-time linear dynamical systems is proposed. DoS attacks are also studied in the context of an adversary that jams the communication between the CPS's plant and controller for a limited time [53]. A saddle-point equilibrium is proven to exist between the attacker and controller. Then, for a specific instance of the problem, an optimal policy for jamming attacks is proposed.

Considering deception attacks, the secure estimation and control of linear deterministic systems under sensor attacks is studied in [44]. The problem is formulated as a dynamic error correction problem with sparse vectors, and it is shown that by using state feedback, the system's resilience can be increased. Furthermore, replay attacks on state estimation in wireless networks are considered in [98]. Replay attacks are cast by hijacking the sensors, recording the readings for a certain amount of time, and repeating such readings while injecting an exogenous signal into the system. Such attacks can be detected by injecting a signal unknown to the attacker into the system [98], [99].

Methods to design cyber-attacks that bypass bad data detectors have been proposed. False data injection attacks against state estimators are studied in [85]. It is shown that by using the configura-

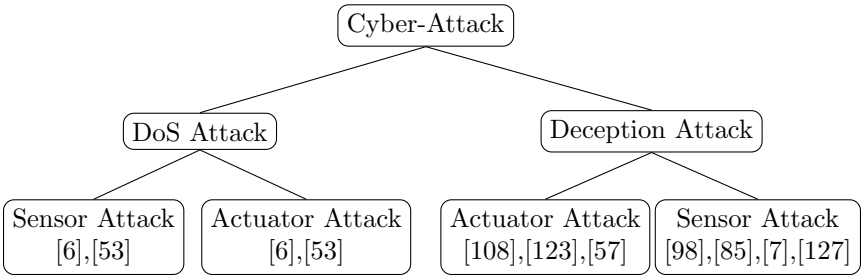


Figure 2.1: Summary of related work on CPS security.

tion of the system, an attacker can launch successful attacks that introduce errors into certain state variables while bypassing existing bad data detectors. Moreover, scenarios demonstrating stealthy deception attack policies are characterized in [123]. It is assumed that the attacker has complete model knowledge and full access to all sensor and actuator channels and is able to perform deception attacks. The stealthy attack policies are illustrated using a water irrigation example.

In addition, stealthy attack policies for networked CPS are characterized in [108]. In particular, it is shown that an attack is stealthy if the measurements due to the attack coincide with the measurements due to some nominal operating condition. Experimental stealthy deception attacks on water irrigation canals controlled by SCADA systems are presented in [7]. A set of stealthy deception attacks for attackers compromising a subset of sensors and actuators is described in [107]. Stealthy attacks with limited resources are considered and improved detection methods are proposed in [67], [68]. A security metric for studying sets of vulnerable sensors is proposed in [119]. The consequences of stealthy attacks are analyzed in [140], [129]. In particular, the work in [127] analyzes attack policies with limited model knowledge and performs experiments showing that such attacks are stealthy and can induce the erroneous belief that the system is in an unsafe state.

All the research described above highlights the significance of security and resilience in CPS. Figure 2.1 provides a summary of the previous work discussed in this section.

2.2 Attack Detection in CPS

2.2.1 A Brief Introduction to Anomaly Detection

To detect attacks against CPS, anomaly-based detection methods can be employed. What distinguishes anomaly-based detectors used in CPS from intrusion detection systems (IDS) used in IT systems is that the latter is based on creating models of network traffic or software behavior whereas the former is based on a representative model of the physical system. There also exist other differ-

Table 2.1: Differences in Attack Detection Between IT Systems and CPS [97]

IT	CPS
Monitoring host- or network-level user/machine activity (e.g., an HTTP request or a web server).	Monitoring the physical processes (and hence laws of physics) which govern behavior of physical devices.
Monitoring user-triggered activities, leading to high false positive rates due to the unpredictability of user behaviors	Monitoring activities which are frequently automated, providing some regularity and predictability for behavior monitoring.
Dealing with mostly non-zero-day attacks, making knowledge-based detection effective.	Dealing with zero-day or highly sophisticated attacks, hence making knowledge-based detection ineffective.
Rarely dealing with legacy components, making modeling of the physical processes governing legacy components unnecessary.	Often dealing with legacy technology, making physical model-based detection an effective technique by specifying the physical processes governing behavior of legacy components.

ences in attack detection between IT systems and CPS that are summarized in Table 2.1 [97]. From this point forward, when we refer to anomaly detectors, we mean anomaly detection in the context of CPS unless otherwise stated.

An anomaly-based detector receives as inputs the sensor measurements from the physical system and the control commands sent to the physical system, and then uses them to identify any suspicious sensor or control commands. Figure 2.2 illustrates a general diagram of such security monitoring architecture. This architecture contains: (1) the physical phenomena of interest (i.e., plant), (2) sensors to observe the physical system and send a time series y_k denoting the value of the measurement at time k, (3) the controller that sends control commands u_k to actuators based on the sensor measurements received, (4) actuators that change the control command to an actual physical change, and (5) a detection module that contains two subcomponents (i.e., predictor and statistical test) as described below.

As shown in Figure 2.3, the detection module comprises two main parts: (1) A predictor which given sensor measurements y_k and control commands u_k, predicts the future expected measurements $\hat{y}_{k+1}$, and (2) An anomaly detector (or statistical test) which given a time series of residuals r_k (i.e., the difference between the received sensor measurement y_k and the predicted measurement $\hat{y}_k$) determines whether to raise an alarm, denoted by H_1, or not, denoted by H_0. Note that in this scheme, it is assumed that the detection module is trusted and its output cannot be attacked. But,

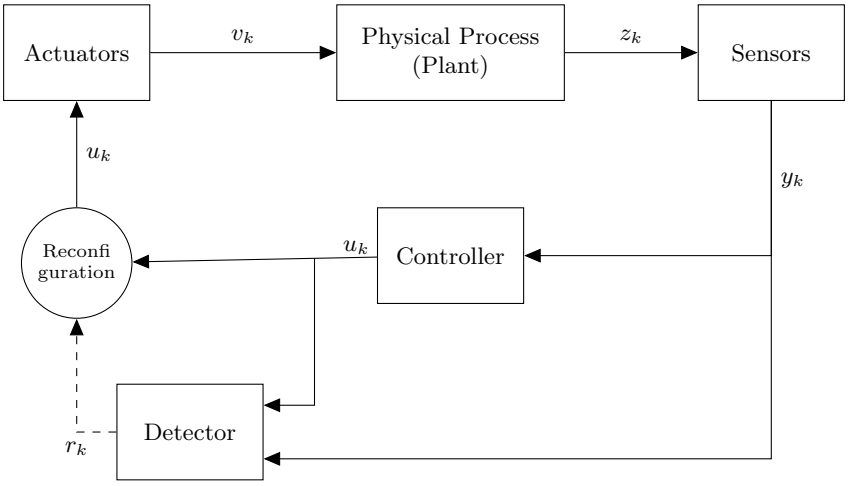


Figure 2.2: Anomaly detection architecture.

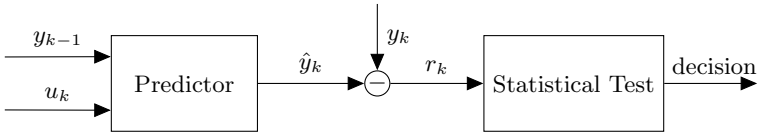


Figure 2.3: Detector components.

the inputs provided to the detection module may be attacked. In particular, in a sensor attack, the attacker deceives the controller about the real state of the plant, and so $z_k \neq y_k$. Further, in an actuator attack, the control command sent to the plant is modified, i.e., $u_k \neq v_k$.

Based on the observed sensor or control signals up to time k, we can use data-driven or model-based methods of the system to predict the expected observations $\hat{y}_{k+1}$. The difference between the predictions $\hat{y}_{k+1}$ computed by the predictor and the measurements y_{k+1} received from the sensors is called a residual r_k. If the sensor measurements y_k are significantly different from the predictions (i.e., if the residual is large), a detection alarm can be generated. In a *stateless test*, an alarm is raised for every single significant deviation at time k, i.e., an alarm is raised if $|y_k - \hat{y}_k| = r_k \geq \eta$ where η is a threshold. In a *stateful test*, an additional statistic S_k is computed that keeps track of the historical changes of r_k. Then, a detection alarm is generated if the statistic exceeds a threshold, i.e., $S_k \geq \eta$, which can be due to either a single large deviation or a persistent deviation across multiple timesteps.

There are many tests that can keep track of the historical behavior of the residual r_k such as taking an average over a time-window, an exponential weighted moving average (EWMA), or using change detection statistics such as the non-parametric CUmulative SUM (CUSUM) statistic [11]. The CUSUM detector assumes a known probability model for observations, which is not suitable

Table 2.2: Detection Methods

Features		Statistical Test	
Cur. In. & Prev. Out.	u_k, y_{k-1}	Stateless	$\lvert r_k\rvert \underset{H_0}{\overset{H_1}{\gtrless}} \eta$
Prev. Sensor Observ.	$y_{k-1}, \ldots, y_{k-N}$	Stateful	$(S_k + \lvert r_k\rvert - b)^+ \underset{H_0}{\overset{H_1}{\gtrless}} \eta$

Table 2.3: Errors in Anomaly Detection

	H_0 is true	H_1 is true
Accept H_0	Correct decision	False negative (missed detection)
Accept H_1	False positive (false alarm)	Correct decision

for attacks that are unknown by definition, and so non-parametric CUSUM (i.e., CUSUM without probability likelihood models) is typically used in the CPS security literature. The non-parametric CUSUM statistic is defined recursively as $S_0 = 0$ and $S_{k+1} = (S_k + |r_k| - b)^+$, where $(x)^+$ represents $\max(0, x)$ and b is selected so that the expected value of $|r_k| - b$ is less than zero under hypothesis H_0 (i.e., b prevents S_k from increasing consistently under normal operation). An alert is generated whenever the statistic is greater than a previously defined threshold $S_k > \eta$ and the test is restarted with $S_{k+1} = 0$. Table 2.2 provides a summary of different detection methods.

As illustrated in Table 2.3, in anomaly detectors, there might be a *false negative*, which means failing to raise an alarm when an anomaly did happen. Further, there might be a *false positive*, which means raising an alarm when the system exhibits normal behavior. It is desirable to reduce the FP and FN probabilities as much as possible. But, there exists a trade-off between them, which can be controlled by changing the threshold η. In particular, by decreasing (increasing) the threshold η, one can decrease (increase) the false-negative $FN(\eta)$ and increase (decrease) the false-positive probability $FP(\eta)$. It is possible to plot the false-positive probability $FP(\eta)$ as a function of the false-negative probability $FN(\eta)$ for various threshold values [43].

2.2.2 Model-Based Attack Detection

In this section, we review the literature on model-based attack detection in CPS. Table 2.4 provides a summary.

Attack detection in industrial process control systems is studied in [28]. An anomaly-based detector is designed using a predictor, which is based on an LTI model of the physical system, and a non-parametric CUSUM test. As the attack model, an adversary that launches sensor attacks is

Table 2.4: Summary of related work on attack detection in CPS.

	Detection		Attack		Monitoring	
Paper	Stateful	Stateless	Sensor	Actuator	Active	Static/Dyn.
[108]	-	✓	✓	✓	-	✓
[98]	-	✓	✓	✓	✓	-
[27]	✓	-	✓	-	-	✓
[42]	-	✓	✓	-	-	✓
[9], [8]	-	✓	✓	-	-	✓
[133]	✓	-	✓	-	✓	-
[128], [130]	-	✓	✓	✓	-	✓
[57]	✓	-	-	✓	-	✓
[64]	-	✓	✓	-	-	✓

considered. The adversary has the goal of raising the pressure of a tank beyond safety levels while remaining undetected. The paper characterizes attacks that cannot be detected by the detector and then evaluates their effects by running experiments. It is concluded that the considered case study is resiliently-designed since none of the simulated stealthy attacks result in a safety violation.

Considering a linear descriptor system, fundamental monitoring limitations for CPS are characterized in [108]. An attack is proven to be undetectable by static, dynamic, and active monitors if and only if it excites only the zero dynamics of the system. Such undetectable attacks, which are also known as zero-dynamic attacks, can be performed by only compromising the actuators. While study of zero-dynamic attacks is useful for resilience analysis, most systems are not vulnerable to them (for example, if states are directly measured or if there is no need for state estimation). Even for the vulnerable systems, the attacker has restrictions on achieving its objective since specific scenarios must be followed in launching an attack. To secure the CPS against zero-dynamic attacks, methods which modify the structure of the system are proposed [130], [128].

In active monitoring, unpredictable control commands are sent to the CPS, and then, it is verified whether the sensors respond as expected. If the attacker does not adapt its attack to the unpredictable control command, it may be detected. Active monitoring is applied to state estimation by embedding a watermark in the control signal [98], [99], [100]. It is also used in power systems [101], [33], and other domains, [122], [133]. While active monitoring is a useful detection strategy for systems that are in their steady state, it is not suitable to the systems with highly variable control signal (e.g., in frequency generators in power systems). Even for systems in steady state, if an attacker has perfect knowledge, it can design attack vectors that bypass the detector.

In addition, active monitoring results in performance losses caused by deviating from the optimal control signal. While there have been attempts to minimize this performance loss [100], this may still be an undesirable behavior.

Combined cyber and physical attacks are investigated in [9], [8]. In the attack model, the adversary launches physical attacks on a water distribution system in order to steal water, and then performs sensor attacks to hide the effects of the physical attack. To detect such attacks, the paper proposes using unknown input observers. Nevertheless, it is stated that if the adversary has enough resources to attack enough pipes and sensors, it can always remain stealthy. Such covert attacks are also characterized for linear and nonlinear systems [123], [124].

The effects of adding security measures to a control system is experimentally characterized in [54]. It is shown that adding security measures can create additional time delay that can degrade the system's performances. To compensate for the adverse effect of added security, methods that change the control parameters in real-time are proposed. Further, a framework to optimize the performance and security trade-offs in CPS is proposed in [142]. The framework is based on an extension of evolutionary algorithms known as coevolutionary genetic algorithms, and is able to find the Nash equilibrium for the trade-off model.

An attack detection mechanism based on the energy balance of a system is proposed in [42]. The approach uses passivity which is a property indicating that the system dissipates more energy than it generates. To detect attacks, the supplied energy is estimated, and then it is compared to the sum of the dissipated and stored energy in the CPS. While this approach is novel, its benefits over observer-based methods is not clear. In addition, if the attack is passive, it always remains undetected. Also, functions for energy dissipation, which are required for the detector to work, may be hard or even impossible to obtain for some cases.

Unsupervised clustering has been used to detect attacks in process control systems [64], [70]. The clusters represent the relationship between variables of a process and attacks are identified as anomalies that do not fit the clusters. The approach has the advantage of creating models of the physical systems without a priori knowledge. However, its performance may be weak since it does not consider the time-evolution of the system or the evolution outside of a steady state.

Model-based attack detection has also been studied in other domains such as power systems [21], [48], [63], [114], [137], and medical systems [57], [58]. In power systems, it is shown that an attacker can inject false sensor signal for state estimation that will not raise an alarm [85]. There has been some follow up research on this problem [138], [32], [125], [79]. Further, in medical systems, overdoes attacks on insulin pumps are studied in [57]. Supervised learning methods are employed to learn

normal patient infusion patterns, and then, detection is performed using an average of the residuals.

2.3 Game Theory for Detection

In this section, we review game-theoretical approach for the problem of attack detection in CPS.

Game-theoretic principles are formally applied to intrusion detection to develop a decision and control framework in [3], [4]. Distributed intrusion detection is studied as a game between an IDS and an attacker using a model that represents the imperfect flow of information from the attacker to the IDS through a network. Then, the existence of a unique Nash equilibrium and best-response strategies is investigated, and long-term interactions are analyzed using repeated games and a dynamic model. Further, the detection framework is extended using a stochastic and dynamic game in which the sensors observing and reporting the attacks to the IDS are modeled as a finite-state Markov chain [5]. Then, the game is investigated under different assumptions on the information available to players.

The problem of detecting and classifying an attacker based on its attack type is studied in [38]. In the model, the defender strategically selects its classification policy by choosing a threshold that balances the cost of missed detections and false alarms, while the attacker tries to maximize its payoff based on its type. A characterization of the Nash equilibria is then presented in mixed strategies that can be computed in polynomial time. Furthermore, randomized detection thresholds using a general model of adversarial classification is studied in [82]. The goal of using randomized thresholds is to prevent an attacker from launching stealthy attacks that stay just below the threshold. Both Nash and Stackelberg equilibria are analyzed based on the true-positive to false-positive curve of the detector. Then, it is shown that the randomized thresholds may force an attacker to design less harmful attacks, which lowers the expected cost of the defender.

Using a zero-sum non-stationary stochastic game, the problem of finding the optimal policy that switches between control-cost optimal and secure controllers in the presence of replay attacks is proposed in [95]. It is proven that the optimal strategy exists, and a suboptimal algorithm is proposed. Further, an approach for finding optimal detection thresholds for multiple IDS employed in CPS is proposed in [74]. In the model, the attacker mounts an attack against a subset of systems and the defender detects and mitigates the attack if the IDS of at least one targeted system raises an alarm. It is shown that finding optimal attacks and defenses is computationally expensive, and then polynomial-time heuristic algorithms are proposed for computing approximately optimal strategies.

The problem of strategic threshold-selection by a collection of independent self-interested users is

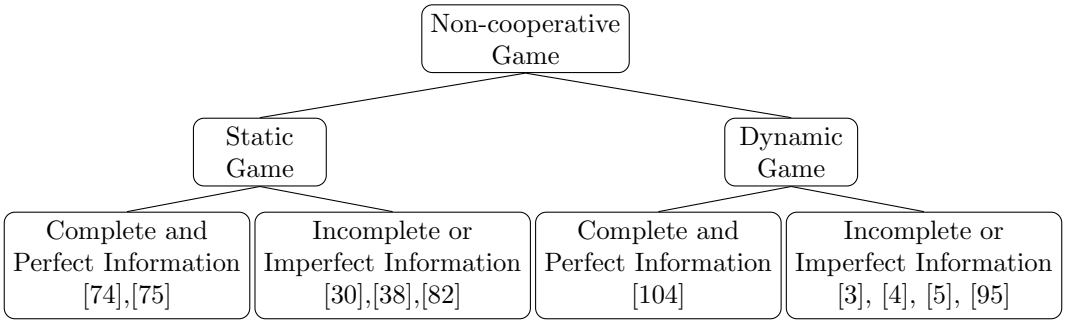


Figure 2.4: Summary of related work on game-theoretical approaches for anomaly detection in CPS.

considered in [75]. The Stackelberg multi-defender equilibria corresponding to short-term strategic dynamics is characterized, and a polynomial-time algorithm for computing short-term equilibria is presented. The Nash equilibria of the simultaneous game between all users and the attacker is also characterized which models long-term dynamics. It is concluded that if an equilibrium exists, in both cases, it is unique and socially optimal. Moreover, the problem of optimal signature-based IDS configuration under resource constraints is studied in [145]. The work uses a cooperative game to study intrusion detection according to some known attack graphs.

Signaling games are also used to study intrusion detection [40]. For instance, an intrusion detection game based on the signaling game is proposed in order to select the optimal detection strategy that lowers resource consumption in [121]. Intrusion detection in heterogeneous networks, in which detectors monitor the nodes that can be targeted by attackers, is investigated in [30]. The attackers' actions are defined as the probabilities of attacking each of the targets, and the defender's actions are defined as the probabilities of defending against attacks by considering the false alarm rate and the detection rate of the IDS. The best defense strategies are computed for the cases of one or multiple IDS monitoring each attack target. The defender's strategy is determined by the amount of resources that the IDS allocate to each of the targets.

Figure 2.4 provides a taxonomy of the related work discussed in this section. The papers are categorized based on the type of game (i.e., dynamic or static) and the information available to players.

2.4 Machine Learning for Detection

Machine learning has been widely used to study anomaly detection [29], [131]. But, there is little work that applies machine learning classification methods to attack detection in CPS. This may be partially due to the lack of publicly available intrusion data for CPS. Nonetheless, if this limitation

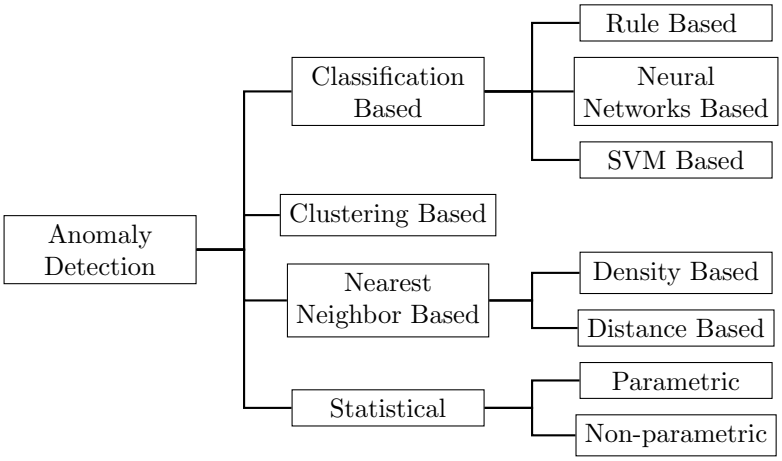


Figure 2.5: Anomaly detection methods using machine learning [29].

can be overcome through using testbeds or simulation environments, machine learning techniques can be highly beneficial for resilient detection in CPS. For instance, machine learning methods can be used to construct data-driven estimators to predict the future states, build supervised- or unsupervised-learning anomaly detectors (as shown in Figure 2.5), and develop secure learning methods for resilient detection.

2.4.1 Attack Detection

A behavior-based machine learning approach for attack detection in CPS is proposed in [61]. The work models the physical process of CPS to detect any anomaly or attack that may try to change the system's behavior. Using a replicate of a real water treatment facility, different data-driven anomaly detectors are implemented and their performances are evaluated. While this work successfully applies machine learning algorithms for attack detection in CPS, the studied attack scenarios are very restrictive. Thus, it is unclear whether the high detection rates reported in the paper are due to the effectiveness of the approach or due to the restrictive attack models. Using the same testbed, the impact of single-point cyber-attacks are experimentally investigated in [1]. Cyber-attacks are launched through a SCADA server connected to programmable logic controllers that govern the actuators and sensors. Several experiments are performed considering different objectives for the attacker. Then, based on the observed experiment results, attack detection mechanisms are proposed.

Data-driven anomaly detection is studied using a miniature industrial gas system with a few sensors and actuators in [102]. To detect attacks, one physical process attribute (i.e., pressure in

pipeline) and two one-class classifiers (i.e., support vector data description and kernel principal component analysis) are used. Then, different attack scenarios are considered to evaluate the detection performance. According to the results presented in the paper, while single-point attacks have a detection rate of 99.25 percent, more complicated attacks have much lower detection rates of approximately 65-70 percent. This shows that the detector's performance significantly decreases when faced by complex attacks. Further, an intrusion detection method based on neural networks is investigated using a water testbed that contains a tank, pump, and level sensor [45]. Network traffic, SCADA mode, water level, and pump status are used as the attributes, and then the method is evaluated considering malicious attacks on the level sensor. Finally, using the same testbed, a one-class approach is studied in [103].

An intrusion detection system that uses neural network-based modeling is proposed in [81]. Network data recorded from an existing critical infrastructure is used as normal behavior, and randomly generated data is used to model intrusions. To learn the model, a combination of two neural network learning algorithms, namely Error back propagation and Levenberg-Marquardt, is employed. Then, the method is evaluated using network data, and it is claimed that the detection approach achieves a perfect detection rate while generating no false positives on test data.

An unsupervised data-driven framework for system-wide anomaly detection is presented in [83]. The framework involves a spatiotemporal feature extraction scheme for discovering and representing causal interactions among the subsystems of a CPS, and a free energy estimation of system-wide patterns using a Restricted Boltzmann Machine (RBM). It is stated that the proposed framework can capture multiple nominal modes with one graphical model and can be effectively used for anomaly detection. Moreover, a formal methods approach to the problem of intrusion detection for CPS security is discussed in [60]. The proposed algorithmic method is capable of inferring a data classifier in the form of a signal temporal logic formula from unlabeled data. The inferred formula can be interpreted in natural language and can be used for online monitoring.

A Bayesian network approach for learning the causal relations between cyber and physical variables as well as their temporal correlations from unlabeled data is presented in [69]. Data transformations are performed to deal with the heterogeneous characteristics of the cyber and physical data so that the integrated dataset can be used to learn the Bayesian network structure and parameters. Scalable algorithms are then presented to detect different anomalies and isolate their respective root-cause using a Bayesian network. The algorithm is evaluated using an unlabeled dataset consisting of anomalies due to faults and cyber-attacks in a commercial building system.

2.4.2 Adversarial Learning

Adversarial learning studies secure adoption of machine learning techniques in adversarial settings. Adversaries can be categorized based on the influence on classifier, security violation, and specificity [20]. The influence is either *causative* which undermines the learning algorithm to cause misclassifications via influencing training data and potentially test data, or *exploratory* which causes misclassifications via affecting test data. The security violation can be an *integrity* violation which allows the adversary to access protected resource and cause misclassification of illegitimate samples, an *availability* violation which denies service to users and can cause misclassification of legitimate and illegitimate samples, or *privacy* violation which allows the adversary to obtain confidential information. The specificity of an attack can be either *targeted* or *indiscriminate*.

Adversarial classifier reverse engineering (ACRE) learning problem is introduced in [86]. In ACRE, an adversary launches an attack to minimize a cost function, while having limited information about the classifier. The adversary is allowed to make polynomial number of membership queries to modify an attack instance to bypass the classifier with minimal cost. In the paper, a query algorithm for reverse engineering of linear classifiers is presented. Furthermore, in [31], a framework for the adversarial learning problem is presented that uses game theory. The paper formulates a game between a classifier and an adversary, and then assuming that the adversary's optimal strategy is known, develops the optimal classifier. Then, experiments are performed which show that the proposed robust classifier outperforms a standard one.

Classifiers that are optimal with respect to a worst-case scenario of feature deletion at test time are investigated in [51]. A minmax game is formulated to formalize the interaction between a classifier and a feature removal mechanism, and it is assumed that the players know the strategy space of each other. The same problem is studied using a zero-sum sequential Stackelberg game where the adversary acts as the leader and the classifier acts as the follower [84].

Optimal support vector machine (SVM) learning strategies are developed considering two attack models, namely free-range and restrained [144]. The strategies minimize the hinge loss given that the adversary maximizes the hinge loss by corrupting the data. Experiments are performed using different datasets, which report improvements in resilience over standard SVM. The approach is extended by considering the hierarchical mixtures of experts in the framework [143].

Feature reduction in adversarial settings is studied in [77]. In particular, the paper considers a model of an adversary that performs feature cross-substitution attacks. To make learning more robust to such attacks, a heuristic method is presented. In addition, as a general solution, a mixed-

integer program with constraint generation is presented. The program implicitly trades off overfitting and feature selection in an adversarial setting using a sparse regularizer along with an evasion model. Furthermore, a general-purpose scalable retraining framework that can boost robustness of an arbitrary learning algorithm in the face of adversarial models is proposed in [78].

The literature on adversarial learning is extensive. Previous work also studies evasion attacks [15], [17], [19], [72]; poisoning attacks [18], [16], [118]; adversarial examples against machine learning [52], [71], [120]; and secure learning [23], [22].

2.5 Comparison to this Dissertation

The work presented in this thesis addresses specific problems in view of the challenges highlighted in Section 1.2. In particular, we seek to develop methods for the design, analysis, and evaluation of resilient anomaly detectors used in CPS. As stated above, what distinguishes our work from previous work is incorporating domain-specific properties of the underlying physical system in the detector design. In particular, in Chapter 3, we define a notion of damage caused by attacks that depends on the state of the physical system. This is then used to optimally configure anomaly detectors such that they become highly sensitive in critical states. In Chapter 4 and 5, we model the adverse effects of incorrect detection decisions on the physical system. We then optimally re-design the detector so that such adverse effects are minimized. In Chapter 6, we study the problem of adversarial regression in CPS. Aside from the novelty in the problem formulation, what makes this chapter unique is that it provides practical evidence to support the benefits of resilient anomaly detectors. Overall, exploiting the tight interaction between the anomaly detector and the physical system for the benefit of improved resilience, is what distinguishes this work form previous works.

Chapter 3

A Game-Theoretic Approach for Selecting Optimal Thresholds for Attack Detection in Dynamical Environments

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

3.1 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 malicious attack could 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 [56] 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 [11]. 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 [55].

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. 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 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.

The rest of this chapter is organized as follows. In Section 3.2, we discuss related work. In Section 3.3, we introduce the system model. In Section 3.4, we present the game-theoretic model and define the problem of computing optimal time-dependent detection thresholds. In Section 3.5, we analyze the time-dependent detection strategy and present an algorithm to obtain optimal thresholds. We also provide an algorithm to compute optimal fixed thresholds, which do not change with time. In Section 3.6, we present the problem of computing optimal time-dependent thresholds in the presence of both faults and attacks, and we present an algorithm to solve the problem. In Section 3.7, we evaluate our algorithm using a case study of contamination attacks in water distribution systems. Finally, we offer concluding remarks in Section 3.8.

3.2 Related Work

As discussed in Chapter 2, the problem of optimal design of anomaly detection systems has been studied in a variety of different ways in the academic literature [132, 29]. Nevertheless, to the best of our knowledge, prior work has not particularly 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 [2]. 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 [136]. 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 [88]. 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.

In a game-theoretic setting, signaling games have been used to model intrusion detection [109, 40].

An intrusion detection game based on a signaling game is proposed in order to select the optimal detection strategy that lowers resource consumption [121]. 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 [3, 4]. 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 [134, 73]. 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 [110]. 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.

Contaminant intrusion in water distribution network has been considered in water security literature [34, 35]. In particular, data-driven water monitoring approaches have received considerable attention due to the advances in smart monitoring technologies [94, 65]. Bayesian sequential analysis is integrated with neural network models to detect possible quality threats in water distribution systems [113]. Further, a dynamic thresholds scheme for contamination event detection is presented by defining optimal detection thresholds as the ones that maximize detection rate [10]. 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 [49]. 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 [90]. 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 chapter, we showed that time-dependent thresholds significantly outperform the fixed threshold in terms of minimizing the losses.

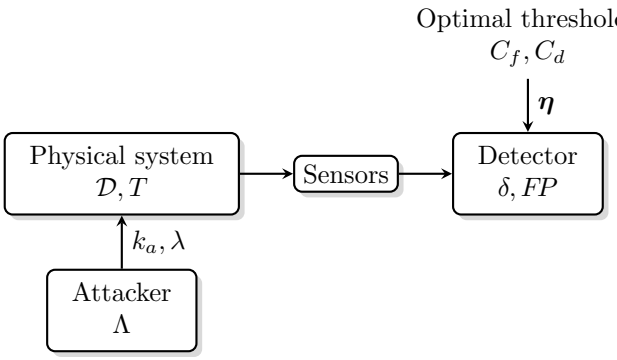


Figure 3.1: System description.

3.3 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 [50]. 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 3.1 shows a list of symbols used in this chapter. In addition, Figure 3.1 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

Table 3.1: 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)

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 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 [11]. 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) [11].

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 $\delta : \mathbb{R}_+ \times \Lambda \rightarrow \mathbb{N} \cup \{0\}$, where $\delta(\eta_k, \lambda)$ is the detection delay (in timesteps) when the threshold is η_k and the type of the attack is $\lambda \in \Lambda$. We assume that for each $\lambda \in \Lambda$, $\delta(\eta_k, \lambda)$ is a continuous function of η_k. Further, we denote the false-positive probability (i.e., probability of raising a false alarm during a single timestep) by $FP : \mathbb{R}_+ \rightarrow [0, 1]$, where $FP(\eta_k)$ is the false-positive probability when the detection threshold is η_k. We assume that FP is decreasing and δ is increasing with respect to η_k, which is true for most typical detectors, including sequential change detectors. For example, in Section 3.7, we obtain detection delay and false-positive probability for a CUSUM detector.

3.4 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.

3.4.1 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 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,

$$\mathcal{P}(\boldsymbol{\eta}, k_a, \lambda) = \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda) \ , \tag{3.1}$$

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 $N = |\Gamma|$, 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

$$\mathcal{L}(\boldsymbol{\eta}, k_a, \lambda) = N \cdot C_d + \sum_{k=1}^{T} C_f \cdot FP(\eta_k) + \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda) \ , \tag{3.2}$$

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.

3.4.2 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* (3.1).

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 [66].

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*

$$\underset{\substack{\boldsymbol{\eta},\\ (k_a,\lambda)\in bestResponses(\boldsymbol{\eta})}}{\operatorname{argmin}} \quad \mathcal{L}(\boldsymbol{\eta}, k_a, \lambda), \tag{3.3}$$

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

Our objective is to compute the optimal thresholds efficiently.

3.5 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 3.1) 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 3.2) 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:

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

Algorithm 3.1 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
            $S(n, \boldsymbol{m}, \eta_{\text{prev}}, \eta) \leftarrow \textsc{Cost}(n+1, \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}, \eta) + C_f \cdot FP(\eta) + C_d$
          end if
        end for
        $\eta^*(n, \boldsymbol{m}, \eta_{prev}) \leftarrow \text{argmin}_\eta\, S(n, \boldsymbol{m}, \eta_{prev}, \eta)$
        $\textsc{Cost}(n, \boldsymbol{m}, \eta_{prev}) \leftarrow \min_\eta S(n, \boldsymbol{m}, \eta_{prev}, \eta)$
      end if
    end for
  end for
end for
$\boldsymbol{m} \leftarrow \langle 0, \ldots, 0 \rangle,\ \eta_0^* \leftarrow$ arbitrary
for all $n = 1, \ldots T$ **do**
  $\eta_n^* \leftarrow \eta^*(n, \boldsymbol{m}, \eta_{n-1}^*)$
  $\boldsymbol{m} \leftarrow \langle \min\{\delta(\eta_n^*, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}$
end for
return $(\textsc{Cost}(1, \langle 0, \ldots, 0 \rangle, \text{arbitrary}), \boldsymbol{\eta}^*)$

Algorithm 3.2 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**
  $(TC(P), \boldsymbol{\eta}^*(P)) \leftarrow \textsc{MinimumCostThresholds}(P)$
end for
$P^* \leftarrow \text{argmin}_{P \in \mathit{SearchSpace}} TC(P)$
return $\boldsymbol{\eta}^*(P^*)$

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*

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

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 $\eta_k^* = \max_{\eta\,:\,\delta(\eta,\lambda) \le \delta(\eta_k,\lambda)} \eta$. 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, $\eta_k \le \eta_k^*$ 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. □

Consequently, for the remainder of this chapter, 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 3.1) 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 search (Algorithm 3.2) computes the optimal thresholds by finding an optimal damage bound P and using Algorithm 3.1. In the first algorithm, we use a dynamic-programming approach, iterating backwards through the timesteps. For each timestep, we assume that the optimal thresholds for the remaining timesteps (under certain conditions) have already been computed, and we compute the optimal threshold for the current timestep in polynomial-time. In the second algorithm, we use an exhaustive search but we show that the cardinality of our search space is polynomial in the size of the input.

Lemma 1. *For any given damage bound $P \in \mathbb{R}$, Algorithm 3.1 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

$$\forall k_a \in \{1, \ldots, T\}, \lambda \in \Lambda : \ \mathcal{P}(\boldsymbol{\eta}, k_a, \lambda) \le P\,. \tag{3.4}$$

The algorithm returns the minimum cost attained, or if no thresholds exist satisfying (3.4)*, it returns infinity as the cost.*

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

$$\min_{(\eta_n, \ldots, \eta_T)} \sum_{k=n}^{T} C_f \cdot FP(\eta_k) \tag{3.5}$$

subject to

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

If there are no thresholds that satisfy the damage bound P under these conditions, we let $\textsc{Cost}(n, \boldsymbol{m})$ be ∞.[1]

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

$$\textsc{Cost}(n, \boldsymbol{m}) = \begin{cases} \infty & \text{if } \bigvee_{\lambda \in \Lambda} \sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P, \\ \min_\eta \textsc{Cost}(n+1, \langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}) + C_f \cdot FP(\eta) & \text{otherwise.} \end{cases} \tag{3.6}$$

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

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 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, $\textsc{Cost}(1, \langle 0, \ldots, 0 \rangle)$ 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 (3.6). 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 $\eta_1^*, \ldots, \eta_{n-1}^*$.[2]

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 (3.6)). Otherwise, the cost also has to include the cost C_d of changing the threshold. Consequently, we first define

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

Then, similar to Equation (3.6), 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. □

[2]Note that in Algorithm 3.1, 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.

Next, we show how Algorithm 3.2 finds optimal thresholds.

Theorem 2. *Algorithm 3.2 computes optimal thresholds that minimize the defender's loss (see Definition 3).*

Proof. For any damage bound P, using the algorithm MinimumCostThresholds (Algorithm 3.1), 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

$$\min_{P} \; TC(P) + P \tag{3.7}$$

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 $\mathcal{L}'$ is lower than the loss $\mathcal{L}^*$ for the solution $\boldsymbol{\eta}^*$ of the above formulation. Let P' be the damage resulting from the attacker's best-response 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 3.2 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

$$\left\{ \sum_{k=k_a}^{k_a+\delta} \mathcal{D}(\lambda, k) \;\middle|\; \exists\, k_a \in \{1, \ldots, T\},\, \delta \in \Delta,\, \lambda \in \Lambda \right\}. \tag{3.8}$$

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. □

Proposition 1. *The running time of Algorithm 3.2 is* $\mathcal{O}(T^2 \cdot |\Delta|^{|\Lambda|+2} \cdot |\Lambda|^2 \cdot |E|)$.

Note that since detection delay values can be upper-bounded by T, the running time of Algorithm 3.2 is also $\mathcal{O}(T^{|\Lambda|+4} \cdot |\Lambda|^2 \cdot |E|)$.

Proof. In the dynamic-programming algorithm (Algorithm 3.1), 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 3.2 is $\mathcal{O}(T^2 \cdot |\Delta|^{|\Lambda|+2} \cdot |\Lambda|^2 \cdot |E|)$. □

Finally, note that the running time of the algorithm can be substantially reduced in practice by computing $\textsc{Cost}$ 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.

3.5.1 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 present Algorithm 3.3. 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 3.3 computes an optimal fixed threshold in* $\mathcal{O}(T \cdot |E| \cdot |\Lambda|)$ *steps.*

Algorithm 3.3 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**
        $P' \leftarrow \mathcal{P}(\eta, k_a, \lambda)$
        $L' \leftarrow P' + C_f \cdot FP(\eta) \cdot T$
      end if
    end for
  end for
  if $L' < L^*$ **then**
    $L^* \leftarrow L'$
    $\eta^* \leftarrow \eta$
  end if
end for

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 $\mathcal{O}(T \cdot |E| \cdot |\Lambda|)$. □

3.6 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

$$\mathcal{P}_F(\boldsymbol{\eta}) = \frac{1}{T \cdot |\Lambda|} \sum_{k_a=1}^{T} \sum_{\lambda \in \Lambda} \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda). \tag{3.9}$$

Then, the combined loss due to faults and attacks can be represented as the average of the loss (3.9) due to random faults and the loss (3.1) due to attacks. Therefore, the defender's total loss

with both random faults and best-response attacks is

$$\mathcal{L}_C(\boldsymbol{\eta}, k_a, \lambda) = N(\boldsymbol{\eta}) \cdot C_d + \sum_{k=1}^{T} C_f \cdot FP(\eta_k) + \frac{1}{2}(\mathcal{P}_F(\boldsymbol{\eta}) + \mathcal{P}(\boldsymbol{\eta}, k_a, \lambda)) \tag{3.10}$$

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

$$\underset{\substack{\boldsymbol{\eta}, \\ (k_a,\lambda)\in \text{bestResponses}(\boldsymbol{\eta})}}{\operatorname{argmin}} \mathcal{L}_C(\boldsymbol{\eta}, k_a, \lambda),$$

Algorithm. First, we define the following subproblem, given that P is a real number.

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

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, using the same argument presented in Theorem 2, we can find optimal thresholds by solving,

$$\min_{P} TC_C(P) + P, \tag{3.11}$$

and an optimal solution $\boldsymbol{\eta}^*$ to $TC_C(P)$ for an optimal P is also an optimal solution to (3.11).

To solve $TC_C(P)$, we define 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 $TC_C(P) = \text{Cost}(P, 1, (0, \ldots, 0), \eta_{n-1})$ for any η_{n-1}, 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,

$$\text{Cost}(P, n, \boldsymbol{m}, \eta_{n-1}) =$$
$$\begin{cases} \infty & \text{if } \bigvee_\lambda \frac{1}{2} \sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P, \\ \min_{\eta_n} \quad \text{Cost}(P, n+1, \langle \min\{\delta(\eta_n, \lambda), m_\lambda + 1\}\rangle_{\lambda \in \Lambda}, \eta_n) + 1_{\{\eta_{n-1} \neq \eta_n\}} C_d & \\ \qquad\quad + C_f \cdot FP(\eta_n) + \frac{1}{2} \frac{1}{T \cdot |\Lambda|} \sum_{\lambda' \in \Lambda} \sum_{k'_a = n - m_{\lambda'}}^{T} 1_{\{n \geq \sigma(\boldsymbol{\eta}, k'_a, \lambda')\}} \mathcal{D}(k, \lambda') & \text{otherwise.} \end{cases} \tag{3.12}$$

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.

3.7 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.

3.7.1 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 [50]. To model normal behavior, we use data collected by a utility in the United States available at [25]. 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[3]. 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

[3]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 [55].

considering contamination attacks.

3.7.1.1 Contamination Attack.

We simulate contamination attacks using the approach presented in [65, 94]. For each water quality parameter i, the data collected from the water quality sensors is normalized by subtracting the dataset mean μ_i from each water quality measurement $x_i(k)$, and dividing this difference by the standard deviation σ_i of the dataset, i.e., $z_i(k) = \frac{x_i(k)-\mu_i}{\sigma_i}$. Then, contamination attacks, which are characterized by their magnitude, are simulated and superimposed in the normal data. That is, for an attack of magnitude λ_i, where $\lambda_i \geq 1$, we multiply $z_i(k)$ by λ_i. Then, we denormalize (i.e., return to original scale) the data via multiplying $\lambda_i \cdot z_i(k)$ by σ_i and adding μ_i to the result. Note that this is a typical method of generating contamination events [113]. These mentioned magnitudes can represent the sensitivity of a quality parameter to different toxic chemicals, where a large magnitude means the quality parameter is highly sensitive to the specific chemical. A list of toxic chemicals and their impact on different quality parameters can be found in [55].

3.7.1.2 Damage Function.

Figure 3.2 presents a typical water demand during a day [92]. 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.

3.7.2 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.

3.7.2.1 Estimator.

We construct an estimator using an artificial neural network (ANN) for each water quality parameter [113]. For each parameter, the inputs to its corresponding ANN are the parameter's lagged

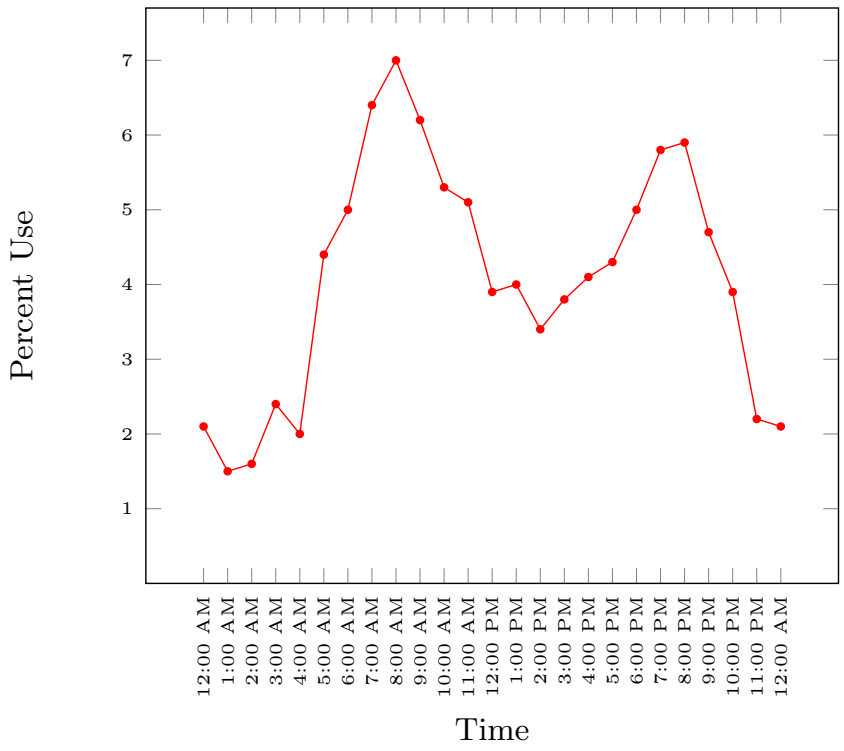


Figure 3.2: Hourly water demand during a day [92].

Table 3.2: 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

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 [111]. Table 3.2 shows the estimator's performance using mean squared error (MSE) and coefficient of determination (R^2) as performance criteria.

3.7.2.2 Detection Algorithm.

We use the CUSUM method as the detection algorithm. CUSUM is a sequential algorithm frequently used for change detection [105, 136]. 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 3.3, 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 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 3.3, 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.

3.7.3 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 3.2 to compute the optimal time-dependent threshold. Figure 3.4 shows the obtained thresholds for each timestep. The resulting optimal loss is

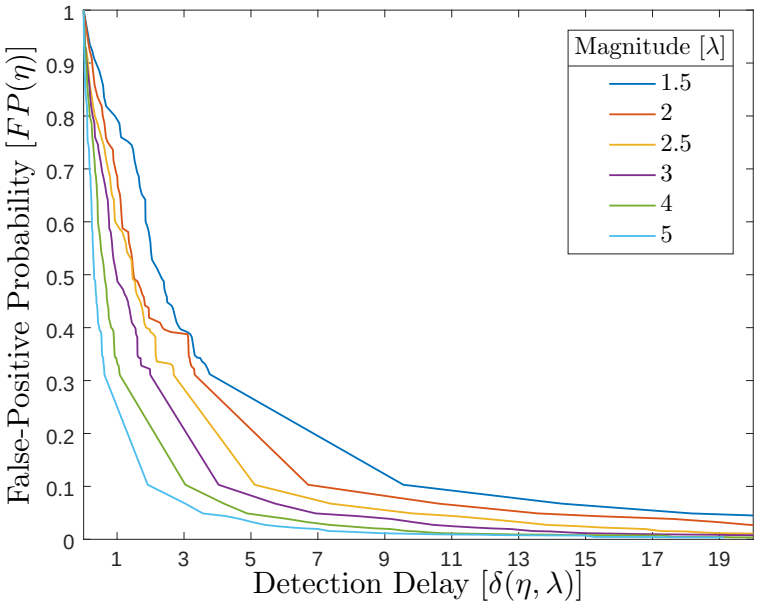


Figure 3.3: Trade-off between detection delay and false-positive probability (total chlorine).

$\mathcal{L}^* = 187.72$. Figure 3.4 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 $\mathcal{P}^* = \sum_{k=116}^{120} \mathcal{D}(k, \lambda) = 120.00$. 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 $\mathcal{L}^* = 222.45$. Figure 3.5 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 $\mathcal{P}^* = \sum_{k=44}^{44+6} \mathcal{D}(k, \lambda) = 144.00$. Note that if the attacker starts the attack at any other timestep, the damage caused before detection is less than $\mathcal{P}^*$. We observe that the optimal loss obtained by the time-dependent threshold is significantly smaller than the loss obtained by the fixed threshold.

3.7.3.1 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

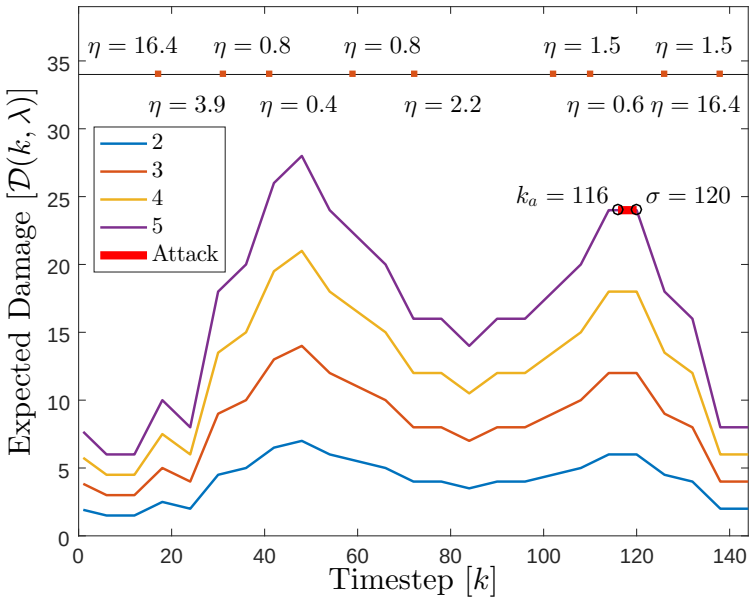


Figure 3.4: Best-response attack against the optimal time-dependent threshold has the magnitude $\lambda = 5$ and starts at $k_a = 116$.

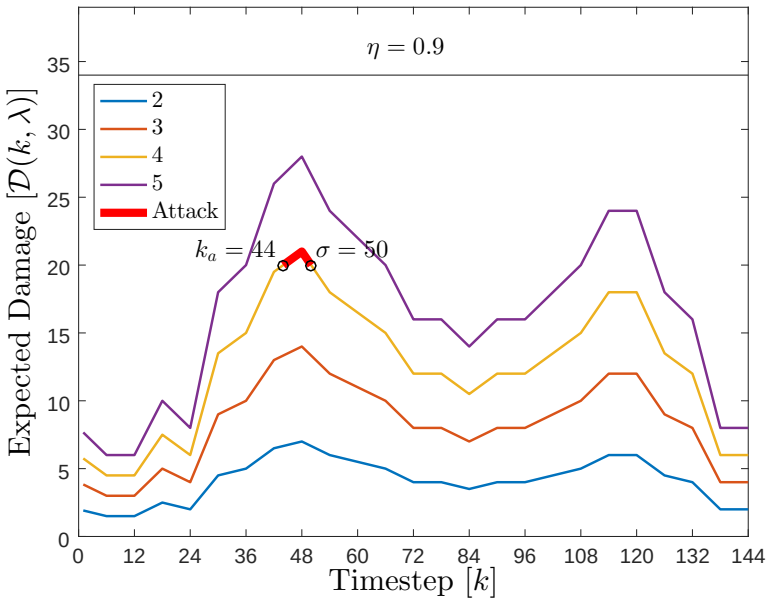


Figure 3.5: Best-response attack against the optimal fixed threshold has the magnitude $\lambda = 4$ and starts at $k_a = 44$.

Table 3.3: 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

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 the payoff attained by the attacker and the loss incurred by the defender. Table 3.3 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.

3.7.3.2 Sensitivity Analysis.

Figure 3.6 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 3.7 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 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.

3.7.3.3 Running Time.

We now compare the running time of Algorithm 2 with an algorithm that finds the optimal thresholds using an exhaustive search. Figure 3.8 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.

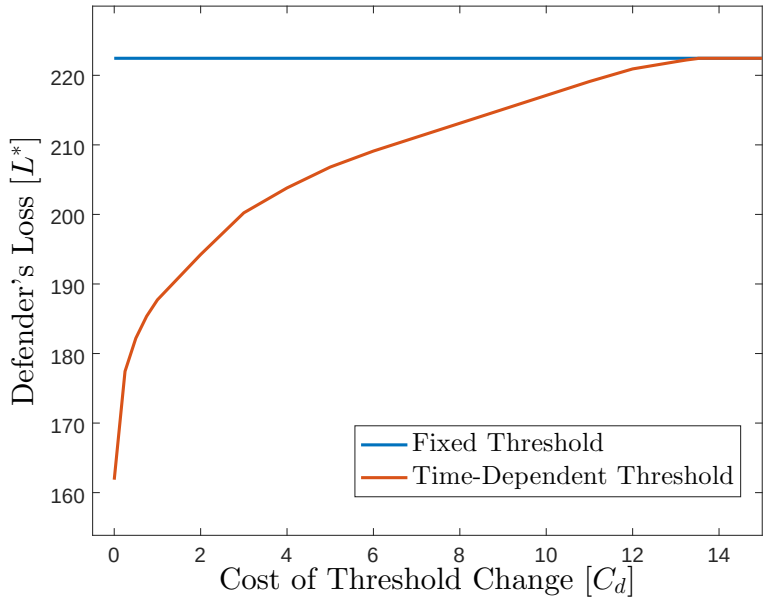


Figure 3.6: The defender's loss as a function of cost of threshold change.

3.7.4 Random Faults

Figure 3.9 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 (3.2)) and loss due to combination of either faults or attacks (i.e., Equation (3.10)). 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., (3.10), when thresholds η are selected. Similarly, we let $L_A^*(\eta)$ be the loss considering only attacks, i.e., Equation(3.2), when thresholds η are selected.

We observe that $\mathcal{L}_C^*(\eta_C^*)$ outperforms $\mathcal{L}_C^*(\eta_A^*)$ and $L_A(\eta_A^*)$ outperforms $L_A(\eta_C^*)$. 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.

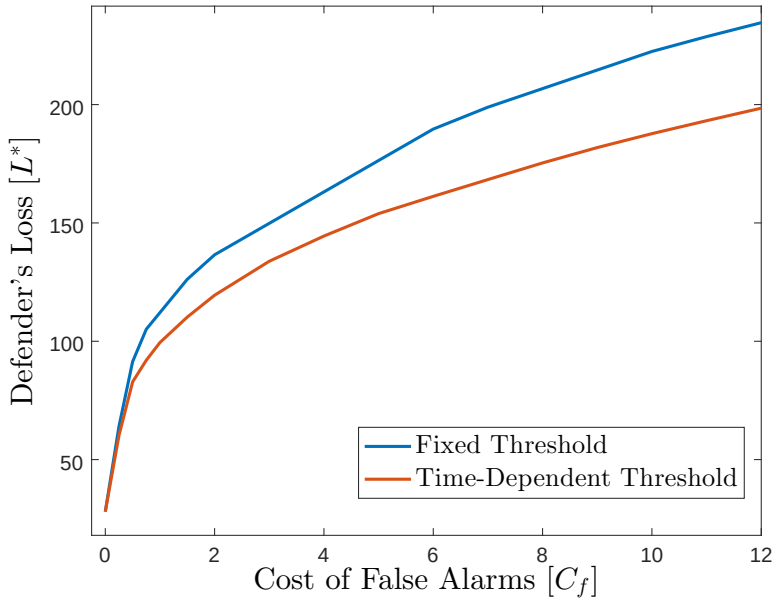


Figure 3.7: The defender's loss as a function of cost of false alarms.

3.8 Conclusion

In this chapter, 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 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.

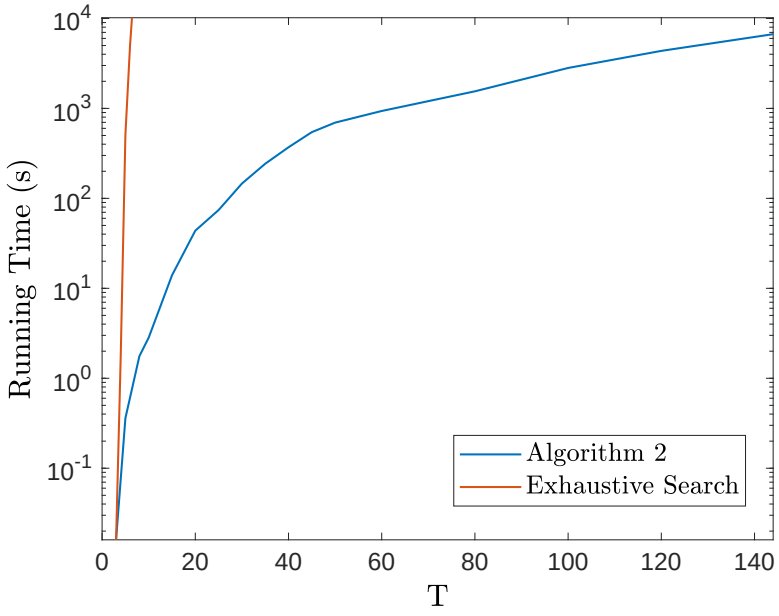


Figure 3.8: Running time of Algorithm 2 compared to exhaustive search.

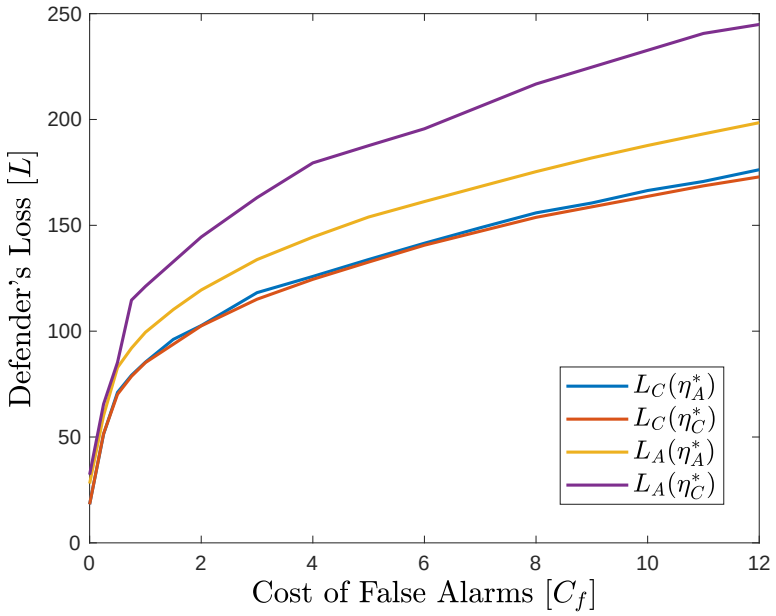


Figure 3.9: The defender's loss as a function of cost of false alarms for time-dependent thresholds. η_A^* is the optimal threshold for attacks and η_C^* is the optimal threshold for combination of faults and attacks.

Chapter 4

Optimal Detection of Faulty Traffic Sensors Used in Route Planning

In a smart city, real-time traffic sensors may be deployed for various applications, such as route planning. Unfortunately, sensors are prone to failures, which result in erroneous traffic data. Erroneous data can adversely affect applications such as route planning, and can cause increased travel time. To minimize the impact of sensor failures, we must detect them promptly and accurately. However, typical detection algorithms may lead to a large number of false positives (i.e., false alarms) and false negatives (i.e., missed detections), which can result in suboptimal route planning. In this chapter, we devise an effective detector for identifying faulty traffic sensors using a prediction model based on Gaussian Processes. Further, we present an approach for computing the optimal parameters of the detector which minimize losses due to false-positive and false-negative errors. We also characterize critical sensors, whose failure can have high impact on the route planning application. Finally, we implement our method and evaluate it numerically using a real-world dataset and the route planning platform OpenTripPlanner.

4.1 Introduction

In smart cities, real-time traffic sensors may be deployed for various applications. However, sensors are prone to failures, which result in erroneous traffic data. Erroneous data can adversely affect the performance of applications. To minimize the impact of sensor failures, we must detect them promptly and with high accuracy. However, typical detection algorithms may lead to a large number of false positives and false negatives, which can result in suboptimal performance.

Anomaly detection of faulty traffic sensors has been studied in the literature. Typical approaches include using data-driven methods that incorporate historical and real-time data to detect anomalies [87], [146], [117], [139]. However, existing approaches may result in high performance-losses in traffic applications, mainly due to false-positive (FP) and false-negative (FN) errors. In order to minimize the losses, it is desirable to reduce the FP and FN rates as much as possible. But, there exists a trade-off between them, which can be changed through a detection threshold. To address this, it is necessary to take into account the traffic application when designing anomaly detectors, and quantify the losses in the traffic application caused by the FP and FN errors. By selecting the right

detection threshold, the performance losses caused by FPs and FNs can be minimized.

In this chapter, we study the problem of finding optimal thresholds for anomaly detection of faulty traffic sensors, considering route planning as the application of interest. The objective is to select the optimal thresholds of anomaly detectors in order to optimize the performance of the route planning application in the presence of faulty sensors. We devise an effective detector for identifying faulty traffic sensors using a prediction model based on Gaussian Processes. Further, we present an approach for computing the optimal parameters of the detector which minimize losses due to false-positive and false-negative errors. We also characterize critical sensors, whose failure can have high impact on the traffic application. Finally, we implement our method and evaluate it numerically using a real-world dataset and the route planning platform OpenTripPlanner [93]. Our evaluation results show that the proposed strategy successfully minimizes the performance loss and identifies the critical sensors.

The remainder of this chapter is organized as follows. In Section 2, we discuss related work. In Section 3, we present the background for route planning and Gaussian Process regression. In Section 4, we introduce the system model. In Section 5, we define a notion of optimal detection, present a method to obtain near-optimal thresholds, and define critical sensors. In Section 6, we implement our method and evaluate it numerically. Concluding remarks are presented in Section 7.

4.2 Related Work

There are many papers that study traffic prediction. The work in [80] uses multivariate kernel regression models to predict traffic flow in a network, considering route planning as the application. In [39], the paper provides a travel time prediction algorithm in a small scale simulated network. The work in [126] constructs robust algorithms for short-term traffic flow prediction. Finally, in [62], classical time series approaches are used for short-term speed prediction in a network.

The problem of anomaly detection of traffic sensors is reviewed in [87]. The paper categorizes different methods into the three levels of macroscopic, mesoscopic, and microscopic, and provides practical guidelines for anomaly detection. The work in [146] presents three methods to detect faulty traffic measurements. The methods are based on Pearson's correlation, cross-correlation, and multivariate ARIMA. Finally, the work in [117] presents a test, which is based on the relationship between flows at adjacent sensors to detect faulty loop detectors. Nevertheless, since previous papers use static thresholds, their methods result in high losses due to FPs and FNs.

The problem of optimal parameter selection for anomaly detection is studied in [74]. The paper

shows that computing optimal attacks and defenses is computationally expensive, and proposes heuristic algorithms for computing near-optimal strategies. More details about this work can be found in Chapter 2. Also, we discussed the problem of finding optimal thresholds for anomaly-based detectors implemented in dynamical systems in the face of strategic attacks in the previous chapter.

4.3 Background

4.3.1 Route Planning

Let $G = (V, E)$ be a directed graph with a set V of vertices and a set E of arcs. Each arc $(u, v) \in E$ has an associated nonnegative cost $c(u, v)$. The cost (i.e., length) of a path is the sum of the costs of its arcs. In the point-to-point shortest path problem, one is given as input the graph G, a query $q = (o, d)$, where $o \in V$ is an origin and $d \in V$ is a destination, and the objective is to find a minimum-cost (i.e., shortest) path from o to d in G. In the many-to-many shortest path problem, a set of queries Q is given, and the goal is to find the minimum-cost path for each query $q = (o, d) \in Q$.

There exist many route planning algorithms that compute optimal solutions in an efficient manner [12]. Among these methods, the bidirectional Dijkstra's algorithm with binary heaps computes point-to-point shortest path in $\mathcal{O}(|E| + |V| \log |V|)$. Further, the Floyd-Warshall algorithm solves all pairs shortest paths in $\mathcal{O}(|V|^3)$. A large number of methods have been designed to improve running time of shortest-path algorithms. For example, contraction hierarchies and arc flags have been successfully used [36].

4.3.2 Gaussian Process Regression

GPs provide a Bayesian paradigm to learn an implicit functional relationship $y = f(\boldsymbol{x})$ from a training dataset $\{(\boldsymbol{x}_i, y_i); i = 1, 2, ..., n\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ represents the vector of observed input variables (i.e., predictors), and y_i is the observed target value. A comprehensive discussion of GPs in machine learning can be found in [115].

GPs directly elicit a prior distribution on the function $f(\boldsymbol{x})$, and assume it to be a GP a priori,

$$f(\boldsymbol{x}) \sim GP\left(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x'})\right). \tag{4.1}$$

For a new point $\boldsymbol{x}_*$, the goal is to predict $y_* = f(\boldsymbol{x}_*)$. Given that the regression function is a GP, the distribution of the values of f at any finite number of points is a multivariate Gaussian distribution.

Therefore,

$$\begin{pmatrix} \boldsymbol{y} \\ y_* \end{pmatrix} \sim \mathcal{N}\left(\mu(x), \begin{pmatrix} K & K'_* \\ K_* & K_{**} \end{pmatrix}\right), \tag{4.2}$$

where K is the covariance matrix for the labeled points, K_* is the covariance vector between the new point and the labeled points, and K_{**} is the measurement noise. Then,

$$\Pr(y_* \mid \boldsymbol{y}) \sim \mathcal{N}\left(K_* K^{-1}\boldsymbol{y}, K_{**} - K_* K^{-1} K'_*\right). \tag{4.3}$$

The prediction of a GP model depends on the choice of covariance function, which identifies the expected correlation between the observed data. Typically, a parametric family of functions is used, and the hyperparameters are inferred from the data. Examples of the commonly used covariance functions include polynomial kernel, automatic relevance determination (ARD), and radial basis function (RBF). Methods for learning the hyperparameters are based on maximization of the marginal likelihood, which can be performed using gradient-based optimization algorithms.

4.4 System Model

In this section, we present the system model. We first define a model of transportation network. Then, we construct a detector for identifying faulty traffic sensors using a prediction model based on Gaussian Processes.

4.4.1 Transportation Network

Consider a transportation network modeled as a graph $G = (V, E)$, where edges represent road segments and vertices represent connections between road segments (e.g., traffic junctions). We assume that a subset $S \subseteq E$ of the road segments are monitored by sensors that measure traffic state (e.g., speed, occupancy, flow) at discrete timesteps $k \in \mathbb{N}$. The measurements of these sensors are transmitted to a navigation service, which given a set of queries $Q(k)$ at timestep k, computes the corresponding shortest paths. For segments without a traffic sensor, we assume the navigation service uses either previously computed values or predicted values using measurements of adjacent sensors.

Traffic sensors may be faulty due to miscalibration or hardware failure. If a sensor $s \in S$ is faulty, there is a discrepancy between the actual and measured values. In other words, if $a_s(k)$ is the actual value and $m_s(k)$ is the measured value of faulty sensor s, then $m_s(k) = a_s(k) + \varepsilon_s(k)$,

where $\varepsilon_s(k)$ is the fault value at time k. In this model, we do not consider faults that result in no data being sent, since such cases can easily be filtered out by an operator.

4.4.2 Gaussian Process-Based Detector

Given the sensor measurements, we need to decide whether some sensors are faulty. We assume that the number of sensors that simultaneously become faulty is low, which is true in practice. As a result, for any sensor, the majority of nearby sensors that have not been marked faulty provide reliable traffic data, and so we can use these nearby sensors to predict the value measured by the sensor in question. To detect faults, we then compare the predictions to the measurements, and if there is a significant difference between the predicted values and the received measurements, an alarm indicating presence of a fault in that particular sensor is triggered.

4.4.2.1 Traffic Prediction

As our traffic predictor, we use GPs, which is a kernel-based machine learning method. Kernel-based methods have gained special attention for traffic prediction because of their generalization capability and superior nonlinear approximation. Among different kernel-based methods, previous work shows that GPs outperform other methods such as ARIMA and neural networks [141]. We use GPs because in addition to the above advantages, it allows for explicit probabilistic interpretation of forecasting outputs.

As the kernel function, we decide for the commonly used ARD squared exponential,

$$K(\boldsymbol{m}(k), \boldsymbol{m}(k)') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(m_i(k) - m_i'(k))^2}{\sigma_i^2}\right), \tag{4.4}$$

where $\boldsymbol{m}(k)$ and $\boldsymbol{m}(k)'$ are vectors of measurements, and σ_f and $\{\sigma_i\}_{i=1}^{d}$ are hyperparameters.

We let the target variable be the predicted traffic value p_s (e.g., traffic flow or occupancy) of sensor $s \in S$ at timestep k. Further, we let the predictor variables be the measured traffic values of other sensors at the same timestep. In practice, two sensors are highly correlated if they are in close proximity. Therefore, it is possible to select predictor variables as the measured values of d closest sensors from the target sensor, where the choice of d depends on the network structure. This way, the predicted traffic value is defined as $p_s(k) = f(m_{V(s)}(k))$, where $V(s)$ is the set of d closest sensors from s.

4.4.2.2 Detection Algorithm

We can efficiently detect failures for each sensor $s \in S$, by comparing the measured traffic value $m_s(k)$ with the predicted traffic value $p_s(k)$. We use Cumulative sum control chart (CUSUM) as the detection algorithm, which is a sequential analysis technique typically used for monitoring change detection [105].

Consider sensor $s \in S$, with a sequence of measurements $m_s(1), ..., m_s(k)$ and corresponding traffic predictions with means $p_s(1), ..., p_s(k)$ and standard deviations $\sigma_s(1), ..., \sigma_s(k)$. The standardized residual signal is defined as

$$z_s(k) = \frac{m_s(k) - p_s(k)}{\sigma_s(k)}. \tag{4.5}$$

Moreover, upper and lower cumulative sums are defined as,

$$U_s(k) = \max(0, U_s(k-1) + z_s(k) - b_s), \tag{4.6}$$

$$L_s(k) = \min(0, L_s(k-1) + z_s(k) + b_s), \tag{4.7}$$

where $U_s(k) = L_s(k) = 0$ for $k = 1$, and b_s is a small constant.

Denoting the detection threshold at timestep k by $\eta_s(k)$, a measurement sequence violates the CUSUM criterion at the sample $z_s(k)$ if it obeys $U_s(k) > \eta_s(k)$ or $L_s(k) < -\eta_s(k)$. Formally, letting H_0 and H_1 be the null and fault hypothesis, the decision rule is described by

$$d_s(U_s(k), L_s(k)) = \begin{cases} H_1 & \text{if } U_s(k) > \eta_s(k) \text{ or } L_s(k) < -\eta_s(k) \\ H_0 & \text{otherwise} \end{cases}. \tag{4.8}$$

4.4.2.3 False-Negative and False-Positive Trade-off

In anomaly detectors, there might be a *false negative*, which means failing to raise an alarm when a fault did happen. Further, there might be a *false positive*, which means raising an alarm when the sensor exhibits normal behavior. It is desirable to reduce the FP and FN probabilities as much as possible. But, there exists a trade-off between them, which can be controlled by changing the threshold. In particular, by decreasing (increasing) the threshold, one can decrease (increase) the FN probability and increase (decrease) the FP probability.

We represent the FN probability for each sensor s by the function $FN_s : \mathbb{R}_+ \rightarrow [0, 1]$, where

$FN_s(\eta_s(k))$ is the probability of FN when the threshold is $\eta_s(k)$, given that the sensor is faulty. Similarly, we denote the attainable FP probability for each sensor s by $FP_s : \mathbb{R}_+ \to [0,1]$, where $FP_s(\eta_s(k))$ is the FP probability when the threshold is $\eta_s(k)$, given that the sensor is in normal operation. It is possible to plot the FP probability as a function of the FN probability for various threshold values [43] (e.g., see Figure 4.3).

4.5 Optimal Detection

In this section, we formulate the problem of finding optimal thresholds for anomaly detection of traffic sensors, considering route planning as their primary application. The objective is to select the optimal thresholds for anomaly detectors in order to minimize the losses caused by false positives and false negatives. Then, we present an algorithm to find near-optimal detection thresholds. Finally, we characterize critical sensors, whose failure can have high impact on the traffic application.

4.5.1 Optimization Problem

First, consider the set of queries Q, and a route planning algorithm that takes as inputs the set of queries and the measured and predicted traffic values, and outputs the optimal routes. For a single query $q \in Q$ and sensor $s \in S$, we denote by $P_q(m_s)$ the optimal route computed using the measured traffic values for all sensors, and we denote by $P_q(p_s)$ the optimal route using the predicted value p_s for sensor s and the measured values $\boldsymbol{m}_{-s}$ for all other sensors. Finally, for a given route r and sensor s, let $T(r, m_s)$ and $T(r, p_s)$ be the total travel time based on the measured m_s and predicted p_s values for sensor s, respectively, and the measured values $\boldsymbol{m}_{-s}$ for all other sensors.

Then, $T(P_q(p_s), m_s)$ is the measured travel time of the shortest route computed using the predicted value p_s for sensor s. Similarly, $T(P_q(m_s), m_s)$ is the measured travel time of the shortest route computed using the measured value m_s. We define the loss caused by a false positive as follows:

$$C^{FP}_{s,q}(p_s, m_s) = T(P_q(p_s), m_s) - T(P_q(m_s), m_s), \tag{4.9}$$

that is, the difference in measured travel time between using either the predicted or the measured value for sensor s.

The rationale behind the above expression is the following. In case of a FP, according to the detector, the measured value m_s is incorrect, but it is actually correct. Consequently, we choose a route that is computed using our prediction p_s instead of the optimal route, which would be computed using the measurement m_s. To quantify the loss, we need to compare the travel times of

the two routes, and we must use the measured traffic value m_s for this comparison since that is the correct value in this case.

Similarly, for a FN, $T\left(P_q\left(m_s\right), p_s\right)$ is the predicted travel time of the shortest route using measured value m_s, and $T\left(P_q\left(p_s\right), p_s\right)$ is the predicted travel time of the shortest path using predicted value p_s. The loss caused by a FN is

$$C_{s,q}^{FN}(p_s, m_s) = T\left(P_q\left(m_s\right), p_s\right) - T\left(P_q\left(p_s\right), p_s\right), \tag{4.10}$$

that is, the difference in predicted travel time between using either the measured or the predicted value for sensor s. Note that in (4.9) and (4.10), the values of P and T can be computed using existing route planning algorithms [12].

Next, let $FP_s(\eta_s(k))$ and $FN_s(\eta_s(k))$ be the probabilities of false-positive and false-negative errors when detection threshold $\eta_s(k)$ is selected. Further, let p_f be the probability of fault, and let $p_n = 1 - p_f$ be the probability of normal operation. For a given query q, the total loss caused by FPs and FNs is,

$$L_{s,q}(\eta_s(k)) = FP_s(\eta_s(k)) \cdot C_{s,q}^{FP}(p_s, m_s) \cdot p_n + FN_s(\eta_s(k)) \cdot C_{s,q}^{FN}(p_s, m_s) \cdot p_f \ . \tag{4.11}$$

Considering the set of all queries Q, the total loss is

$$L_s(\eta_s(k), Q) = \sum_{q \in Q} L_{s,q}(\eta_s(k)), \tag{4.12}$$

which allows us to define the notion of optimal detection threshold for a sensor.

Definition 1 (Optimal Detection)**.** *The detection threshold $\eta_s^*(k)$ is optimal for sensor s if it minimizes the loss function* (4.12)*. Formally, $\eta_s^*(k)$ is optimal for sensor s if*

$$\eta_s^*(k) \in \underset{\eta_s(k)}{\operatorname{argmin}} L_s(\eta_s(k), Q). \tag{4.13}$$

Figure 4.1 shows the flow of information in our approach. At each timestep k, given measurements $\boldsymbol{m}(k)$, the predictor computes the predicted measurements $\boldsymbol{p}(k)$. Then, given a set of queries $Q(k)$, and the predictions and measurements, the thresholds $\boldsymbol{\eta}(k)$ are computed for the detectors using the algorithm presented next.

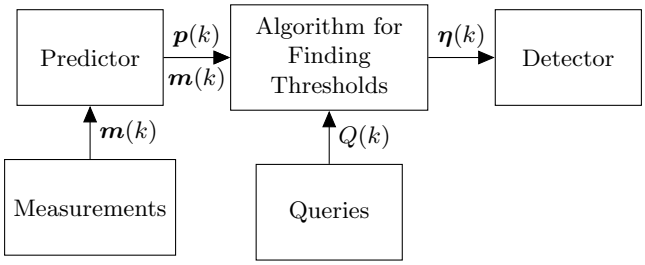


Figure 4.1: Information flow in our approach.

4.5.2 Algorithm for Obtaining Thresholds

We present Algorithm 4.1 to find near-optimal detection thresholds. The algorithm implements a random-restart hill climbing technique. If the FP to FN trade-off curve is convex, which makes (4.12) convex, we are able to compute optimal thresholds using convex optimization methods. However, this is not generally the case, as trade-off curves tend to be non-convex (see Figure 4.3 for an instance of a trade-off curve).

The algorithm considers each sensor separately, and finds its corresponding detection threshold. At each iteration, the algorithm selects a new starting point and finds a local minimum using gradient-based optimization. In order to avoid unnecessary computation, we skip computing detection thresholds for sensors with very similar measured and predicted traffic values. Formally, for sensor $s \in E$, we select detection threshold $\eta_s = \infty$, if $|z_s(k)| < b$. This is because the detector's statistics $U_s(k)$ and $L_s(k)$ are decreasing and it is unlikely that an alert would be raised if one was not raised before.

4.5.3 Critical Sensors

Value of the optimal loss gives insight on the criticality of traffic sensors. Fault on a sensor that has high loss value degrades the system's performance more than fault on a sensor with low loss value. We formally define the set of δ-critical sensors below.

Definition 2 (Critical Sensors)**.** *Set of δ-critical sensors in a time period $[1, T]$ is defined as the set of sensors which have the average optimal loss values of greater than or equal to δ. That is to say, a sensor s is critical if $\frac{1}{T}\sum_{k=1}^{T} L_s(\eta_s^*(k), Q(k)) \geq \delta$.*

Identifying critical sensors is beneficial, since it allows us to locate the most vulnerable elements of a network, which should be strengthened first to increase the robustness of a network. For example, if we have a limited budget which permits us to replace only a subset of the sensors with more robust

Algorithm 4.1 Algorithm for Obtaining Thresholds

Input Q, $\boldsymbol{FP}(\eta)$, $\boldsymbol{FN}(\eta)$, α, γ
Initialize: $\boldsymbol{\eta} \leftarrow \boldsymbol{\eta}_0$, $\boldsymbol{L}^* \leftarrow \infty$
for all $s \in S$ **do**
    if $|z(k)| \leq b$ **then**
        $\eta_s^* \leftarrow \infty$
    else
        while $i < N$ **do**
            $\eta_{s,new} \hookleftarrow FP_s^{-1}(\text{Uniform}([0,1]))$
            $\eta_{s,old} \leftarrow 0$
            while $|L_s(\eta_{s,new}, Q) - L_s(\eta_{s,old}, Q)| > \alpha$ **do**
                $\eta_{s,old} \leftarrow \eta_{s,new}$
                $\eta_{s,new} \leftarrow \eta_{s,old} - \gamma \nabla_{\eta_s} L_s(\eta_{s,old}, Q)$
            end while
            if $L_s(Q, \eta_{s,new}) < L_s^*$ **then**
                $\eta_s^* \leftarrow \eta_{s,new}$
                $L_s^* \leftarrow L_s(\eta_{s,new}, Q)$
            end if
            $i \leftarrow i + 1$
        end while
    end if
end for
return $\boldsymbol{\eta}^*$

ones, then we should start with the critical sensors.

4.6 Evaluation

In this section, we implement our method and evaluate it numerically using a route planning platform.

4.6.1 System Model

4.6.1.1 Traffic Data

We use a traffic dataset obtained from the Caltrans Performance Measurement System (PeMS) database [24]. The database provides real-time and historical traffic data from over 39,000 individual sensors, which span the freeway system across metropolitan areas of the State of California. Figure 5.5 shows the location of sensors in our case study, in which a total of 40 sensors are considered. We use the 5-minute aggregated data collected on the weekdays of September 3, 2016 to September 17, 2016. The dataset contains 115,200 data points. The first 7 days are used as training data, and the remaining 7 days are used as test data.

To simulate faults, we use models for a specific set of fault types and ranges of fault magnitudes,

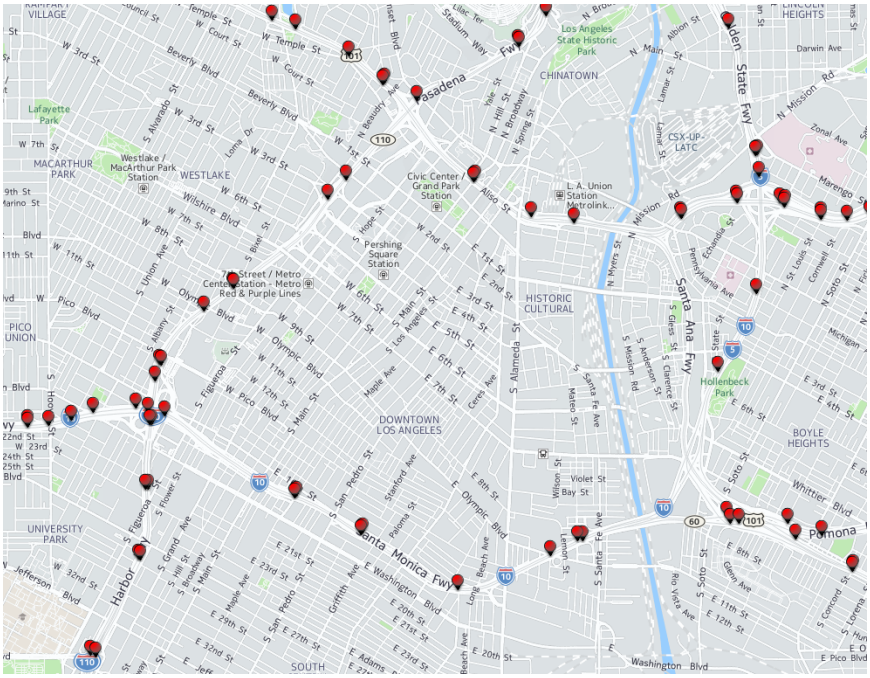


Figure 4.2: A map of traffic sensors installed in Downtown Los Angeles.

which is similar to the approach presented in [139]. The fault models are: 1) Constant Relative Overcount (caused by e.g., unsuitable sensitivity levels); range: 3% to 7% of the actual values (i.e., $\varepsilon_s(k) = u_s a_s(k)$ where $0.03 \leq u_s \leq 0.07$), 2) Conditional Undercount (caused by e.g., sensor saturation); range: 7% to 13% (i.e., $\varepsilon_s(k) = u_s a_s(k)$ where $-0.13 \leq u_s \leq -0.07$).

Next, for each sensor, we construct a predictor using the measurements of its d closest sensors as the predictor variables. We select $d = 10$ since it results in the minimum overall prediction error. We choose $b_s = 0.05$ for all the detectors, to make them sensitive to small shifts in the mean. We evaluate each detector's performance by plotting the FP probability against the FN probability at various threshold values. Figure 4.3 shows the trade-off curve of the detector implemented for a sensor, whose identifier in the PeMS dataset is VDS 774685.

4.6.1.2 Route Planner

We use OpenTripPlanner (OTP), which is an open source platform for multi-modal route planning [93]. OTP relies on open data standards including OpenStreetMap for street networks. The default routing algorithm in OTP is the A^* algorithm with a cost-heuristic to prune the search. For improved performance on large networks, it also uses contraction hierarchies.

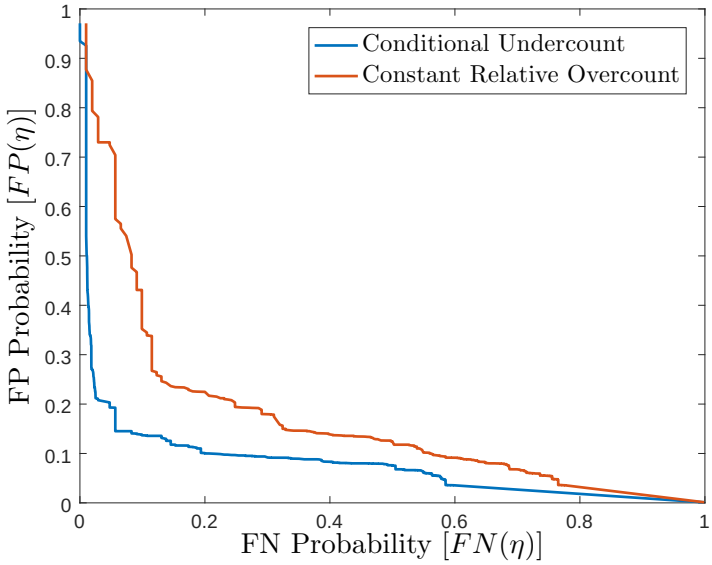
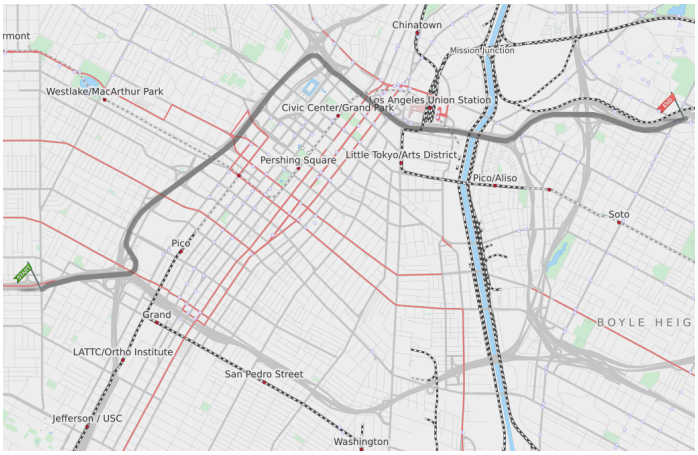


Figure 4.3: Trade-off between the false-positive and false-negative probabilities.

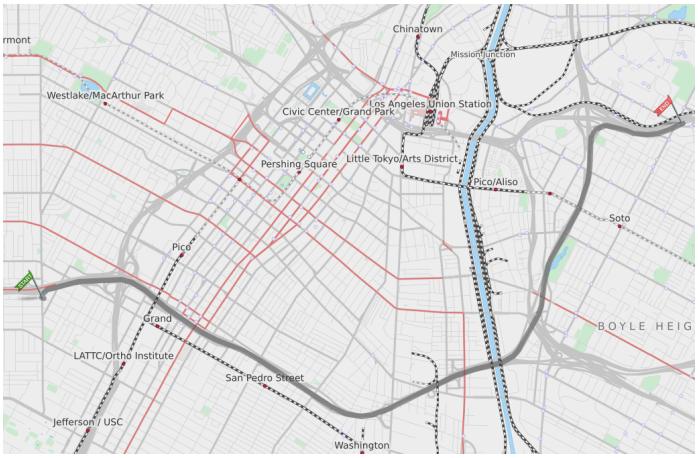
4.6.2 Results

We simulate a route planning scenario in OTP, where the edge costs (i.e., travel times) are updated using our traffic data. For a source and destination as shown in Figure 4.4a, we consider 1000 queries made on September 15, from 9:00 am to 10:00 am. Figure 4.4a shows the shortest route when a particular sensor (i.e., VDS 774685) is healthy, and Figure 4.4b shows the shortest route when the same sensor has a conditional undercount fault. Note that if the fault remains undetected (i.e., false negative), a suboptimal route (Figure 4.4b) will be selected instead of the optimal route (Figure 4.4a). In another scenario, assume an alarm is triggered under normal operation (i.e., false positive). This means that the predicted value is used for route planning instead of the accurate measurement value, which depending on the prediction accuracy, may result in a suboptimal route planning solution.

We use Algorithm 4.1 to find optimal thresholds that minimize losses due to FPs and FNs. We assume that for each sensor, the probability of fault is $p_f = 0.05$. For the previously considered sensor, at $k = 1$ (i.e., from 9:00 am to 9:05 am), the loss value (4.12) as a function of the threshold is shown in Figure 4.5. In this case, Algorithm 4.1 finds the optimal thresholds. For the Conditional Undercount, the optimal threshold and the minimum loss are $\eta = 0.17$ and $L = 16.2$, whereas for the Constant Relative Overcount, the optimal threshold and the minimum loss are $\eta = 0.39$ and $L = 30.0$.



(a)



(b)

Figure 4.4: Reroute occurs due to a conditional undercount fault false negative. (a) Normal. (b) Fault. (Green flag is the source and red flag is the destination.)

Further, Table 4.1 shows the average optimal loss for some sensors, i.e., $\frac{1}{T}\sum_{k=1}^{T} L_s(\eta_s^*(k), Q(k))$. As a baseline, we also compute the minimum loss when the thresholds have static values at all the timesteps. That is, for all k, we assign $\eta_s(k) = \eta_s^*$, where $\eta_s^* \in \text{argmin}_{\eta_s} \sum_k L_s(\eta_s, Q)$. We observe that our method achieves significantly smaller losses compared the static case. The loss values can also be used to identify the set of δ-critical sensors. For example, 50.0-critical sensors are made bold in the table.

4.7 Conclusions

We studied the problem of finding optimal detection parameters for anomaly detection of traffic sensors, considering route planning as application. We constructed a predictor using Gaussian pro-

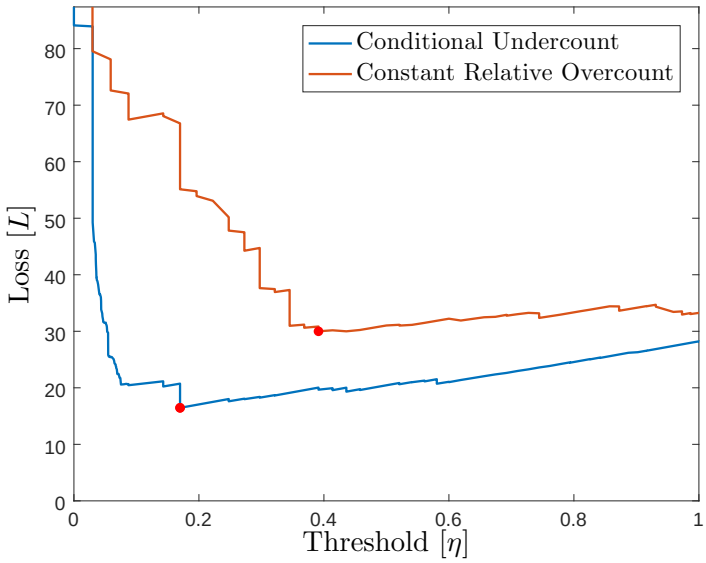


Figure 4.5: Loss as a function of detection threshold.

Table 4.1: Average Optimal Losses

Sensor ID	Cond. Undercount		Cons. Rel. Overcount	
	Optimal	Static	Optimal	Static
774685	16.2	31.2	30.0	38.1
774672	18.0	27.6	22.1	36.7
772501	15.6	24.3	12.8	19.2
763453	**51.8**	74.3	**57.5**	80.9
737158	43.0	59.6	**54.8**	71.4

cesses, which was then used for anomaly detection. We studied how to find the optimal detection parameters, which minimize losses due to FP and FN errors. We also characterized critical sensors, whose failure can have high impact on the traffic application. We implemented our method and evaluated it numerically using a route-planning platform. Our evaluations indicated that the proposed detection method successfully minimizes the performance losses.

Chapter 5

Application-Aware Anomaly Detection of Sensor Measurements in Cyber-Physical Systems

Detection errors, i.e., false alarms and missed detections, are inevitable in anomaly detection systems. Such errors can cause highly degraded performance in CPS applications, as false alarms result in recovery that is not needed, and missed detections result in failing to perform recovery. In this chapter, we present a framework for *application-aware* anomaly detection, that is, an anomaly detector that configures itself such that the application performance in the presence of detection errors is as close as possible to the performance that could have been obtained if there were no detection errors. We evaluate our result using a case study of real-time control of traffic signals, and show that our application-aware detector significantly outperforms several baseline detectors.

5.1 Introduction

Sensors deployed in CPS applications for monitoring and control purposes are prone to anomalies (e.g., failures and cyber-attacks). To detect anomalies and prevent their harmful effects, anomaly detection systems (ADS) are implemented. However, ADS suffer from false positives (i.e., false alarms) and false negatives (i.e., missed detections), which may result in high performance degradation in CPS applications. In particular, false positives result in recovery that is not required, and false negatives result in failing to perform recovery when it is indeed required. Such detection errors can cause incorrect measurements being transmitted to the controller, and thus result in obtaining non-optimal or even destabilizing control decisions, which may compromise the performance of the system. For example, detection errors may result in disastrous events such as reactor explosion in process control systems, water contamination in water distribution networks, and extremely heavy traffic congestion in intelligent transportation systems [132, 76].

To address this, it is necessary to take into account the CPS application when designing anomaly detectors, and to quantify the losses in the application caused by potential detection errors. In order to minimize the losses, while it is desirable to reduce the detection errors as much as possible, there exists a trade-off between them, which can be changed through a detection threshold. Therefore, by selecting the right detection threshold, the performance losses caused by detection errors can be minimized.

Our goal is to perform these steps using a novel approach which takes into account the behavior of the controller in the configuration of the anomaly detector. We call the framework *Application-Aware Detection*. In such framework, the detector is aware of the interactions between the controller and the application, and so it can compute how each detection decision can affect the underlying application. Knowing this, the detector attempts to make detection decisions that will result in the least performance loss in the underlying application if the detection decision is not accurate due to false positives and false negatives.

Previous works have proposed different anomaly detection methods for CPS [132]. In addition, there is a wide body of literature on machine learning-based anomaly detection [29]. However, there is little work that takes into account the tight interaction between the detector and the controller of a CPS, which as we show in this work, if taken into account, can result in improved performance and robustness. To the best of our knowledge, this is the first time that such approach is used for improved detection performance in CPS.

In this chapter, we propose the application-aware anomaly detection framework for detecting anomalies in sensors measurements in CPS. First, we devise an effective detector for identifying anomalies in sensor measurements using machine learning regression. Second, we propose an approach to recover from anomalies in order to maintain operation when detection alerts are triggered. Then, we formulate the problem of application-aware detection, in which the anomaly detector is optimally configured such that the performance loss in the presence of detection errors is minimized. In particular, the thresholds are selected such that the performance of the system in the presence of detection errors is as close as possible to the performance that could have been obtained if there were no detection errors. We show that the application-aware detection problem is computationally challenging, and then we present an efficient algorithm to find near-optimal solutions. We also study two special variations of the application-aware detection problem, that is, single detector and detectors with equal threshold. We optimally solve both special cases, which aside from practical advantages, can provide insights into the novelty of the approach. We then perform simulation experiments on a case study of real-time control of traffic signals. We evaluate our approach numerically and show its benefits in comparison to standard anomaly detection practices. Finally, we offer concluding remarks and discuss the advantages and disadvantages of the application-aware detector, and how it can become suitable for real-world deployment.

We believe this framework can be useful in systems where there is a significant number of sensors with high variations in sensor values, which can potentially cause many false positives and false negatives. A real-world example of such CPS application would be real-time control of traffic signals,

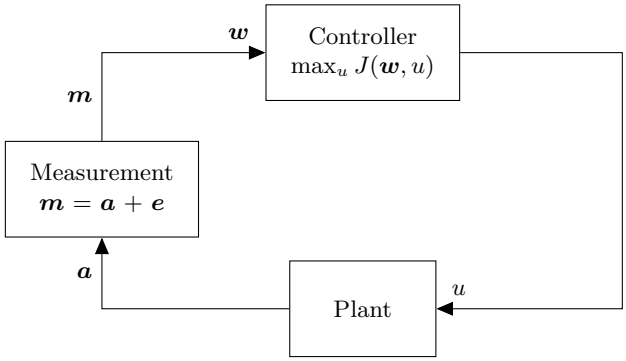


Figure 5.1: System Model. Note that in this case $\boldsymbol{w} = \boldsymbol{m}$.

as in large cities, there are thousands of sensors that could become anomalous.

The rest of this chapter is organized as follows. In Section 5.2, we introduce the system model and present the problems that are studied in this paper. In Section 5.3, we discuss the regression-based anomaly detection framework. In Section 5.4, we present the application-aware detection problem for detection error-tolerant selection of thresholds in anomaly detectors. In Section 5.5, we analyze the application-aware detection problem and present an algorithm to obtain near-optimal solutions. In Section 5.6, we study two special variations of the application-aware detection problem, that is, single detector and detectors with equal threshold. In Section 5.7, we evaluate our approach numerically using a case study of real-time control of traffic signals. Finally, we offer concluding remarks in Section 5.8.

5.2 Problem Statement

In this section, we present the system model. We also present a running example of real-time control of traffic signals that is used throughout the chapter to demonstrate the approach.

Notation

Vectors are denoted by bold symbols. Vector $\boldsymbol{y}$ at timestep k is described by $\boldsymbol{y}^k$. We omit the timestep symbol when all symbols have same timestep k. However, timestep symbol is used when there are different timesteps present or when it eases understanding. Given vector $\boldsymbol{y}$ and set of indices I, vector $\boldsymbol{y}_I$ is defined as a vector with same size as $\boldsymbol{y}$ that has the same size as $\boldsymbol{y}$ for indices in I, and is zero otherwise. Given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, the union operator computes the sum of them, that is, $\boldsymbol{x} \cup \boldsymbol{y} = \boldsymbol{x} + \boldsymbol{y}$. For a list of symbols used in this chapter, see Table 5.1.

Table 5.1: List of Symbols

Symbol	Description
S	Set of sensors
a_s	actual value for sensor s
m_s	Measured value for sensor s
p_s	Predicted value for sensor s
$TP_s(\tau)$	True positive probability of the detector for sensor s given detection threshold τ
$FP_s(\tau)$	False positive probability of the detector for sensor s given detection threshold τ
$TN_s(\tau)$	True negative probability of the detector for sensor s given detection threshold τ
$FN_s(\tau)$	False negative probability of the detector for sensor s given detection threshold τ
w_s	Recovered measurement transmitted to the controller for sensor s
r_s	Residual signal for sensor s

5.2.1 System Model

Consider a CPS, e.g., intelligent transportation system and process control system, that provides some service or utility. At each timestep, given transmitted measurements $\boldsymbol{w}$ containing information about the system, the controller computes a control input u that maximizes the utility function $J(\boldsymbol{w}, u)$[1]. In other words, the controller finds the optimal control input u^* defined as

$$u^* \in \underset{u}{\operatorname{argmax}}\, J(\boldsymbol{w}, u)\ , \tag{5.1}$$

where the optimal utility is denoted by $J^*(\boldsymbol{w})$.

Anomalous Sensors. Sensors may be anomalous due to hardware failures or sensor attacks. If sensor $s \in S$ is anomalous, there is a discrepancy between the actual and observed measured values. In other words, if a_s is the actual value and m_s is the observed measurement at a timestep, for an anomalous sensor we have $m_s = a_s + e_s$, where $e_s \in \mathbb{R}$ is the error value at that timestep. Figure 5.1 illustrates this idea.

5.2.2 Anomaly Detection, Recovery, and Resilience

Anomalies may incur extensive damage to the system and degrade the performance significantly. Our first problem is to construct an anomaly detection method in order to detect anomalies in sensor measurements.

[1]For a minimization problem $\min_u J'(\boldsymbol{w}, u)$, we can simply use $J(\boldsymbol{w}, u) = -J'(\boldsymbol{w}, u)$.

Problem 1 (Anomaly Detection)**.** *Construct an anomaly detection method in order to detect anomalies in sensor measurements.*

Suppose we have constructed such anomaly detection method. Upon detection, the system must recover from anomalies and continue operation. Therefore, our second problem is to design a recovery method to accommodate this.

Problem 2 (Recovery)**.** *Design a recovery approach in order to continue operation in the presence of detection alerts.*

Suppose we have designed such recovery approach that computes the recovered vector of measurements in the presence of detection alerts. In anomaly detectors, there are detection errors, that is, false positives (i.e., false alarms) and false negatives (i.e., missed detections). If there are no detection errors, the recovered vector of measurements would be close to the actual values (of course assuming that the recovery approach works well). However, in the presence of detection errors, false positives result in recovery that is not required, and false negatives result in failing to perform recovery when it is indeed needed.

To see the effect of detection errors on the application, let $\boldsymbol{w}'$ denote the recovered measurement vector, which will result in the utility $J' = \max_u J(\boldsymbol{w}', u)$. However, if there were no detection error, we could have obtained the optimal utility $J^* = \max_u(\boldsymbol{a}, u)$. Our final problem, which is the problem of resilience and the main contribution of this chapter, is to optimally configure our anomaly detectors through selection of detection thresholds so that the actual obtained utility in the presence of detection errors (i.e., J') is as close as possible to the utility that would have been obtained if there were no detection errors (i.e., J^*).

Problem 3 (Resilience)**.** *Find detection thresholds such that the obtained utility in the presence of detection errors is as close as possible to the utility that would have been obtained if there were no detection errors.*

We call the detector that solves the above problem the *Application-Aware Detector*.

5.2.3 Example: Real-Time Control of Traffic Signals

We present a running example of real-time control of traffic signals that is used throughout the chapter. In what follows, we describe the widely-popular max-pressure controller for optimal control of traffic signals with minor modifications in assumptions. In the original max-pressure algorithm presented in [135], traffic state is represented using exogenous demands that are then routed through

the network using routing ratios. In this work, instead of using exogenous demands that are then transformed to internal demands through using routing ratios, we assume that the internal demands are directly provided. Note that this does not affect the max-pressure algorithm as the algorithm effectively uses internal demands in its computations.

Max-Pressure Controller. Consider a network of intersections I with road links L. Movement from a link $i \in L$ to a link $j \in L$ is denoted by a pair $(i, j) \in E$. Further, let each movement (i, j) have a queue associated with it, and at each timestep, let $x(i, j)$ represent the length of this queue. The length of the queue shows how many vehicles intend to travel from i to j. For each movement (i, j), the pressure is defined as

$$P(i, j) = x(i, j) - \sum_{p} x(j, p) \ ,$$

which is simply the number of cars in the queue minus the total number of cars in the downstream queues.

Each intersection n has a traffic signal with a set of admissible stages Φ_n. Each stage $u_n \in \Phi_n$ is a set of simultaneous movements that are permitted by the traffic signal. If u_n permits a movement (i, j), then $u_n(i, j) = 1$, otherwise $u_n(i, j) = 0$. Let $c(i, j)$ be the saturation flow of movement (i, j). Given a stage $u_n \in \Phi_n$, pressure-release (i.e., utility) for intersection n is defined as

$$J_n(u_n) = \sum_{i,j} c(i, j)\, P(i, j)\, u_n(i, j) \ .$$

Algorithm 5.1 presents the max-pressure (MP) controller in detail. At each intersection n, the MP controller selects the stage u_n that results in the maximum pressure-release. In other words, the MP controller computes

$$u_n^* \in \operatorname*{argmax}_{u_n \in \Phi_n} \sum_{i,j} c(i, j)\, P(i, j)\, u_n(i, j) \ . \tag{5.2}$$

Note that at each intersection, the MP control selects a stage that depends only on the queues adjacent to the intersection. It is shown that the MP controller maximizes network throughput [135].

The overall utility for traffic network can be calculated by adding individual utilities for the intersections. That is, $J(x, u) = \sum_n J_n(x, u_n)$ where $u = \{u_n\}_{n \in I}$. Note that using this representation, the MP optimization problem becomes the same as (5.1).

Algorithm 5.1 Max-Pressure Controller [135]

Input: $x(i,j)$ for all $(i,j) \in E$

1: **for all** $n \in I$ **do**
2: **for all** $(i,j) \in E$ **do**
3: $P(i,j) \leftarrow x(i,j) - \sum_p x(j,p)$
4: **end for**
5: $u_n^* \leftarrow \text{argmax}_{u_n \in \Phi_n} \sum_{i,j} c(i,j)\, P(i,j)\, u_n(i,j)$
6: **end for**
7: **return** $\{u_n^*\}_{n \in I}$

5.3 Anomaly Detection

In this section, we construct a regression-based anomaly detector for identifying anomalous sensor measurements. We then discuss detection errors and some metrics that are used to characterize them.

5.3.1 Regression-Based Anomaly Detector

To protect the system against anomalies, we must detect them quickly and accurately. Many different anomaly detection systems have been proposed in the literature. For a comprehensive review of anomaly detection methods, we refer the reader to [29] for machine-learning based detectors and [132] for detectors used in CPS. In this work, we use regression-based anomaly detectors because in addition to state-of-the-art detection performance, such detectors require no knowledge of the physical system, can take into account complex and nonlinear behaviors of the system, and are easy to implement and can be highly scalable.

Architecture. Figure 5.2 shows the architecture of regression-based anomaly detector. The detector consists of two main components: 1) Predictor and 2) Statistical Test. The predictor predicts the value of a sensor given some information about the system state (e.g., current value of other sensors, previous control inputs). Then, the statistical test compares the computed prediction to the observed measurement and decides whether the sensor is normal or anomalous. We describe each component in more detail considering our running example.

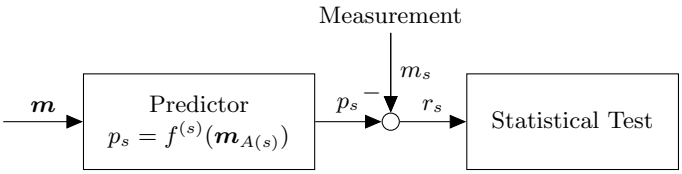


Figure 5.2: Regression-Based Anomaly Detector

5.3.1.1 Predictor

Our goal is to find a function $f^{(s)}$ that maps spatial or temporal features to the actual value of a sensor s (e.g., traffic flow or occupancy). In practice, two traffic sensors are highly correlated if they are in close proximity. Thus, we let the features be the measured values of other adjacent sensors at the same timestep, denoted by $\boldsymbol{m}_{A(s)}$ where $A(s)$ is a set of sensors adjacent to $A(s)$ found using cross-validation. Note that this approach is particularly applicable to traffic networks as there are usually many redundant sensors in the network. The function $f^{(s)}$ can then be obtained using suitable machine learning regression algorithm such as deep neural networks [89], Gaussian Processes [47], and many others [96]. Thus, for sensor s, we obtain the prediction as $p_s = f^{(s)}(\boldsymbol{m}_{A(s)})$.

5.3.1.2 Statistical Test

The statistical test efficiently detects anomalies for each sensor $s \in S$ by comparing the measured value $m_s(k)$ with the predicted value $p_s(k)$. Given a set of measured values $\boldsymbol{m} = \langle m_s \rangle_{s \in S}$ and predicted values $\boldsymbol{p} = \langle p_s \rangle_{s \in S}$, residual signals are computed as $\boldsymbol{r} = |\boldsymbol{m} - \boldsymbol{p}|$. Then, given the residuals, the statistical test makes detection decisions $\boldsymbol{d} = \langle d_s \rangle_{s \in S}$, where for each sensor s, the decision d_s is either normal or anomalous.

Different detection algorithms can be used to implement the statistical test [11]. In this work, we consider a stateless threshold-based detector defined as follows. Given detection thresholds $\boldsymbol{\tau} = \langle \tau_s \rangle_{s \in S}$, for each sensor s, if the residual r_s is less than or equal to the threshold τ_s, then s is marked normal and otherwise, s is marked anomalous. Thus

$$d_s = \begin{cases} \text{normal } (s \in N) & \text{if } r_s \leq \tau_s \\ \text{anomalous } (s \in A) & \text{otherwise} \end{cases}. \tag{5.3}$$

5.3.2 Detection Error

In anomaly detectors, there might be a *false negative*, which means failing to raise an alarm when an anomaly did happen. Further, there might be a *false positive*, which means raising an alarm when the system exhibits normal behavior. It is desirable to reduce the false positive and false negative probabilities as much as possible. But, there exists a trade-off between them, which can be controlled by changing the detection threshold. In particular, by decreasing (increasing) the threshold, one can decrease (increase) the FN probability and increase (decrease) the FP probability.

We represent the FN probability for each sensor s by the function $FN_s : \mathbb{R}_+ \rightarrow [0,1]$, where $FN_s(\tau_s)$ is the probability of FN when the threshold is τ_s, given that the sensor is anomalous. Similarly, we denote the attainable FP probability for each sensor s by $FP_s : \mathbb{R}_+ \rightarrow [0,1]$, where $FP_s(\tau_s)$ is the FP probability when the threshold is τ_s, given that the sensor is in normal operation. The true positive and true negative probabilities are also denoted by $TP_s(\tau_s)$ and $TN_s(\tau_s)$. Clearly, we have $TP_s(\tau_s) = 1 - FN_s(\tau_s)$ and $TN_s(\tau_s) = 1 - FP_s(\tau_s)$.

5.4 Application-Aware Anomaly Detection

In this section, we present the problem of application-aware anomaly detection. First, we describe an approach for recovery in order to continue operation in the presence of detection alerts. Then, we quantify the utility losses in the application caused by potential detection errors. Followed by this, we formulate the problem of application-aware anomaly detection, i.e., the problem of finding detection thresholds so that the obtained utility in the presence of detection errors is as close as possible to the utility that could have been obtained if there were no detection errors.

Architecture. Figure 5.3 shows the architecture of the application-aware anomaly detection framework. If there is a detection alert, the prediction is routed to the application, instead of the measurement. The threshold of each detector is selected such that in the presence of detection error, the routed value (i.e., measurement or prediction) still obtains a utility close to the utility that could have been obtained if there were no detectors. (Note that in the figure, the predictor is not connected to the anomaly detector since this framework is applicable to any threshold-based detector, and not only regression-based detectors.)

5.4.1 Recovery

We present a recovery approach in order to continue operation in the presence of detection alerts. If sensor s is marked normal, then the observed measurement m_s is transmitted to the controller.

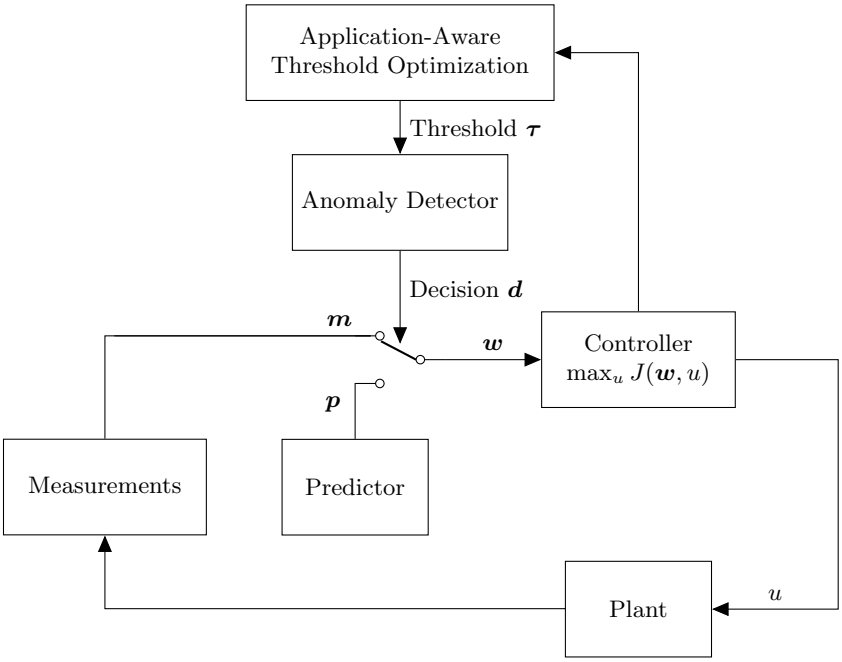


Figure 5.3: Architecture of Application-Aware Anomaly Detection.

However, if sensor s is marked anomalous, then the observed measurement is discarded and instead, the prediction p_s is transmitted to the controller. The switch in Figure 5.3 illustrates the idea. To formally represent this, let w_s denote the *recovered measurement* transmitted to the controller. Then, w_s can be described as

$$w_s = \begin{cases} m_s & \text{if } s \text{ is normal} \\ p_s & \text{if } s \text{ is anomalous} \end{cases}.$$

For our threshold-based detector defined by (5.3), the measurement of sensor s is marked normal if $|p_s - \ m_s| \leq \tau_s$ and anomalous otherwise. Therefore, for threshold-based detectors, the above equation can be re-written as

$$w_s(\tau_s) = \begin{cases} m_s & \text{if } |m_s - p_s| \leq \tau_s \\ p_s & \text{otherwise} \end{cases}. \tag{5.4}$$

Note that in this case, given prediction p_s and measurement m_s, the value of w_s depends on the threshold τ_s. From now on, when we want to highlight this dependence, we use the notation $w_s(\tau_s)$ instead of w_s. To summarize, given vectors of predictions $\boldsymbol{p}$, measurements $\boldsymbol{m}$, and thresholds $\boldsymbol{\tau}$,

using (5.4), we are able to compute the recovered measurement vector $\boldsymbol{w}(\boldsymbol{\tau})$ that is transmitted to the controller.

Note that using this approach, we assume that when a measurement is normal, it provides the best obtainable value for the sensor. Also, we assume that when a measurement is anomalous, the prediction provides the best obtainable value for the sensor. Handling uncertainties in sensor measurements and errors in predictions are beyond the scope of this chapter.

5.4.2 Worst-Case Utility Loss Due to Detection Error

The control input u (i.e., defined by (5.1)) depends on the recovered measurements $\boldsymbol{w}(\boldsymbol{\tau})$ (i.e., defined by (5.4)), and the recovered measurements $\boldsymbol{w}(\boldsymbol{\tau})$ depend on the detection thresholds $\boldsymbol{\tau}$. Therefore, the value of control input depends on thresholds $\boldsymbol{\tau}$. For example, if the thresholds are small (large), there will be many (few) detection alarms, and so predictions (measurements) will often be transmitted to the controller. Unfortunately, this will be problematic in the presence of detection errors.

Given threshold $\boldsymbol{\tau}$, let N be the set of sensors that are marked normal (i.e., $\forall s \in N, r_s \leq \tau_s$) and let A be the set of sensors that are marked anomalous (i.e., $\forall s \in A, r_s > \tau_s$). Based on the recovery method (5.4), the predictions are used for marked-anomalous sensors in A and measurements are used for marked-normal sensors in N to create the recovered measurement vector, i.e., $\boldsymbol{w} = \boldsymbol{p}_A \cup \boldsymbol{m}_N$. Next, given the recovered measurements $\boldsymbol{p}_A \cup \boldsymbol{m}_N$, the controller computes the control input $u_0 \in \text{argmax}_u J(\boldsymbol{p}_A \cup \boldsymbol{m}_N, u)$, concisely denoted by $U(\boldsymbol{p}_A \cup \boldsymbol{m}_N)$. This is expected to obtain the utility $J(\boldsymbol{p}_A \cup \boldsymbol{m}_N, u_0)$. However, the expected utility is obtained only if there is no detection error. Unfortunately, if there is a detection error, a different and potentially much lower utility is obtained.

Obtained Utility vs. Optimal Utility. We now quantify the actual obtained utility in presence of detection errors. Let $fp \subseteq A$ be the set of false positives, that is, sensors in fp are normal but they are marked anomalous. Since these sensors are normal, the measurements $\boldsymbol{m}_{fp}$ should have been transmitted to the controller, but due to false positives, the predictions were mistakenly transmitted. Similarly, let $fn \subseteq N$ be the set of false negatives, that is, sensors in fn are anomalous but they are marked normal. Since these sensors are anomalous, the predictions $\boldsymbol{p}_{fn}$ should have been transmitted to the controller but the measurements were mistakenly transmitted. Hence, for the control input $u_0 = U(\boldsymbol{p}_A \cup \boldsymbol{m}_N)$ computed above, the obtained utility will actually be $J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, u_0)$. On the other hand, if there were not detection errors, the optimal

control input would have been $u^* \in \text{argmax}_u J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, u)$, concisely denoted by $U(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn})$.

Utility Loss. To put this all together, given decisions A and N (computed given $\boldsymbol{r}$ and $\boldsymbol{\tau}$ as (5.3)), and the detection performance sets tp, fp, tn, and fn, the probability of occurrence of such detection error scenario is

$$\Pr(\boldsymbol{\tau}, tp, fp, tn, fn) = \prod_{s \in tp} TP_s(\tau_s) \cdot \prod_{s \in fp} FP_s(\tau_s) \cdot \prod_{s \in tn} TN_s(\tau_s) \cdot \prod_{s \in fn} FN_s(\tau_s) \, . \tag{5.5}$$

As discussed above, in this case, we could have obtained the optimal utility $J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, U(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}))$, but we obtained the smaller utility $J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, U(\boldsymbol{p}_A \cup \boldsymbol{m}_N))$. Thus, we incurred a utility loss of

$$\Delta J = \underbrace{J^*(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn})}_{\text{Optimal Utility}} - \underbrace{J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, U(\boldsymbol{p}_A \cup \boldsymbol{m}_N))}_{\text{Obtained Utility}} \, . \tag{5.6}$$

Hence, the expected utility loss of detection error scenario $tp \subseteq A$, $fp = A - tp$, $tn \subseteq N$, and $fn = N - tn$ is

$$C(\boldsymbol{\tau}, tp, fp, tn, fn) = \Pr(\boldsymbol{\tau}, tp, fp, tn, fn) \cdot \Delta J \, . \tag{5.7}$$

where $\Pr(\boldsymbol{\tau}, tp, fp, tn, fn)$ is obtained using (5.5) and ΔJ is obtained using (5.6).

Worst-Case Analysis. Since the sets of false positives and false negatives are not know a priori, we need to consider any possible scenario, and find the worst-cases. We define the worst-case loss due to detection errors below.

Definition 1 (Worst-Case Detection Error Loss)**.** *Given the thresholds $\boldsymbol{\tau}$ and the residuals $\boldsymbol{r}$, the worst-case loss due to detection errors is defined as*

$$L(\boldsymbol{\tau}) = \max_{\substack{tp \subseteq A, tn \subseteq N \\ fp = A - tp \\ fn = N - tn}} C(\boldsymbol{\tau}, tp, fp, tn, fn) \, , \tag{5.8}$$

where $C(\boldsymbol{\tau}, tp, fp, tn, fn)$ is defined as (5.7)*, and A and N are found using* (5.3)*.*

5.4.3 Formulation of Application-Aware Detector

Application-Aware Detector

To obtain resilience against the utility loss due to detection errors, the designer must choose the thresholds that result in the best performance with respect to the worst-case loss (5.8). An application-aware anomaly detector achieves this by finding the optimal thresholds $\boldsymbol{\tau}^*$.

Definition 2 (Application-Aware Detector)**.** *The Application-Aware Anomaly Detector is the detector that minimizes the loss* (5.8) *by finding the optimal thresholds* $\boldsymbol{\tau}^*$*, in other words*

$$\boldsymbol{\tau}^* \in \operatorname*{argmin}_{\boldsymbol{\tau}} L(\boldsymbol{\tau})\,. \tag{5.9}$$

If we are not able to change the thresholds at each timestep, and instead can change thresholds every T timesteps, we define

$$\bar{L}(\boldsymbol{\tau}) = \frac{1}{T}\sum_{k=1}^{T} L^k(\boldsymbol{\tau})\ , \tag{5.10}$$

and then we find thresholds that minimize the above equation.

Definition 3. *The Application-Aware Detector in a time period* T*, is the detector that minimizes the loss* (5.10) *by finding the optimal thresholds* $\boldsymbol{\tau}^*$*, in other words*

$$\boldsymbol{\tau}^* \in \operatorname*{argmin}_{\boldsymbol{\tau}} \bar{L}(\boldsymbol{\tau})\ . \tag{5.11}$$

Clearly, (5.9) is a special case of (5.11) as the latter becomes the former when $T = 1$.

5.5 Analysis

In this section, we analyze and solve the application-aware detection problems (5.9) and (5.11). First, we analyze the problem of worst-case detection error loss (5.8), and we prove that solving this problem is computationally challenging. We then present an efficient algorithm to obtain approximately optimal solutions. Second, we present Algorithm 5.3 to solve the application-aware detection problem (5.9) and obtain near-optimal thresholds. Finally, we propose Algorithm 5.4 to solve the problem of application-aware detection in a time period. The algorithm implements a variation of simulated annealing algorithm and finds near-optimal detection thresholds.

5.5.1 Algorithm for Worst-Case Detection Error Loss Problem

We begin our analysis by studying the computational complexity of finding worst-case loss due to detection errors (5.8). To this end, we formulate the problem of finding a worst-case loss as a decision problem.

Definition 4 (Worst-Case Detection Error Problem (Decision Version))**.** *Given a set of sensors S, detection thresholds $\boldsymbol{\tau}$, residuals $\boldsymbol{r}$, and desired loss L^*, determine whether there exists a detection error scenario that incurs the detection error loss of at least L^*.*

The following theorem establishes the computational complexity of finding a worst-case detection error.

Theorem 1. *Worst-Case Detection Error Problem (WCDE) is NP-Hard.*

Proof. We prove the above theorem using a reduction from a well-known NP-hard problem, the Maximum Independent Set Problem.

Definition 5 (Maximum Independent Set Problem (Decision Version))**.** *Given an undirected graph $G = (V, E)$ and a threshold cardinality k, determine whether there exists an independent set of nodes (i.e., a set of nodes such that there is no edge between any two nodes in the set) of cardinality k.*

Given an instance of the Maximum Independent Set Problem (MIS), that is, a graph $G = (V, E)$ and a threshold cardinality k, we construct an instance of the WCDE as follows:

- Let the set of sensors be $S := V$.
- Let $p_s = 0$ and $m_s = 1$ for every sensor $s \in S$.
- For every sensor $s \in S$, let $\tau_s = \epsilon$ where $\epsilon < 1$, so that $A = S$ and $N = \emptyset$.
- Let $TP_s(\tau_s) = FP_s(\tau_s) = TN_s(\tau_s) = FN_s(\tau_s) = 0.5$ for every sensor $s \in S$.
- Let the dimension of the control signal be $|S|$. For each element i of u, let $u_i \in \{0, 1\}$.
- Let the utility function be $J(\boldsymbol{w}, u) = \|\boldsymbol{w} \circ u\|_1$ if the non-zero elements in $\boldsymbol{w}$ form a non-empty independent set, and $-\|u\|_1$ otherwise.
- Finally, let the threshold loss be $L^* := \left(\frac{1}{2}\right)^{|S|} k$.

Clearly, the above reduction can be performed in polynomial time. Hence, it remains to show that the constructed instance of WCDE has a solution *if and only if* the given instance of MIS does.

MIS then WCDE. First, suppose that MIS has a solution, that is, there exists an independent set I of k nodes. We claim that the set $fp = I$ and $tp = S - I$ is a solution to WCDE. We have

$$\Delta J = J^*(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp}) - J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp}, U(\boldsymbol{p}_A))) = J^*(\boldsymbol{m}_{fp}) - J(\boldsymbol{m}_{fp}, <0>_{i\in 1}^{|S|}) = \|\boldsymbol{m}_{fp}\|_1 - 0 = k$$

Since $\Pr(\boldsymbol{\tau}, tp, fp, tn, fn) = \left(\frac{1}{2}\right)^{|S|}$ for any given sets of detection error, we obtain $L(\boldsymbol{\tau}) = \left(\frac{1}{2}\right)^{|S|} \cdot k$.

Not MIS then Not WCDE. Second, suppose that MIS has no solution, that is, every set of at least k nodes is non-independent. Then, we have that $J(\boldsymbol{w}, u) < k$ for every $\boldsymbol{w}$; otherwise, there would exist a set of at least k nodes in I that are independent of each other, which would contradict our supposition. Then, since $\Pr(\boldsymbol{\tau}, tp, fp, tn, fn) = \left(\frac{1}{2}\right)^{|S|}$, we conclude $L(\boldsymbol{\tau}) < \left(\frac{1}{2}\right)^{|S|} \cdot k$. □

We present Algorithm 5.2 which uses a greedy approach to obtain the worst-case loss due to detection errors. The algorithm starts considering a scenario of perfect detection, that is, $tp = A$, $fp = \emptyset$, $tn = N$ and $fn = \emptyset$. In each iteration, the algorithm moves an element from either tp or tn to respectively fp or fn that maximally increases the utility loss. If no such element exists, the algorithm terminates with the best solution found so far.

The runtime of Algorithm 5.2 depends on the function J, which depends on the considered application. If there is an oracle that computes $U(\boldsymbol{w})$ and $J(\boldsymbol{w}, U(\boldsymbol{w}))$ in constant time, the runtime of Algorithm 5.2 is linear with respect to $|S|$. That is, the runtime of Algorithm 5.2 is $\mathcal{O}(|S|)$.

5.5.2 Algorithm for Application-Aware Detection Problem

To solve the application-aware detection problem, we first prove the following lemma. The lemma shows that the application-aware detection problem is equal to the problem of selecting a set of normal sensors N and a set of anomalous sensors A, which has a much smaller search space than the original problem.

Lemma 1. *For sensor s with residual r_s, the optimal threshold with respect to* (5.9) *satisfies* $\tau_s \in \{0, r_s, r_s^+, M\}$.

Proof. We need to prove that for sensor s with residual r_s, the optimal threshold with respect to (5.9) is in the set $\{0, r_s, r_s^+, M\}$. First, let us recall that the optimal threshold is

$$\boldsymbol{\tau}^* \in \operatorname*{argmin}_{\boldsymbol{\tau}} \max_{\substack{tp\subseteq A, tn\subseteq N \\ fp=A-tp \\ fn=N-tn}} \Pr(\boldsymbol{\tau}, tp, fp, tn, fn) \cdot \Delta J \ ,$$

where $\Delta J = J^*(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}) - J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, U(\boldsymbol{p}_A \cup \boldsymbol{m}_N))$.

Algorithm 5.2 Algorithm for Computing Worst-Case Loss

```
1: function WORST_LOSS(τ, m, p)
2:     for all s ∈ S do
3:         (|m_s − p_s| ≤ τ_s) ? N ← N ∪ {s} : A ← A ∪ {s}
4:     end for
5:     tp ← A, fp ← ∅
6:     tn ← N, fn ← ∅
7:     L* ← 0
8:     while tp ≠ ∅ or tn ≠ ∅ do
9:         (C_i, i) ← max_{i⊆tp} C(τ, tp \ i, fp ∪ {i}, tn, fn)
10:        (C_j, j) ← max_{j⊆tn} C(τ, tp, fp, tn \ j, fn ∪ {j})
11:        if C_i < L_j then
12:            C ← C_j
13:            tn ← tn \ j
14:            fn ← fn ∪ {j}
15:        else
16:            C ← C_i
17:            tp ← tp \ i
18:            fp ← fp ∪ {i}
19:        end if
20:        if C* < C then
21:            C* ← C
22:        else
23:            return C*
24:        end if
25:    end while
26:    return C*
27: end function
```

Suppose there exists a set of optimal thresholds $\boldsymbol{\tau}^*$ such that some of its elements are not in the set mentioned above. Let s be one such sensor, that is, $\tau_s^* \notin \{0, r_s, r_s^+, M\}$. First, let $0 < \tau_s^* < r_s$. Clearly, $\Delta J(\boldsymbol{\tau}^*) = \Delta J(\boldsymbol{\tau}_{-s} \cup \{0\}) = \Delta J(\boldsymbol{\tau}_{-s} \cup \{r_s\})$. Then, we write $\frac{\Pr(\boldsymbol{\tau}', tp, fp, tn, fn)}{\Pr(\boldsymbol{\tau}^*, tp, fp, tn, fn)} = \frac{TP_s(\tau_s')}{TP_s(\tau_s^*)} > 1$ if $\tau_s' = 0$, and so τ_s^* can not be the optimal threshold if s is in the set of true positives. Also, $\frac{\Pr(\boldsymbol{\tau}', tp, fp, tn, fn)}{\Pr(\boldsymbol{\tau}^*, tp, fp, tn, fn)} = \frac{FP_s(\tau_s')}{FP_s(\tau_s^*)} > 1$ if $\tau_s' = r_s$, and so τ_s^* can not be the optimal threshold if s is in the set of false positives either. Second, let $r_s^+ < \tau_s^* < M$. Again, we have $\Delta J(\boldsymbol{\tau}^*) = \Delta J(\boldsymbol{\tau}_{-s} \cup \{r_s^+\}) = \Delta J(\boldsymbol{\tau}_{-s} \cup \{M\})$. Then, we write $\frac{\Pr(\boldsymbol{\tau}', tp, fp, tn, fn)}{\Pr(\boldsymbol{\tau}^*, tp, fp, tn, fn)} = \frac{TN_s(\tau_s')}{TN_s(\tau_s^*)} > 1$ if $\tau_s' = M$. Also, $\frac{\Pr(\boldsymbol{\tau}', tp, fp, tn, fn)}{\Pr(\boldsymbol{\tau}^*, tp, fp, tn, fn)} = \frac{FN_s(\tau_s')}{FN_s(\tau_s^*)} > 1$ if $\tau_s' = r_s^+$. This means that τ_s^* can not be the optimal threshold if s is in the set of true negatives or false negatives either. This contradicts our supposition, and thus, $\tau_s \notin \{0, r_s, r_s^+, M\}$ can never be correct. This concludes our proof. □

Following the above lemma, we present Algorithm 5.3 to obtain application-aware detection thresholds. The algorithm begins by initializing all sensors as normal, that is, $N = S$ and $A = \emptyset$. In each iteration, the algorithm moves a sensor from N to A, which maximally decreases the worst-case loss. To compute the worst-case loss, Algorithm 5.2 is used.

Algorithm 5.3 Algorithm for Design of Application-Aware Detector

```
1: function APPLICATION_AWARE(m, p)
2:     N ← S, A ← ∅
3:     L* ← ∞
4:     while A ≠ S do
5:         (L, s) ← argmin_{s∈N} WORST_LOSS(A ∪ {s}, N \ {s}, m, p)
6:         if L* < L then
7:             L* ← L
8:             A ← A ∪ {s}
9:             N ← N \ {s}
10:        else
11:            return L*
12:        end if
13:    end while
14:    return L*
15: end function
```

Similar to the previous algorithm, the running time depends on the function J and the considered application. If there is an oracle that returns $U(\boldsymbol{w})$ and $J(\boldsymbol{w}, U(\boldsymbol{w}))$ in constant time, the runtime of Algorithm 5.3 is $\mathcal{O}(|S|^2)$.

5.5.3 Algorithm for Application-Aware Detection in a Time Period

We present Algorithm 5.4 which solves the problem of application-aware detection in a time period T (2). The algorithm is based on a variation of simulated annealing algorithm, and finds near-optimal thresholds $\boldsymbol{\tau}$. The idea is to start with an arbitrary solution $\boldsymbol{\tau}$ and improving it iteratively. In each iteration, we generate a new candidate solution $\boldsymbol{\tau}'$ in the neighborhood of $\boldsymbol{\tau}$. If the candidate solution $\boldsymbol{\tau}'$ is better in minimizing the loss, then the current solution is replaced with the new one. However, if $\boldsymbol{\tau}'$ increases the loss, the new solution replaces the current solution with only a small probability. This probability depends on the difference between the two solutions in terms of loss as well as a temperature parameter which is a decreasing function of the number of iterations. These random replacements decreases the likelihood of getting stuck in a local minimum.

In Algorithm 5.4, PERTURB$(\boldsymbol{\tau}, n)$ defines the neighborhood of $\boldsymbol{\tau}$ in the nth iteration, from which $\boldsymbol{\tau}'$ is randomly sampled. More specifically, PERTURB$(\boldsymbol{\tau}, n)$ means that each τ_s in $\boldsymbol{\tau}$ is replaced by $\tau'_s = \tau_s + \Delta\tau_s$. Here, for each $s \in S$, $\Delta\tau_s$ is randomly picked from the uniform distribution over $\left[-\alpha\left(\frac{n_{\max}-n}{n_{\max}}\right),\ \alpha\left(\frac{n_{\max}-n}{n_{\max}}\right)\right]$ for some $\alpha \in \mathcal{R}_+$. Moreover, since τ'_s is nonnegative, we replace it with 0 if $\tau'_s < 0$.

Algorithm 5.4 Algorithm for Design of the Application-Aware in a Time Period

Input: $\boldsymbol{m}$, $\boldsymbol{p}$
Initialize: $\boldsymbol{\tau}$, $n \leftarrow 1$, T_0
$L(\boldsymbol{\tau}) \leftarrow \textsc{Worst\_Loss}(\boldsymbol{\tau}, \boldsymbol{m}, \boldsymbol{p})$
while $n \leq n_{\max}$ **do**
    $\boldsymbol{\tau}' \leftarrow \textsc{Perturb}(\boldsymbol{\tau}, n)$
    $L(\boldsymbol{\tau}') \leftarrow \textsc{Worst\_Loss}(\boldsymbol{\tau}', \boldsymbol{m}, \boldsymbol{p})$
    $c \leftarrow e^{(L(\boldsymbol{\tau}')-L(\boldsymbol{\tau}))/T}$
    if $(L(\boldsymbol{\tau}') < L(\boldsymbol{\tau})) \ \vee \ (\texttt{rand}(0,1) \leq c)$ **then**
        $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau}'$, $L(\boldsymbol{\tau}) \leftarrow L(\boldsymbol{\tau}')$
    end if
    $T \leftarrow T_0 \cdot e^{-\beta n}$
    $n \leftarrow n + 1$
end while
return $\boldsymbol{\tau}$

5.6 Special Cases

In this section, we consider two special cases for the application-aware detection problem (5.9). The first special case is single detector, which means that either there is a single detector in the system or each detector is optimized independently and irrespective of other detectors. The second special case is detectors with equal thresholds, where there are multiple detectors that have the same thresholds..

5.6.1 Single Detector

Consider a scenario where $|S| = 1$. This means that either there is a single detector in the system, or each detector is optimized independently and irrespective of other detectors. Let $S = \{a\}$ be the considered sensor, and let $r_a = |p_a - m_a|$ be the residual of the sensor at a timestep. First, we consider a threshold τ'_a where $r_a > \tau'_a$ for this sensor. This threshold results in a detection alert, and so the set of marked-anomalous sensors becomes $A = \{a\}$ and the set of marked-normal sensors becomes $N = \emptyset$. Next, to find the worst-case detection error loss (5.8) for this threshold, there are two possibilities for tp: 1) $tp = \{a\}$ and $fp = \emptyset$, and 2) $tp = \emptyset$ and $fp = \{a\}$. For $tp = \{a\}$, i.e., no detection error, we can write

$$C(\tau'_a, \{a\}, \emptyset, \emptyset, \emptyset) = TP(\tau'_a) \cdot \Big(J(p_a, U(p_a)) - J(p_a, U(p_a)) \Big) = 0 \, .$$

For the second scenario, i.e., $tp = \emptyset$ and $fp = \{a\}$, we should have used the measurement m_a but we used the prediction p_a, and so the expected utility loss is

$$C(\tau'_a, \emptyset, \{a\}, \emptyset, \emptyset) = FP(\tau'_a) \cdot \Big(J(m_a, U(m_a)) - J(m_a, U(p_a)) \Big) .$$

Therefore, the worst-case utility loss for the threshold τ'_a, where $r_a > \tau'_a$, is obtained using

$$L(\tau'_a) = \max C(\tau'_a, \emptyset, \{a\}, \emptyset, \emptyset) = C(r_a^-, \emptyset, \{a\}, \emptyset, \emptyset)$$

Next, we consider a threshold τ''_a such that $r_a \leq \tau''_a$. This threshold results in no detection alert, and so $A = \emptyset$ and $N = \{a\}$. To compute the worst-case detection error loss, there are two possibilities for detection error: 1) $tn = \{a\}$ and $fn = \emptyset$, and 2) $tn = \emptyset$ and $fn = \{a\}$. Similar to the above scenario, for the first case which corresponds to no detection error, we obtain $\Delta J = 0$ and so $C(\tau''_a, \emptyset, \emptyset, \{a\}, \emptyset) = 0$. For the second case, we obtain

$$C(\tau''_a, \emptyset, \emptyset, \emptyset, \{a\}) = FN(\tau''_a) \cdot \Big(J(p_a, U(p_a)) - J(p_a, U(m_a)) \Big) .$$

Therefore, the worst-case detection error loss for the threshold τ''_a, where $r_a \leq \tau''_a$, is

$$L(\tau''_a) = \max C(\tau''_a, \emptyset, \emptyset, \emptyset, \{a\}) = C(r_a, \emptyset, \emptyset, \emptyset, \{a\}) .$$

The application-aware detector selects the threshold $\tau_a^* \in \{r_a^-, r_a\}$ that solves

$$\min(\, C(r_a^-, \emptyset, \{a\}, \emptyset, \emptyset), C(r_a, \emptyset, \emptyset, \emptyset, \{a\}) \,) .$$

In other words, the optimal threshold is

$$\tau_a = \begin{cases} r_a^- & \text{if } C(r_a^-, \emptyset, \{a\}, \emptyset, \emptyset) \leq C(r_a, \emptyset, \emptyset, \emptyset, \{a\}) \\ r_a & \text{otherwise} \end{cases} . \tag{5.12}$$

5.6.2 Detectors with Equal Thresholds

We consider a case where all detectors have equal thresholds. Let $\bar{\tau}$ represent this threshold value, that is, $\bar{\tau} = \tau_1 = \ldots = \tau_d$. Next, let $r_1, r_2, \ldots, r_d$ be the residual values. The result below is a direct consequence of Lemma 1.

Algorithm 5.5 Application-Aware Detector for Equal Thresholds

Input: $\boldsymbol{m}$, $\boldsymbol{p}$

1: $\boldsymbol{r} \leftarrow |\boldsymbol{m} - \boldsymbol{p}|$

2: **for all** $\bar{\tau} \in$ `SolutionSpace` **do**

3: $\quad (L(\bar{\tau}), \bar{\tau}) \leftarrow$ WORST_LOSS$(\bar{\tau}, \boldsymbol{m}, \boldsymbol{p})$

4: **end for**

5: $\bar{\tau}^* \leftarrow \text{argmin}(L(\bar{\tau}), \bar{\tau})$

6: **return** $\bar{\tau}^*$

Corollary 1. *The optimal threshold $\bar{\tau}$ for detectors with equal thresholds belongs to the following set*

$$\textit{SolutionSpace} = \{0, r_1^-, r_1^+, r_2^-, r_2^+ \ldots, r_d^-, r_d^+, M\} \ .$$

Based on the above corollary, since the solution space is finite, we can find the optimal thresholds by a linear-time search, as presented by Algorithm 5.5.

5.7 Experiment

In this section, we apply our approach to a case study of max-pressure control of traffic signals in a traffic network. First, we construct regression-based anomaly detectors for traffic sensors, and we generate the trade-off curves for their performance. Then, we implement the application-aware detector, and evaluate its performance compared to a baseline "application-unaware" detector. Throughout the section, we use SUMO (Simulation of Urban MObility), which is a micro simulator for traffic applications [14].

Traffic Network

Consider a traffic network in a 3-by-3 grid with a total of 9 intersections, as show in Figure 5.4. Each intersection connects 4 standard two-way lanes with four possible movements {EW, WE, NS, SN}. Traffic volume of each movement is monitored by the set of sensors $S = \{s_{EW}, s_{WE}, s_{NS}, s_{SN}\}$. The sensors send traffic measurements $\boldsymbol{m} = \{m_{EW}, m_{WE}, m_{NS}, m_{SN}\}$ at each timestep. Each traffic signal has two phases $\Phi = \{\phi_{\{EW,WE\}}, \phi_{\{NS,SN\}}\}$. The max-pressure controller computes the optimal stage $u^* \in \Phi$ using (5.2).

The utility (i.e., pressure-release) function of the traffic network can be written as sum of the pressure-release of each individual intersection, and for each intersection, the utility depends only

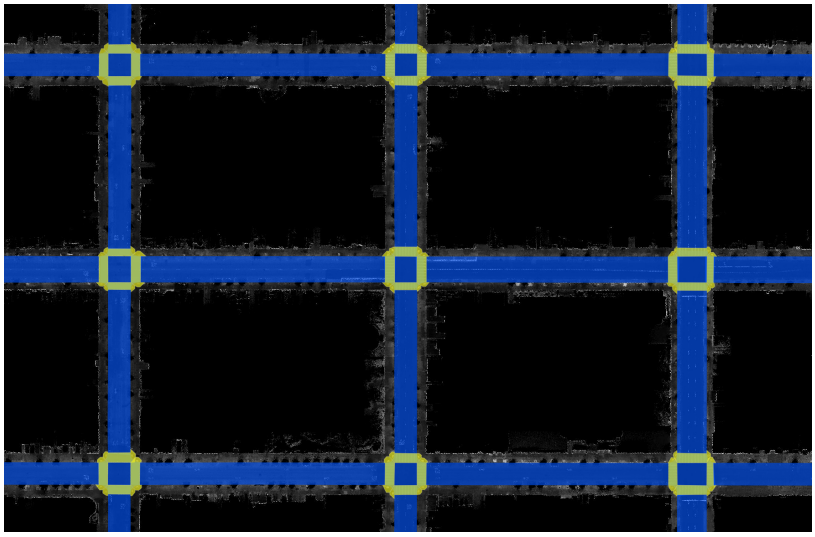


Figure 5.4: The 3-by-3 grid.

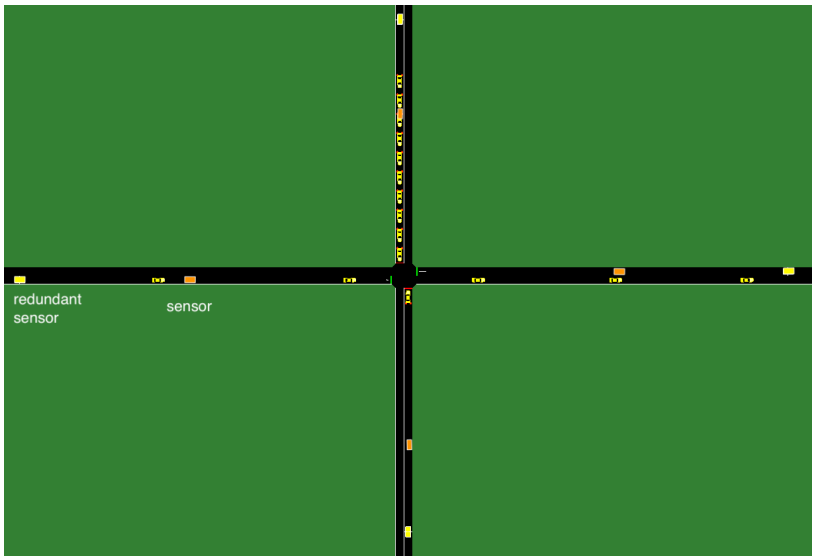


Figure 5.5: Each intersection in the 3-by-3 grid. For illustration, a critical and a redundant sensor are shown in the figure. For each lane, a second redundant sensor is placed with twice as much distance as the distance between the first redundant sensor and the critical sensor. All 8 total redundant sensors are used to predict the value of a sensor.

on its corresponding lanes. This means that maximizing the pressure-release of the traffic network is equal to maximizing the pressure-release of individual intersections, and so the application-aware threshold of each intersection can be designed independently of other intersections. Based on this observation, in what follows, we discuss how the detector is designed for an intersection and then extend it to all intersections.

Anomaly Model. Traffic measurements may be anomalous due to failures or other undesired

events. To simulate the negative effects of anomalies on the system, we consider several realistic anomaly models [139],

1. *Overcount*: Additive error equal to 3% to 7% of the actual values, i.e., $e_s(k) = u_s a_s(k)$ and $0.03 \leq u_s \leq 0.07$.

2. *Undercount*: Subtractive error equal to 7% to 13% of the actual values, i.e., $e_s(k) = u_s a_s(k)$ and $-0.13 \leq u_s \leq -0.07$.

3. *Gaussian Noise*: Error with zero mean and standard deviation $\sigma = 15$ to $\sigma = 35$, i.e., $e_s(k) \sim \mathcal{N}(0, \sigma^2)$ and $15 \leq \sigma \leq 35$.

Regression-Based Detector and Trade-off Curves

To protect the system against anomalies, we construct anomaly detectors. We suppose there are 8 (2 on each side) redundant sensors that are adjacent to the four critical sensors mentioned above. We use the values of these sensors to design regression-based anomaly detectors for the critical sensors. As discussed in the preceding chapters, such detector consists of two main components: 1) Predictor and 2) Statistical Test.

Predictor. We collect simulation data that represents the traffic behavior under normal operation. We simulate the network for 4 hours considering a Poisson distribution as the demand for each movement. We collect sensor measurements in 10-second aggregates. The data from the first 2 hours is used to train the predictors, and the data from the remaining 2 hours is used to obtain the trade-off curves. Following our previous discussion, for each predictor, we use the current value of the 8 redundant sensors as the features. Then, we train the predictor using linear regression algorithm. We obtain the performance metrics $MSE_{train} = 2.14$ and $MSE_{test} = 2.87$. Note that more complex regression algorithms (e.g., Gaussian Processes [47]) can be used as well, however, we obtained satisfactory result with a simple linear regression model.

Trade-off Curve. To generate the trade-off curve, first, we simulate the anomalies on the test data, and evaluate the performance of the detector by counting the number of true positives and false negatives. Similarly, we simulate the system under normal operation and evaluate the performance of the detector to obtain the number of true negatives and false positives. We repeat the steps while varying the detection threshold in order to obtain the trade-off curve (i.e., true positive probability as a function of false positive probability). Figure 5.6 shows the resulting trade-off curve.

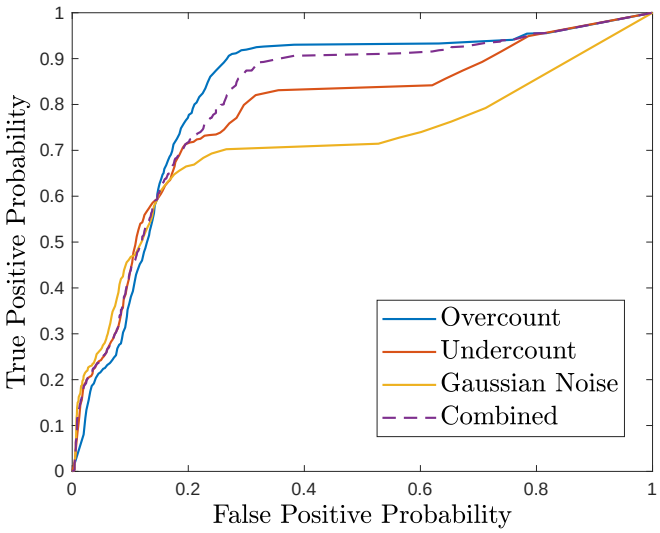


Figure 5.6: Trade-off between true positive and false positive errors for s_{EW}.

Application-Aware Detector

Given the trade-off curve, we implement the application-aware detector by finding the detection thresholds that minimize the worst-case expected utility loss. First, we show how the optimal threshold can be computed at each single timestep. Then, we present the results.

Computing the Optimal Threshold. To show how the optimal threshold is computed at each timestep, suppose $m_{NS} = p_{NS} = 15$ at a given timestep. Further, suppose there is no traffic on SN and WE. If $w_{EW} \geq 15$, the max-pressure controller selects the stage $u = \phi_{\{EW,WE\}}$, and otherwise, it selects $u = \phi_{\{NS,SN\}}$. Next, consider the following scenarios for m_{EW} and p_{EW}:

1. $m_{EW} = 30$ and $p_{EW} = 10$. In this case, the threshold that solves the following equation is the optimal threshold: $\min\big(C(20^-, \emptyset, \{s_{EW}\}, \emptyset, \emptyset), C(20, \emptyset, \emptyset, \emptyset, \{s_{EW}\})\big)$, where $C(20^-, \emptyset, \{s_{EW}\}, \emptyset, \emptyset) = FP(20^-) \cdot \big(J^*(30) - J(30, U(10))\big) = 30 - 15 = 15$, and $C(20, \emptyset, \emptyset, \emptyset, \{s_{EW}\}) = FN(20) \cdot \big(J^*(10)) - J(10, U(30))\big) = 15 - 10 = 5$. Thus, the optimal threshold is $\tau^* = 20$. To see why this is the optimal threshold, note that if $\tau = 20$, then there will be no detection alert and the measurement ($m_{EW} = 30$) will be used in computing the optimal control input (which is EW). If this is incorrect (due to a false negative), then $p_{SE} = 10$ is the actual value. In this case, the pressure-release of $\phi_{\{NS,SN\}}$ will be 10. In the perfect detection case (no detection error), we could have obtained 15 by selecting NS. Thus, the detection error loss is $15 - 10 = 5$. On the other hand, if $\tau = 20^-$, the prediction will be used by the controller. This results in the movement NS being selected. However, if there is a false positive, which means $m_{EW} = 30$ is the actual value, then EW should

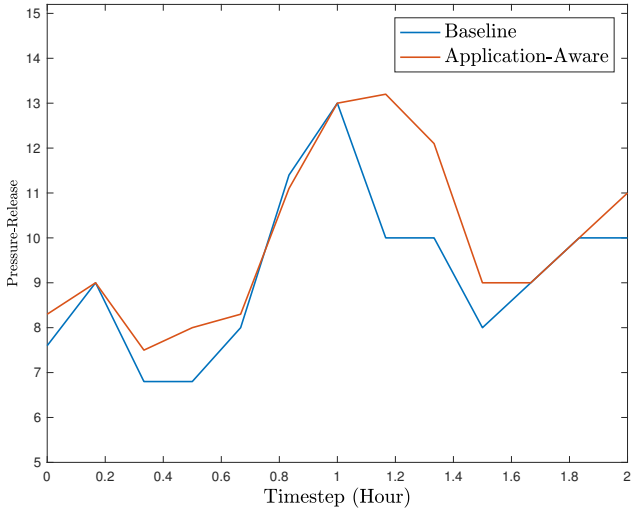


Figure 5.7: Utility (i.e., pressure-release) during a 2-hour interval.

have been selected as the optimal stage, which could have obtained a utility of 30. Thus, the utility loss in this case is $30-15=15$, which is larger than the utility loss for the other threshold.

2. $m_{EW}=30$, $p_{EW}=10$, and $a_{NS}=25$. The application-aware detector computes the minimum of $FP^{-}(20)\cdot\big(J^{*}(30)-J(30,U(10))\big)=30-25=5$, and $FN(20)\cdot\big(J^{*}(10)-J(10,U(30))\big)=25-10=15$. In this case, the threshold $\tau=20^{-}$ is the optimal threshold.

3. $m_{EW}=30$, $p_{EW}=25$, and $a_{NS}=15$. The application-aware detector computes $FP(20^{-})\cdot\big(J^{*}(30)-J(30,U(25))\big)=30-30=0$ and $FN(20)\cdot\big(J^{*}(25)-J(25,U(30))\big)=25-25=0$, which means that both thresholds are optimal. The same results can be obtained if $a_{NS}\geq m_{EW}$ and $a_{NS}\geq p_{EW}$.

Comparison. We now compare the results to a baseline detector that does not take into account the underlying application in the detector configuration. The threshold of the baseline detector is selected such that it obtains a false alarm probability equal to the false alarm probability of the application-aware detector. That is, we calculate the total number of false alarms for the application-aware detector, and then select the threshold that obtains the same false alarm probability for the baseline. In this case, in 2 hours of evaluation time where each of the anomalies may occur with probability of 0.05, each application-aware detector on average had a false alarm probability of 0.047. The threshold values for the application-aware detector varied from 1.3 to 26.5 with a mean of 5.6. The threshold for the baseline detector is selected as 3.9.

Figure 5.7 shows the pressure-release comparison between the two cases during the 2-hour interval. Each tick in the figure aggregates the results from 12 minutes (i.e., 72 timesteps). Based on the results, the application-aware detector performs better than the baseline detector in most cases. Baseline detector performs only slight better in the aggregated timestep 0.8 which could be due to detection errors by the application-aware detector. However, in the rest of the period, the proposed detector performs better.

5.8 Conclusions

We presented the application-aware anomaly detection framework for detection anomalies in sensor measurements in cyber-physical systems. An application-aware anomaly detector configures itself such that the application performance in the presence of detection errors is as close as possible to the performance that could have been obtained if there were no detection errors. We evaluated our result using a case study of real-time control of traffic signals, and showed that our application-aware detector significantly outperforms several baseline detectors.

Chapter 6

Adversarial Regression for Stealthy Attacks in Cyber-Physical Systems

Attacks manipulating the sensor measurements of a cyber-physical system (CPS) can be tuned by the attacker to cause a spectrum of damages. Attackers can attempt to remain undetected and hide their sensor manipulations by following the expected behavior of the system, while manipulating just enough sensor information that achieves their malicious goals. In this chapter, we study the problem of adversarial regression in CPS. We consider a safety-critical CPS that is monitored by regression-based anomaly detectors. An adversary attempts to drive the system to an unsafe state by perturbing the values of a subset of sensors while remaining undetected. We solve the adversarial regression problem, considering linear regression- and neural network regression-based detectors. Then, we present a resilient detector that mitigates the impact of stealthy attacks through resilient configuration of detection thresholds. We numerically evaluate the adversarial regression problem, and demonstrate the effectiveness of the resilient detector using a case study of a process control system.

6.1 Introduction

While it is crucial to detect intrusions into safety-critical CPS, most traditional intrusion detectors are faced with major drawbacks that make them insufficient for CPS. Specification-based detection methods require precise and accurate definitions of system behavior that may be impossible to obtain due to system complexity and nonlinearity. Signature-based methods are insufficient since the system's dynamics may preclude any succinct signatures that are needed. Further, formal models are ineffective in real-world CPS applications since they are often simplified to be tractable. As a solution, data-driven intrusion detectors are proposed which use machine learning models at their core. Data-driven detectors provide many advantages such as fast and accurate detection, applicability to complex and nonlinear systems, and relatively simple implementation, which make them suitable for anomaly and intrusion detection in CPS. Nevertheless, there are concerns about security vulnerabilities of such detectors. In particular, as previous work on adversarial machine learning shows, strategic adversaries may be able to exploit vulnerabilities of machine learning algorithms and evade being detected while achieving their malicious goals [86, 20].

Although prior studies make significant progress toward design of anomaly detectors for CPS in adversarial environments, several key gaps remain. First, previous work such as [28, 132] consider simplified mathematical models of the physical system, and consequently their approach might not generalize well in realistic environments. Second, while works such as [61, 1] consider data-driven approaches that can generalize well, they only consider very weak and restrictive attack models (e.g., an attacker that can only manipulate one sensor), and thus it is unclear whether the high detection rates reported in these works are due to the effectiveness of the approach or due to restrictive and unrealistic attack models.

In this chapter, we study the adversarial regression problem in CPS. We consider a safety-critical CPS that is monitored by regression-based anomaly detectors. An omniscient adversary that is capable of perturbing the values of a subset of sensors attempts to drive the system to an unsafe state (e.g., raising the pressure of a reactor beyond its safety limit in a process control system). The attacker needs to remain undetected during the duration of the attack. We solve the attacker problem, which we call the *adversarial regression* problem, considering different types of regression-based detectors. In particular, we solve the problem for detectors that use linear regression, neural network regression, and an ensemble of the two as their regression algorithms. Then, keeping in mind that it is impossible to completely secure the system against stealthy attacks, we present a resilient detector that mitigates the impact of stealthy attacks without increasing the number of false alarms. The resilient detector achieves this by careful selection of detection thresholds. Finally, we numerically evaluate the adversarial regression problem, and demonstrate the effectiveness of the resilient detector through applying our methods to a case study of process control systems.

The rest of this chapter is organized as follows. In Section 6.2, we discuss related work. In Section 6.3, we define sensor attacks in CPS and present a model of regression-based anomaly detectors. In Section 6.4, we present the adversarial regression problem for finding worst-case stealthy attacks against CPS. In Section 6.5, we analyze the adversarial regression problem for anomaly detectors that use linear regression, neural network, and an ensemble of the two as their regressors. In Section 6.6, we propose the resilient detector. In Section 6.7, we evaluate our contributions using a case study of process control systems. Finally, we offer concluding remarks in Section 6.8.

6.2 Related Work

Machine learning-based anomaly detectors have been widely studied in the literature as discussed in Chapter 2. In addition, many different anomaly detectors have been particularly designed for attack

detection in CPS [132]. However, to the best of our knowledge, there is little work that effectively applies regression-based detectors for attack detection in CPS, and studies their vulnerabilities to adaptive attackers. This may be due to the lack of publicly available normal data for CPS, or due to the safety-critical nature of such systems that makes it very challenging to collect attack data, as doing so may be damaging to the system. Nonetheless, if such limitations in data collection are overcome, regression-based detectors can be highly effective for anomaly detection in CPS.

Adversarial Machine Learning

Adversarial learning studies secure adoption of machine learning techniques in adversarial settings. A categorization of different adversary models is presented in [20]. Further, the work in [15] considers use of multiple classifiers to obtain better security performance in adversarial settings. A significant number of recent works study the adversarial learning problem for neural networks used in computer vision applications [41], [106], [52], [112].

Adversarial classifier reverse engineering (ACRE) learning problem was first introduced in [86]. In ACRE, an adversary launches an attack to minimize a cost function, while having limited information about the classifier. The adversary is allowed to make polynomial number of membership queries to modify an attack instance to bypass the classifier with minimal cost. In the paper, a query algorithm for reverse engineering of linear classifiers is presented. Furthermore, in [31], a framework for the adversarial learning problem is presented that uses game theory. The paper formulates a game between a classifier and an adversary, and then assuming that the adversary's optimal strategy is known, develops the optimal classifier. Then, experiments are performed which show that the proposed robust classifier outperforms a standard one.

Machine Learning-Based Attack Detection in CPS

A behavior-based machine learning approach for attack detection in CPS is proposed in [61]. The work models the physical process of CPS to detect any anomaly or attack that may try to change the behavior of system. Using a replicate of a real water treatment facility, different data-driven anomaly detectors are implemented and their performance is evaluated. While this work successfully applies machine learning algorithms for attack detection in CPS, the considered threat models are highly simple and restrictive (e.g., attacking only one sensor). Thus, it is unclear whether the high detection rates reported in the paper are due to the effectiveness of the approach or due to the restrictive and unrealistic attack models. Using the same testbed, the impact of single-point cyber-attacks are experimentally investigated in [1]. Cyber-attacks are launched through a SCADA

server connected to programmable logic controllers that govern the actuators and sensors. Several experiments are performed considering different objectives for the attacker. Then, based on the observed experiment results, attack detection mechanisms are proposed. This work also suffers from the same limitation of considering oversimplified and unrealistic attack scenarios.

Data-driven anomaly detection is studied using a miniature industrial gas system with a few sensors and actuators in [102]. To detect attacks, one physical process attribute (i.e., pressure in pipeline) and two one-class classifiers (i.e., support vector data description and kernel principal component analysis) are used. Then, different attack scenarios are considered to evaluate the detection performance. According to the results presented in the paper, while single-point attacks have a detection rate of 99.25 percent, more complicated attacks have much lower detection rates of approximately 65-70 percent. This shows that the performance of the detector significantly decreases when faced by complex attacks. Further, an intrusion detection method based on neural networks is investigated using a water testbed that contains a tank, pump, and level sensor [45]. Network traffic, SCADA mode, water level, and pump status are used as the attributes, and then the method is evaluated considering malicious attacks on the level sensor. Finally, using the same testbed, a one-class approach is studied in [103].

6.3 System Model

In this section, we define sensor attacks in CPS. We then present a regression-based anomaly detector for detecting sensor attacks.

Notation

Each element s of vector y at timestep k is denoted by y_s^k. We omit the timestep symbol unless we need to distinguish between several timesteps. Further, parameter x of detector s is described by $x^{(s)}$. For a list of symbols used in this chapter, see Table 6.1.

6.3.1 Sensor Attacks in CPS

Consider a CPS such as an industrial process control system or an intelligent transportation system, that is monitored by a set of sensors. Sensors may be under attack by malicious adversaries that have penetrated into the system through exploiting zero-day security vulnerabilities. If sensor $s \in S$ is under attack, there is a discrepancy between the actual and observed measured values. In other words, if y^k is the actual value and $\tilde{y}^k$ is the observed measurement at timestep k, for an attacked

Table 6.1: List of Symbols

Symbol	Description
S	Set of sensors
y_s	Actual value of sensor s
$\tilde{y}_s$	Observed measurement value of sensor s
$\hat{y}_s$	Predicted value of sensor s
r_s	residual of sensor $s \in S$
S_c	Set of critical sensors
δ_s	Error added to sensor $s \in S$
B	Budget of the attacker
D	Set of sensors with detectors
$f^{(s)}$	Regression-based predictor of detector $s \in D$
$\tau^{(s)}$	Threshold value of detector $s \in D$

sensor s, we have $\tilde{y}(k) = y^k + \delta_s^k$, where δ_s^k is the added error value at timestep k.

Undetected sensor attacks may cause significant harm to the system and degrade the performance extensively. This is because corrupted measurements transmitted to the controller may result in control decisions which lead the system to an undesirable or unsafe state. For example, as we show in our experiments for a process control system, undetected attacks may result in disastrous events such as reactor explosion.

6.3.2 Regression-Based Anomaly Detection

To protect the system against sensors attacks, we must detect them quickly and accurately. Many different anomaly detection systems have been proposed in the literature. For a comprehensive review of anomaly detection methods, we refer the reader to [29] for data-driven detectors and [132] for detectors used in CPS. In this work, we consider machine learning (ML) regression-based anomaly detectors due to the many advantages that they provide. ML regression-based detectors: 1) require no knowledge of the physical system; 2) can take into account complex and nonlinear behaviors of the system which makes them more realistic (as opposed to detectors that use simplified mathematical models of the system); and 3) are easy to implement and can be highly scalable depending on the algorithm that is used.

Figure 6.1 presents the architecture of the ML regression-based anomaly detector. The detector consists of two components: 1) Predictor and 2) Statistical Test. The predictor predicts the value of a sensor given some information about the system state (e.g., current value of other sensors, previous control inputs). Then, the statistical test compares the computed prediction to the observed measurement and decides whether the sensor is normal or anomalous.

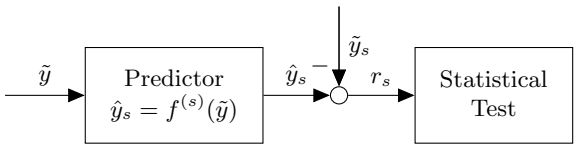


Figure 6.1: Regression-Based Anomaly Detection

6.3.2.1 Predictor

As discussed above, the goal of the predictor is to predict the value of a sensor given some information about the system state (e.g., value of other sensors at a previous timestep). In other words, for sensor s, our goal is to find a function $f^{(s)}$ that maps features $\tilde{y}$ to value of sensor s, i.e., $\hat{y}_s = f^{(s)}(\tilde{y}_{-s})$.

To do so, first, large amounts of data that correctly represent the normal system behavior are collected. Next, for sensor $s \in S$ (e.g., the pressure of a reactor), we let the target variable be its actual value at timestep k, denoted by $\tilde{y}_s(k)$. Further, as feature variables, we can select the current or previous values of other sensors, previous control inputs, and other information about the system. We let $\tilde{y}$ denote such feature vector.

The predictor can be implemented using suitable machine learning regression algorithms. For example, in this work, we consider linear regression and neural network models. To ensure that the predictor can successfully generalize, we evaluate it using our hold-out test data. For example, for the prediction result of our case study, see Figure 6.6.

6.3.2.2 Statistical Test

For each detector $s \in D$, the statistical test efficiently detects attacks or anomalies by comparing the measured value $\tilde{y}_s(k)$ with the predicted value $\hat{y}_s(k)$. Given a set of measured values $\tilde{y}$ and predicted values $\hat{y}$, residual signals are computed as $r = |\hat{y} - \tilde{y}|$. Then, given the residuals, the statistical test makes detection decisions $d = \{d_s\}_{s \in D}$, where for each sensor s, the decision d_s is either normal or anomalous.

Different detection algorithms can be used to implement the statistical test [11]. In this work, we consider a stateless threshold-based detector defined as follows. Given detection threshold $\tau^{(s)} \in \mathbb{R}_+$, for detector of sensor s, if the residual r_s is less than or equal to the threshold

$\tau^{(s)}$, then sensor s is marked normal and otherwise, s is marked anomalous. Thus

$$d_s = \begin{cases} \text{normal} & \text{if } r_s \leq \tau^{(s)} \\ \text{anomalous} & \text{otherwise} \end{cases}. \tag{6.1}$$

6.4 Adversarial Regression

In this section, we present the adversarial regression problem for CPS. The idea is to find a stealthy sensor attack that maximizes the deviation from the actual value for some critical sensor. We describe this in more detail below. Figure 6.2 illustrates the adversarial regression problem.

Threat Model We define the attack model by describing the attacker's capability, knowledge, objective, and strategy [20]. **Capability:** The adversary may control a subset of sensors and perturb their values. The cardinality of this subset has an upper bound B. **Knowledge:** We consider a worst-case scenario where the attacker has complete knowledge of the system, anomaly detectors, implementations, and so on. **Objective:** The attacker's objective is to maximize or minimize the observed value for some critical sensor $a \in S_c$. **Strategy:** The attack must remain undetected during its entire duration.

Formally, the attacker adds bias δ to the actual measurements and transmits corrupted measurements $\tilde{y} = y + \delta$ to the controller. For a given critical sensor $a \in S_c$, the attacker's goal is to maximize or minimize the observed measurement $\tilde{y}_s$. As an example, the attacker may wish to increase the pressure of a reactor to unsafe levels. To accomplish this, (s)he would manipulate *observed* sensor measurements by transmitting smaller-than-desired pressure values, which triggers the controller to increase the pressure. Here we focus on maximization of observed values of critical sensors; minimization can be accomplished similarly.

The main constraint faced by the attacker is to remain stealthy, as determined by the operation of the anomaly detector, since otherwise the attack is detected and can be mitigated. Formally, for any detector $s \in D$ with threshold $\tau^{(s)}$, the difference between the prediction $f^{(s)}(\tilde{y})$ and the measurement $\tilde{y}_s$ must be smaller than or equal to the threshold, i.e., $|\tilde{y}_s - f^{(s)}(\tilde{y})| \leq \tau^{(s)}$. The attacker can manipulate at most B sensors due to constrained resources. Further, the value of any sensor $i \in S$ can change at most by η_i between two consecutive observations, i.e., $|\tilde{y}_i - \tilde{y}_i^{prev}| \leq \eta_i$, where $\tilde{y}_i^{prev}$ is the sensor value at the previous timestep [91].

We formally define the adversarial regression problem below.

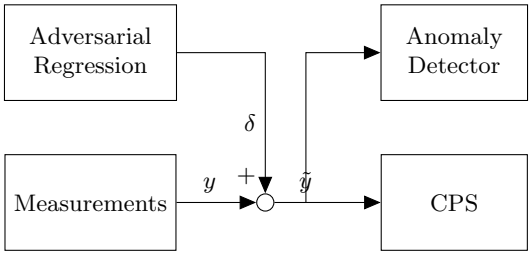


Figure 6.2: Attack Model

Definition 1 (Adversarial Regression)**.** *Given an anomaly detector, a set of critical sensors S_c, and an attack budget B, the adversarial regression problem finds an optimal attack that maximizes the observed measurement for some critical sensor, while evading the detector during the entire duration of the attack. In other words,*

$$\begin{aligned} &\underset{a \in S_c}{\operatorname{argmax}} \ \max_{\tilde{y}} \tilde{y}_a \\ &\text{s.t.} \quad |\tilde{y}_s - f^{(s)}(\tilde{y})| \leq \tau^{(s)}, \forall s \in D \\ &\qquad |\tilde{y}_i - \tilde{y}_i^{k-1}| \leq \eta_i, \forall i \in S \\ &\qquad \|\tilde{y} - y\|_0 \leq B \end{aligned} \tag{6.2}$$

6.5 Analysis of Adversarial Regression Problem

We begin our analysis by showing that the adversarial regression problem is, in general, NP-Hard.

Definition 2 (Adversarial Regression Problem (Decision Version))**.** *Given an anomaly detector, a set of critical sensors S_c, an attack budget B, and a threshold attack $\tilde{y}_c^*$, determine whether there exists an attack with a value of at least $\tilde{y}_c^*$.*

Proposition 1. *The Adversarial Regression Problem is NP-Hard.*

We prove this proposition using a reduction from a well-known NP-hard problem, the Maximum Independent Set Problem.

Definition 3 (Maximum Independent Set Problem (Decision Version))**.** *Given an undirected graph $G = (V, E)$ and a threshold cardinality k, determine whether there exists an independent set of nodes (i.e., a set of nodes such that there is no edge between any two nodes in the set) of cardinality k.*

Proof. Given an instance of the Maximum Independent Set Problem (MIS), that is, a graph $G = (V, E)$ and a threshold cardinality k, we construct an instance of the Adversarial Regression Problem

(ARP) as follows: 1) Let the set of sensors be $S := V \cup \{c\}$, where c is not connected to any node, and let $S_c := \{c\}$ be the only critical sensor in the system. 2) Let $D := S$. For each detector $s \in D$, let $\tau^{(s)} := 0$. Further, let $f^{(s)}(\tilde{y})$ be the number of nonzero elements in $\tilde{y}$, if $\tilde{y}_s \neq 0$ and those elements form an independent set, and zero otherwise. 3) Let $y_s := 0$ for every sensor $s \in S$. 4) Let $B := k + 1$. 5) Finally, let the threshold objective be $y_c^* := k + 1$.

Clearly, the above reduction can be performed in polynomial time. Hence, it remains to show that the constructed instance of ARP has a solution *if and only if* the given instance of MIS does.

First, suppose that MIS has a solution, that is, there exists an independent set A of k nodes. We claim that the set $A' = A \cup \{c\}$ where $\tilde{y}_s = k + 1$ for every $s \in A'$, and $\tilde{y}_s = 0$ otherwise, is a solution to ARP. For nonzero sensors, since A' is independent, the value of $f^{(s)}(\tilde{y})$ is equal to the number of nonzero sensors in A', which is equal to $k + 1$. This satisfies the stealthiness constraint since for nonzero sensors $|\tilde{y}_s - f^{(s)}(\tilde{y})| = |k + 1 - (k + 1)| = 0 \leq \tau^{(s)}$, and for zero sensors $|\tilde{y}_s - f^{(s)}(\tilde{y})| = |0 - 0| = 0 \leq \tau^{(s)}$. Clearly, the budget constraint is also satisfied, and so $y_c^* = k + 1$ is obtained.

Second, suppose that MIS has no solution, that is, every set of at least k nodes is non-independent. As a consequence, we have $f^{(s)}(\tilde{y}) < k + 1$ for every detector, since otherwise, there would exist a set of at least $k + 1$ nodes (which could include c) that are independent of each other, which would contradict our supposition. Since $|\tilde{y}_c - f^{(c)}(\tilde{y})| < \tau_c = 0$ implies that $\tilde{y}_c = f^{(c)}$, ARP cannot have a solution, which concludes our proof. □

Despite the hardness result, we now present algorithmic approaches for solving the adversarial regression problem (6.2) in the context of two widely-used machine learning algorithms: linear regression and neural networks.

6.5.1 Linear Regression

Suppose that the predictor of each ML regression-based anomaly detector implements a linear regression model. In other words, for each detector $s \in D$, we let the predictor be a linear regression model defined as $f^{(s)}(\tilde{y}) = {w^{(s)}}^T \tilde{y} + b^{(s)}$, where ${w^{(s)}}^T \in \mathbb{R}^m$ and $b^{(s)} \in \mathbb{R}$ are the parameters of the linear regression model. Clearly, we also have $w_s^{(s)} = 0$. In this case, the first constraint of the adversarial regression problem (6.2) becomes a set of linear equations. That is, for each $s \in D$, the constraint $|\tilde{y}_s - f^{(s)}(\tilde{y})| \leq \tau^{(s)}$ will be written as $|\tilde{y}_s - {w^{(s)}}^T \tilde{y} - b^{(s)}| \leq \tau^{(s)}$.

For all $i \in S$, we replace $\tilde{y}_i$ with $\tilde{y}_i = y_i + \alpha_i \delta_i$, where $\alpha_i = 1$ implies that the sensor is attacked, and $\alpha_i = 0$ implies that the sensor is not attacked. Thus, the third constraint (i.e., $\|\tilde{y} - y\|_0 \leq B$)

can also be re-written as $\sum_{i=1}^{m} \alpha_i \leq B$. Further, we remove α_i from each $\alpha_i \delta_i$ term, and instead add a new constraint $\delta_i \leq M\alpha_i$ where $M \in \mathbb{R}$ is a large number.

Next, we let $u^{(s)} = w^{(s)} - e_s$, where e_s is a one-hot vector with value of 1 at index s and 0 at other indices. We place all the coefficient vectors $u^{(s)}$ in a matrix as follows, $U = \left[u^{(s_1)} \dots u^{(s_d)}\right]^T$. We also let $\gamma^{(s)} = \tau^{(s)} - y_s + {w^{(s)}}^T y$, and then define $\Gamma = \left[\gamma^{(s_1)} \vdots \gamma^{(s_d)}\right]^T$. The first constraint of (6.2) then becomes $U\delta \leq \Gamma$. Similarly, for negative operand values in the absolute operator, we let ${\gamma^{(s)}}' = \tau^{(s)} + y_s - {w^{(s)}}^T y$ and we place them in a matrix Γ' to obtain the constraint $-U\delta \leq \Gamma'$.

Hence, we obtain the following mixed-integer linear programming (MILP) problem, which solves the adversarial regression problem (6.2) for linear regression

$$
\begin{aligned}
&\underset{a \in S_c}{\text{argmax}} \ \max_{\delta, \alpha} \delta_a \\
&\text{s.t.} \quad U\delta \leq \Gamma, \ -U\delta \leq \Gamma' \\
&\qquad \delta \leq M\alpha \\
&\qquad |\delta_i| \leq \eta_i, \forall i \in S \\
&\qquad \sum_{i=1}^{m} \alpha_i \leq B \\
&\qquad \forall i \in S : \delta_i \in \mathbb{R}, \alpha_i \in \{0, 1\}.
\end{aligned}
\tag{6.3}
$$

6.5.2 Neural Network

Next, suppose the predictor of each ML regression-based detector implements a neural network regression model. In other words, for each detector $s \in D$, we let the predictor $f^{(s)}$ be a feed-forward neural network model defined by parameters $\theta^{(s)}$, where the prediction $\hat{y}_s = f(\tilde{y})$ is obtained by computing a forward-propagation. Because of the neural networks, the first constraint of the adversarial regression problem now becomes a set of highly nonlinear functions, and so we are not able to use techniques from linear optimization as in the previous section.

To tackle this challenge and be able to approximately solve (6.2), we present Algorithm 6.1. The algorithm solves the problem by making small steps in a direction that optimally increases the objective function. At each iteration, the algorithm linearizes the neural networks for all the detectors at their operating points, and replaces them with the obtained linearized instances. Then, for each small step, it uses (6.3) to solve the adversarial regression problem. At each iteration, in order to ensure that the obtained solution is feasible, the algorithm tests the solution in the actual space. If it is infeasible, the iteration is repeated considering a smaller step size. Otherwise, the

Algorithm 6.1 Adversarial Regression for Neural Network

Input: Measurements $\tilde{y}$, critical sensors S_c, budget B, algorithm parameters ϵ_0, ϵ_{min}, n_{max}

$\tilde{y}_0 \leftarrow \tilde{y}$, $\delta_{a^*} \leftarrow 0$, $\epsilon \leftarrow \epsilon_0$
for all $a \in S_c$ **do**
  while number of iterations ¡ n_{max} **and** ϵ ¿ ϵ_{min} **do**
    `// Linearize the neural networks for all the detectors at` $\tilde{y}$
    for all $s \in D$ **do**
      for all $i \in S$ **do**
        $w_i^{(s)} \leftarrow \frac{\partial f^{(s)}(\tilde{y})}{\partial \tilde{y}_i}$
      end for
    end for
    $W \leftarrow [w_1^{(s_1)} \dots w_n^{(s_1)}; \dots; w_1^{(s_d)} \dots w_n^{(s_d)}]$
    `// Solve MILP and check feasibility`
    $\tilde{y}' \leftarrow$SOLVE_MILP(W,ϵ,a,$\tilde{y}$,$\tilde{y}_0$)
    if $\forall s \in D : |\tilde{y}'_s - f^{(s)}(\tilde{y}')| \leq \tau^{(s)}$ **then**
      $\tilde{y} \leftarrow \tilde{y}'$
      $\delta_a \leftarrow \tilde{y}'_a$
      $\epsilon \leftarrow \epsilon_0$
    else
      $\epsilon \leftarrow \frac{\epsilon}{2}$
    end if
  end while
  if $\delta_{a^*} < \delta_a$ **then**
    $a^* \leftarrow a$, $\delta_{a^*} \leftarrow \delta_a$, $\tilde{y}^* \leftarrow \tilde{y}$
  end if
end for
return $\tilde{y}^*$

same process is repeated until a local optimum is found or we reach a desired maximum number of iterations.

The algorithm begins with the initial uncorrupted measurement vector $\tilde{y} = y$. For each neural network $f^{(s)}$, it obtains a linearized model by computing the partial derivative of the output $f^{(s)}(\tilde{y})$ with respect to the inputs at the current operating point $\tilde{y}$, i.e., $\bar{w}_i^{(s)} = \frac{\partial f^{(s)}(\tilde{y})}{\partial \tilde{y}_i}$ is computed for all $i \in S$. Then, for detector $s \in D$, the linearized model can be written as $\bar{f}^{(s)}(\tilde{y} + \Delta) = f^{(s)}(\tilde{y}) + \bar{w}^{(s)^T}\Delta$, where $\Delta \in \mathbb{R}^m$ is a small error vector. We denote the coefficients of the linearized models in matrix form by $W = [w^{(s_1)} \dots w^{(s_d)}]^T$.

Given matrix W, we solve the problem by converting it to the MILP (6.3) as in the previous subsection. The difference is that there is now a new constraint that enforces taking small error steps. In other words, we let $\epsilon \in \mathbb{R}_+$ be the parameter representing the maximum step size. Then, we add the constraint $|\delta| < \epsilon$ to the MILP. In addition, we update the budget constraint as $\|\delta + \Delta\|_0 \leq B$, where Δ is the error vector added at the current iteration, and $\delta = \tilde{y} - y$ is the error added in the previous iterations.

Let $\tilde{y}'$ be the solution obtained by solving the MILP. We check whether this solution is feasible by performing forward-propagation in all the neural networks and checking that no stealthiness constraint is violated. If a stealthiness constraint is violated, which means that our linearized model was not a good approximation of the neural network, we discard the candidate solution, reduce the

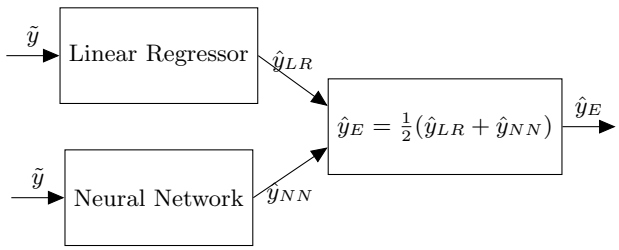


Figure 6.3: Ensemble of Neural Network-Linear Regression.

maximum step size parameter ϵ to improve the approximation, and re-solve the MILP. We repeat this process until a feasible solution is found, in which case the same steps are repeated for a new operating point, or until we reach the maximum number of iterations n_{max}. Finally, we check whether the solution that is found for the current target sensor a outperforms the solution for the best target sensor found so far. The algorithm terminates after considering all the target sensors and returns the optimal attack vector.

6.5.3 Neural Network-Linear Regression Ensemble

It has been shown in the adversarial learning literature that multiple detectors may improve adversarial robustness [15]. We explore this idea for the ML regression-based detector by considering an ensemble model for the predictor. As shown in Figure 6.3, we consider an ensemble predictor that contains a neural network and a linear regression model. Different methods can be used for combining the results, but we consider a bagging approach where the result is computed as the average of the predictors' outputs.

We can see the ensemble as a single neural network that connects a perceptron (i.e., the linear regression model) with our initial neural network at the output layer. Thus, to solve the adversarial regression problem, we can use the same approach as in the case of neural networks by modifying Algorithm 6.1. That is, in line 9, we compute the linearized coefficient matrix as $W = \frac{1}{2}(W_{NN} + W_{LR})$ where W_{NN} is the matrix of linearized weights of the neural networks obtained by computing the partial derivatives with respect to input, and W_{LR} is the fixed weights of the linear regression model.

6.6 Resilient Detector Design

In this section, we present an approach to obtain resilience against adversarial regression in CPS. In particular, we discuss a method to select thresholds for our detectors such that a right balance

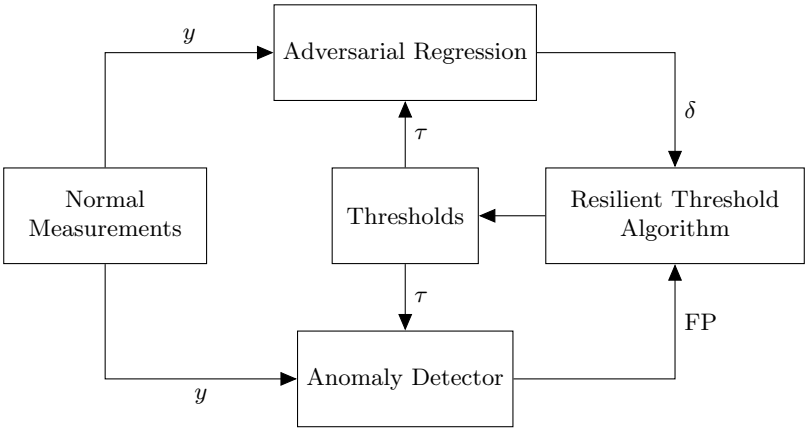


Figure 6.4: Resilient Detector Algorithm.

between the impact of stealthy attacks and the expected time between false alarms. This approach is inspired by the detection metric proposed in [132].

Impact of Attack and False Alarms Given thresholds τ, we define the *impact of attack* as the max value of the adversarial regression problem (6.2) in a time period $\mathcal{K} = \{k_1, \ldots, k_T\}$, and we define it as $D(\tau) = \max_{k \in \mathcal{K}} \tilde{y}_s^k - y_s^k$. To evaluate false alarms, we let T_E be the duration of an experiment under normal operation. Then, for each detector $s \in D$, the number of false alarms can be calculated using $nFA^{(s)}(\tau^{(s)}) = \sum_{k=1}^{T_E} 1_{|\tilde{y}_s^k - f^{(s)}(\tilde{y}^k)| > \tau^{(s)}}$. We then calculate the expected time between false alarms as $\mathbb{E}[T_{fa}^{(s)}(\tau^{(s)})] = \frac{T_E}{nFA^{(s)}(\tau^{(s)})}$, and then calculate it for all detectors as $\mathbb{E}[T_{fa}(\tau)] = \sum_{s \in D} \mathbb{E}[T_{fa}^{(s)}(\tau^{(s)})]$.

We may be able to reduce the attack impact $D(\tau)$ by decreasing the detection threshold. This can effectively force the attacker to launch attacks that follow the behavior of the physical system. By following closer the behavior of the system, the attack impact is then reduced as the attack needs to appear to be a normal behavior of the physical system. However, reducing the threshold will in turn trigger a higher number of false alarms.

Maintaining a balance between the attack impact and the number of false alarms can be a challenging problem [46, 74]. We now present a novel algorithm for resilient detector threshold selection in the context of stealthy sensor attacks described above. Given a desired value for the expected time between false alarms, the algorithm greedily finds thresholds that minimize the attack impact while keeping the expected number of false alarms fixed.

Algorithm for Obtaining Resilient Detector Let τ be initialized based on a desired value for the expected time between false alarms. At each iteration, we solve the adversarial regression

Algorithm 6.2 Resilient Detector

Initialize: Set thresholds τ based on a desired false alarm performance
while number of iterations ¡ n_{max} **do**
  for all $a \in S_c$ **do**
    $(D_a(\tau), a) \leftarrow$ ADVERSARIAL_REGRESSION(τ, a)
  end for
  $A^* \leftarrow \text{argmax}_{A \subseteq S_c} D_a(\tau)$
  $A_{min} \leftarrow \text{argmin}_{A \subseteq S_c} D_a(\tau)$
  for all $a \in A^*$ **do**
    $\tau^{(a)'} \leftarrow \tau^{(a)} - \epsilon$
  end for
  $\Delta FP \leftarrow FP(\tau') - FP(\tau)$
  for all $b \in A_{min}$ **do**
    $\tau^{(b)'} \leftarrow FP_b^{-1}(FP_b(\tau^{(b)}) - \frac{1}{|A_{min}|}\Delta FP)$
  end for
  $D_{A'}(\tau') \leftarrow \max_{a \in S_c}$ ADVERSARIAL_REGRESSION(τ', a)
  if $D_{A'}(\tau') \leq D_{A^*}(\tau)$ **then**
    $\tau \leftarrow \tau'$
    $\epsilon \leftarrow \epsilon_0$
  else
    $\epsilon \leftarrow \frac{\epsilon}{2}$
  end if
end while
return τ

problem to find the largest stealthy attack impact $D_{A^*}(\tau)$, where A^* is the set of critical sensors that have such stealthy attack impact (note that this can be more than one sensor). Similarly, we let A_{min} be the set of sensors that have the smallest stealthy attack impact $D_{A_{min}}(\tau)$. To reduce $D_{A^*}(\tau)$, we assign $\tau^{(a)}$ for each $a \in A^*$ to a smaller threshold. Then, to keep the number of false alarms unchanged, we increase the threshold $\tau^{(b)}$ for each $b \in A_{min}$ to compensate for the false alarms added by detectors of sensors in A^*. These steps are repeated until there is no progress in the solution. We formalize this in Algorithm 6.2, where we let $FP_s(\tau^{(s)}) = \mathbb{E}[T_{fa}^{(s)}(\tau^{(s)})]$ for readability.

6.7 Experiments

In this section, we evaluate our contributions considering a case study of a safety-critical process control system. In particular, we study the well-known Tennessee-Eastman process control system (TE-PCS). First, we design regression-based anomaly detectors for the critical sensors in the system (e.g., pressure of the reactor, level of the product stripper). Then, we consider scenarios where an adversary launches sensor attacks using the adversarial regression problem in order to drive the system to an unsafe state. We evaluate the resilience of the system against such attacks using baseline detectors and the proposed resilient detector.

6.7.1 Tennessee-Eastman Process Control System

We present a brief description of the Tennessee-Eastman Process Control System (TE-PCS). TE-PCS involves two simultaneous gas-liquid exothermic reactions for producing two liquid products [37]. The process has five major units: reactor, condenser, vapor-liquid separator, recycle compressor, and product stripper. The chemical process consists of an irreversible reaction which occurs in the vapour phase inside the reactor. Two non-condensible reactants A and C react in the presence of an inert B to form a non-volatile liquid product D. The feed stream 1 contains A, C, and trace of B; feed stream 2 is pure A; stream 3 is the purge containing vapours of A, B, and C; and stream 4 is the exit for liquid product D.

Safety Constraints and Control Objectives There are 6 safety constraints that must not be violated (e.g, safety limits for the pressure and temperature of reactor). These safety constraints correspond to 5 critical sensors: pressure, level, and temperature of the reactor, level of the product stripper, and level of the separator. Further, there are several control objectives that should be satisfied, e.g., maintaining process variables at desired values and keeping system state within safe operating conditions. The monitoring and control objectives are obtained using 41 measurement outputs and 12 manipulated variables.

Simulation of Sensor Attacks We use the revised Simulink model of TE-PCS [13]. We consider the implementation of the decentralized control law as proposed by [116]. To launch sensor attacks against TE-PCS, we update the Simulink model to obtain an information flow similar to Figure 6.2. That is, the adversary receives all the sensor measurements and control inputs, solves the adversarial regression problem, and then adds the error vector to the actual measurements.

Figure 6.5 shows how a sensor attack may drive the system to an unsafe state. In this scenario, the pressure of the reactor exceeds 3000 kPa which is beyond the safety limit and can result in reactor explosion.

6.7.2 Regression-Based Anomaly Detector

Collection of Normal Data To protect the system against sensor attacks, we build a detector for each critical sensor (i.e., $D = S_c$). To train predictors for the machine learning regression-based detectors, we need data that correctly represent system behavior in different operation modes. To do so, we run simulations that model the system operation for 72 hours. We collect the sensor

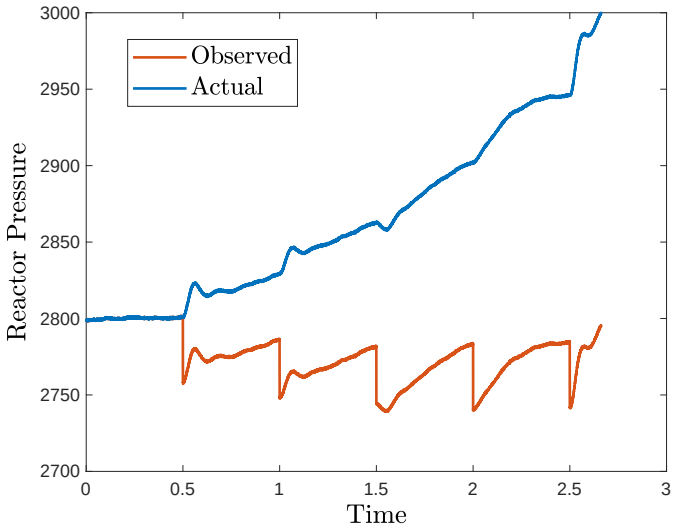


Figure 6.5: Pressure of the reactor when a sensor attack starts at $k = 0.5$. After 2 hours the pressure reaches an unsafe state.

measurements (i.e., xmeas) and control inputs (i.e., xmv). Each simulation consists of 7200 timesteps and thus, for each simulation scenario, we record 7200×53 datapoints. To make sure that the dataset represents different modes of operation, we repeat the same steps for different initial state values. We consider a total of 20 different initial state values which gives us $20 \times 7200 \times 53 \approx 7.5$ million datapoints. Three of the control inputs are always constant and so effectively we only store 50 values at each timestep.

Linear Regression-Based Detector

Using our collected data, we train linear regression models for the critical sensors. We use the current value of the remaining 36 non-critical sensors as well as the 9 non-constant control inputs as features of the model. Figure 6.6 shows the performance of the linear regression predictor on training and test data. Note that since the data is sequential, the train and test data cannot be randomly sampled and instead, we divide the data in two blocks. Also, to be able to compare the performance of predictors trained for different variables, we compute the MSE for normalized values instead of the actual values.

For the temperature variable, we using a ridge (l_2 regularized) linear regression model with $\lambda = 1500$ selected using cross-validation. With this model we obtain $MSE_{train} = 0.70$ and $MSE_{test} = 0.75$.

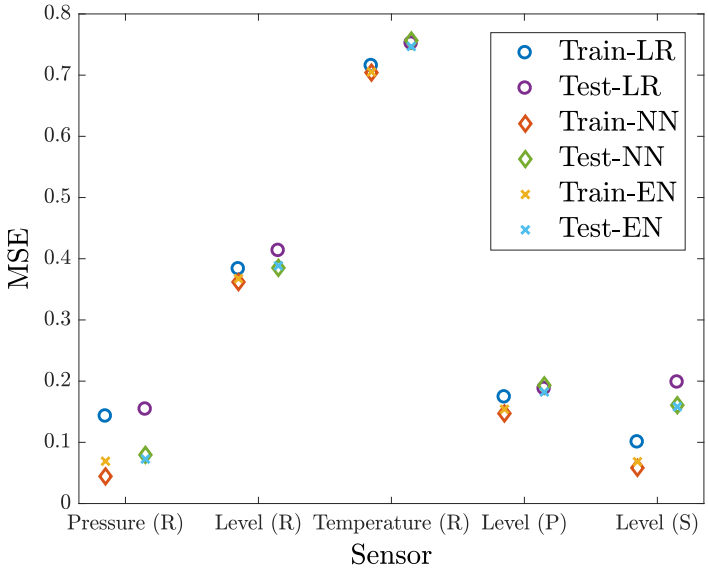


Figure 6.6: MSE of Linear Regression, Neural Network, and Ensemble Model (computed with normalized data).

Table 6.2: MSE of Test Data (Original Scale)

	PR	LR	TR	LP	LS
LR-MSE	9.05	0.26	0.00	1.53	7.17
NN-MSE	6.28	0.25	0.00	1.41	5.90
EN-MSE	5.74	0.26	0.00	1.33	5.81

Neural Network-Based Detector

Next, we train the neural network regression models for the critical sensors. Unlike linear regression models, neural networks require a few parameters that need to be selected (e.g., network architecture, activation function, optimization algorithm, regularization technique). We considered neural networks with 2 to 4 hidden layers and 10 to 20 neurons in each layer. All the neuron in the hidden layers use tanh activation functions. We also experimented with ReLU activation functions but tanh performs better. We trained the networks in Tensorflow for 5000 epochs using Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, and a learning rate of 0.01. Figure 6.6 shows the MSE for the training and test sets, which outperforms the scores of the linear regression model. Figure 6.6 shows the result for the ensemble model as well.

6.7.3 Adversarial Regression

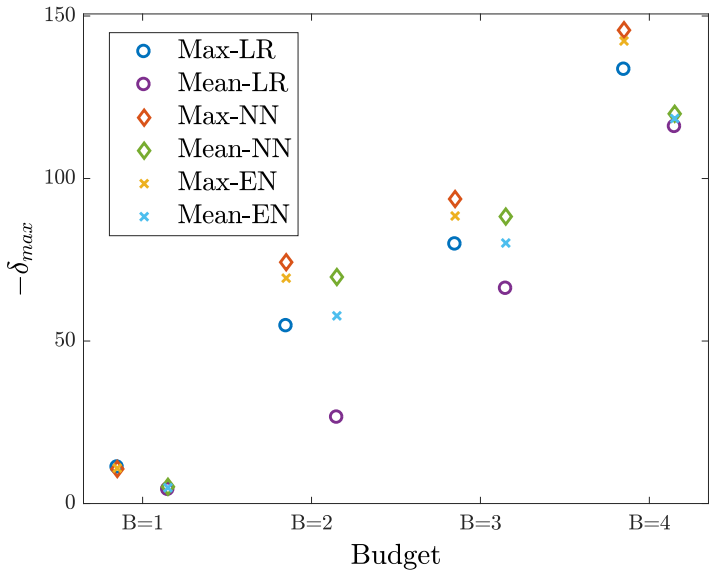
We solve the adversarial regression problem considering linear regression-, neural network-, and the ensemble-based detectors. Figure 6.7a shows the maximum and mean of the solution of the adversarial regression problem as a function of the attacker's budget for the pressure of the reactor. As expected, a powerful attacker that has a higher budget is able to carry out larger stealthy perturbations. Further, Figure 6.7b shows the solution of the adversarial regression for each critical sensor considering different detectors. In the y-axis, the distance parameter is the deviation that is needed in order to reach an unsafe state. Based on our results, level of the separator seems to be the most critical parameter. As it can be seen, the temperature of the reactor is the least critical sensor while the level of the stripper is the most critical. Also, Algorithm 6.1 efficiently computed worst-case attacks, which depending on the selected hyper-parameters, took between 20-100 iterations.

Surprisingly, neural network models tend to be more vulnerable than linear regression. Further, ensembles of linear and neural network regression do not significantly reduce this vulnerability either. The only interesting exception is the product stripper sensor, which is indeed substantially more vulnerable to attacks.

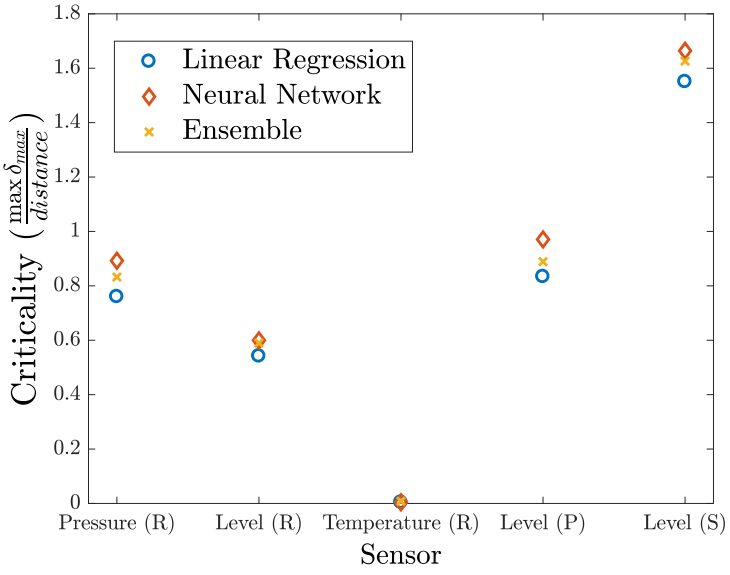
6.7.4 Resilient Detector

We use the resilient detector algorithm to find threshold values that reduce the stealthy attack impact as defined in Section 6.6. We do this in the context of linear regression. Let $T^* = 1$ hour, be the desired value for the expected time between false alarms for all detectors. As the baseline, for each detector $s \in D$, we set threshold values $\tau_s = FP_s^{-1}(T^* \cdot |D|)$. This way, each detector is expected to generate 5 alarms per day as shown in Figure 6.8b.

Then, we use Algorithm 6.2 to change thresholds in order to improve resilience. At first iteration, stripper level is the most critical sensor, and reactor temperature is the least critical sensor, therefore, the algorithm decreases the threshold corresponding to the stripper level, and increases the threshold for the other detectors in order to maintain the same false alarm performance. These steps are repeated until the algorithm converges to a local optimum (which in our implementation took about 20 iterations). As shown in Figure 6.8, the worst-case stealthy attack impact (i.e., stripper level) is reduced compared to the baseline. This is obtained by increasing the stealthy attack impact for sensors that were less critical. Also, note that this improvement in resilience is attained with zero cost (i.e., no increase in false alarms).

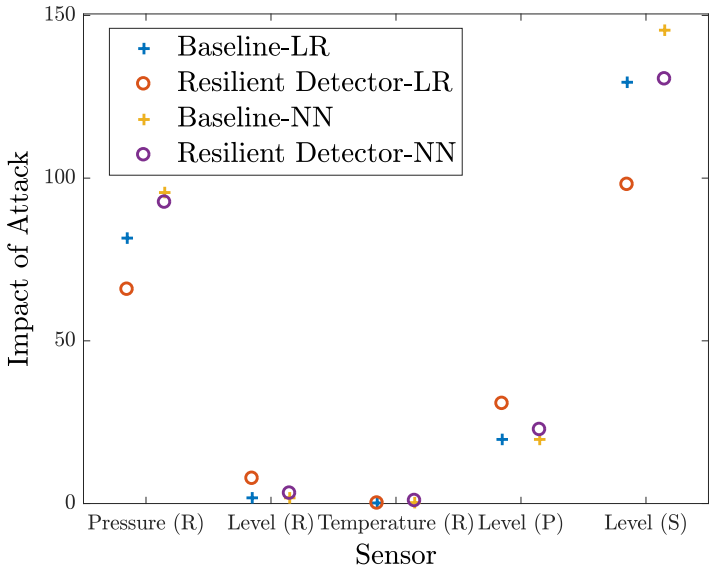


(a)

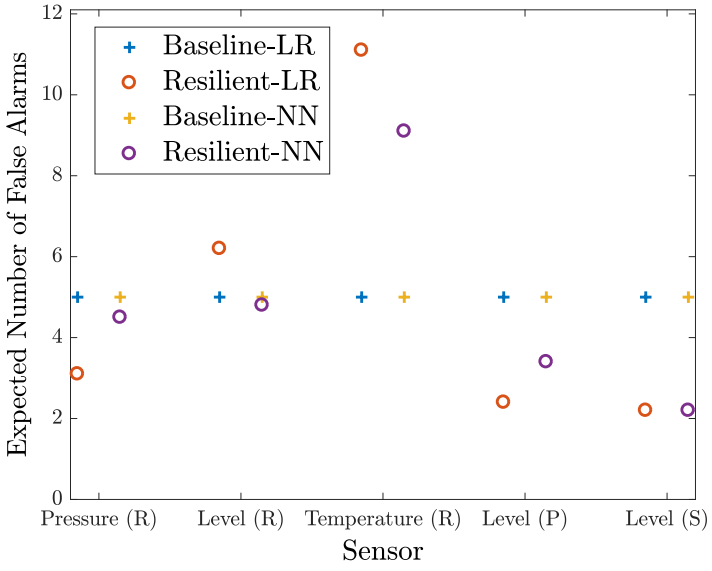


(b)

Figure 6.7: (a) Adversarial regression for the pressure of the reactor considering different budgets. Surprisingly, linear regression outperforms neural networks. In the figure, δ_{max} is the maximum error that can be added to the measurements of a critical sensor at a timestep. (b) Criticality analysis of the five safety-critical sensors. Criticality is defined as the maximum of δ_{max} during a time interval over distance, where distance is the difference between operating point and safety limit for a critical sensor.



(a)



(b)

Figure 6.8: Resilient Detector compared to Baseline. (a) Impact of Attack. (b) Number of False Positives (per day).

6.8 Conclusions

In this chapter, we studied the adversarial regression problem in CPS. We considered a scenario where the CPS is monitored by machine learning regression-based anomaly detectors. As our threat model, we considered an omniscient adversary that is capable of perturbing the values of a subset of

sensors. The adversary's objective is to lead the system to an unsafe state (e.g., raising the pressure of a reactor in a process control system beyond its safety limit) without being detected. We solved the adversarial regression problem considering linear regression and neural network. Surprisingly, we discovered that the neural network model is more vulnerable than the linear regression model. Then, we presented a resilient detector that mitigates the impact of stealthy attacks through resilient configuration of detection thresholds. We numerically evaluated the adversarial regression problem, and demonstrated the effectiveness of the resilient detector using a case study of a process control system.

Chapter 7

Conclusions

The vulnerability of CPS to anomalies has resulted in many research topics in the design, evaluation, and implementation of resilient anomaly detection methods in CPS. The goal of this thesis was to address the various challenges facing the problem of resilient detection in CPS to ensure successful and survivable operation. Throughout the thesis, we observed how incorporating knowledge of the physical system in the design of anomaly detectors can lead to improved resilience. Most of our analysis was performed considering powerful attackers as this can provide an upper bound on the worst performance of the anomaly detection tools. Our results were supported using theoretical results as well as practical implementations. In particular, we validated our approaches considering real-world CPS such as water distribution systems, intelligent transportation systems, and process control systems. We hope that our results aid the theoreticians as well as the practitioners in the design and implementation of Resilient Cyber-Physical Systems.

References

[1] Sridhar Adepu and Aditya Mathur. An investigation into the response of a water treatment system to cyber attacks. In *2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE)*, pages 141–148. IEEE, 2016.

[2] Cesare Alippi and Manuel Roveri. An adaptive CUSUM-based test for signal change detection. In *Proceedings of the 2006 IEEE ISCAS*, pages 5752–5755, 2006.

[3] Tansu Alpcan and Tamer Basar. A game theoretic approach to decision and analysis in network intrusion detection. In *Proceedings of the 42nd IEEE Conference on Decision and Control (CDC)*, volume 3, pages 2595–2600. IEEE, 2003.

[4] Tansu Alpcan and Tamer Başar. A game theoretic analysis of intrusion detection in access control systems. In *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC)*, volume 2, pages 1568–1573. IEEE, 2004.

[5] Tansu Alpcan and Tamer Basar. An intrusion detection game with limited observations. In *Proceedings of the 12th International Symposium on Dynamic Games and Applications*, 2006.

[6] Saurabh Amin, Alvaro A Cárdenas, and S Shankar Sastry. Safe and secure networked control systems under denial-of-service attacks. In *International Workshop on Hybrid Systems: Computation and Control*, pages 31–45. Springer, 2009.

[7] Saurabh Amin, Xavier Litrico, S Shankar Sastry, and Alexandre M Bayen. Stealthy deception attacks on water scada systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, pages 161–170. ACM, 2010.

[8] Saurabh Amin, Xavier Litrico, S Shankar Sastry, and Alexandre M Bayen. Cyber security of water scada systemspart ii: Attack detection using enhanced hydrodynamic models. *IEEE Transactions on Control Systems Technology*, 21(5):1679–1693, 2013.

[9] Saurabh Amin, Xavier Litrico, Shankar Sastry, and Alexandre M Bayen. Cyber security of water scada systemspart i: analysis and experimentation of stealthy deception attacks. *IEEE Transactions on Control Systems Technology*, 21(5):1963–1970, 2013.

[10] Jonathan Arad, Mashor Housh, Lina Perelman, and Avi Ostfeld. A dynamic thresholds scheme for contaminant event detection in water distribution systems. *Water Research*, 2013.

[11] Michèle Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: Theory and application*, volume 104. Prentice Hall, Englewood Cliffs, 1993.

[12] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. In *Algorithm Engineering*, pages 19–80. Springer, 2016.

[13] Andreas Bathelt, N Lawrence Ricker, and Mohieddine Jelali. Revision of the tennessee eastman process model. *IFAC-PapersOnLine*, 48(8):309–314, 2015.

[14] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[15] Battista Biggio, Giorgio Fumera, and Fabio Roli. Multiple classifier systems for robust classifier design in adversarial environments. *International Journal of Machine Learning and Cybernetics*, 1(1-4):27–41, 2010.

[16] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *International Workshop on Multiple Classifier Systems*, pages 350–359. Springer, 2011.

[17] Battista Biggio, Zahid Akhtar, Giorgio Fumera, Gian Luca Marcialis, and Fabio Roli. Security evaluation of biometric authentication systems under real spoofing attacks. *IET biometrics*, 1 (1):11–24, 2012.

[18] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning (ICML)*. Omnipress (arXiv preprint arXiv:1206.6389), 2012.

[19] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer Berlin Heidelberg, 2013.

[20] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, 2014.

[21] Rakesh B Bobba, Katherine M Rogers, Qiyan Wang, Himanshu Khurana, Klara Nahrstedt, and Thomas J Overbye. Detecting false data injection attacks on dc state estimation. In *Preprints of the First Workshop on Secure Control Systems*, volume 2010, 2010.

[22] Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 547–555. ACM, 2011.

[23] Michael Brückner, Christian Kanzow, and Tobias Scheffer. Static prediction games for adversarial learning problems. *Journal of Machine Learning Research*, 13(Sep):2617–2654, 2012.

[24] Caltrans. Performance measurement system (pems). Available: http://pems.dot.ca.gov, 2016. [Accessed: 10/25/2016].

[25] CANARY. Canary: a water quality event detection tool. http://waterdata.usgs.gov/nwis/, 2010. [Online; accessed October 20, 2016].

[26] A. A. Cardenas, S. Amin, and S. Sastry. Secure control: Towards survivable cyber-physical systems. In *2008 The 28th International Conference on Distributed Computing Systems Workshops*, pages 495–500, June 2008.

[27] Alvaro A Cárdenas, Saurabh Amin, and Shankar Sastry. Research challenges for the security of control systems. In *Proceedings of the 3rd Conference on Hot Topics in Security*, page 6. USENIX Association, 2008.

[28] Alvaro A Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 355–366. ACM, 2011.

[29] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[30] Lin Chen and Jean Leneutre. A game theoretical framework on intrusion detection in heterogeneous networks. *IEEE Transactions on Information Forensics and Security*, 4(2):165–178, 2009.

[31] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 99–108. ACM, 2004.

[32] Gyorgy Dan and Henrik Sandberg. Stealth attacks and protection schemes for state estimators in power systems. In *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 214–219. IEEE, 2010.

[33] Katherine R Davis, Kate L Morrow, Rakesh Bobba, and Erich Heine. Power flow cyber attacks and perturbation-based defense. In *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pages 342–347. IEEE, 2012.

[34] Yong Deng, Wen Jiang, and Rehan Sadiq. Modeling contaminant intrusion in water distribution networks: A new similarity-based dst method. *Expert Systems with Applications*, 38(1): 571–578, 2011.

[35] Armando Di Nardo, Michele Di Natale, Mario Guida, and Dino Musmarra. Water network protection from intentional contamination by sectorization. *Water Resources Management*, 27 (6):1837–1850, 2013.

[36] Julian Dibbelt, Thomas Pajor, and Dorothea Wagner. User-constrained multimodal route planning. *Journal of Experimental Algorithmics (JEA)*, 19, 2015.

[37] James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Computers & chemical engineering*, 17(3):245–255, 1993.

[38] Lemonia Dritsoula, Patrick Loiseau, and John Musacchio. Computing the nash equilibria of intruder classification games. In *International Conference on Decision and Game Theory for Security*, pages 78–97. Springer, 2012.

[39] Lili Du, Srinivas Peeta, and Yong Hoon Kim. An adaptive information fusion model to predict the short-term link travel time distribution in dynamic traffic networks. *Transportation Research Part B: Methodological*, 46(1):235–252, 2012.

[40] Mohsen Estiri and Ahmad Khademzadeh. 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), pages 16. IEEE, 2010*, pages 1–6. IEEE, 2010.

[41] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physicalw-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2017.

[42] Emeka Eyisi and Xenofon Koutsoukos. Energy-based attack detection in networked control systems. In *Proceedings of the 3rd International Conference on High Confidence Networked Systems*, pages 115–124. ACM, 2014.

[43] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[44] H. Fawzi, P. Tabuada, and S. Diggavi. Security for control systems under sensor and actuator attacks. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3412–3417, Dec 2012.

[45] Wei Gao, Thomas Morris, Bradley Reaves, and Drew Richey. On scada control system command and response injection and intrusion detection. In *eCrime Researchers Summit (eCrime), 2010*, pages 1–9. IEEE, 2010.

[46] Amin Ghafouri, Waseem Abbas, Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Optimal thresholds for anomaly-based intrusion detection in dynamical environments. In *Proceedings of Decision and Game Theory for Security: 7th International Conference, GameSec 2016, New York, NY, USA, November 2-4, 2016*, volume 9996, page 415. Springer, 2016.

[47] Amin Ghafouri, Aron Laszka, Abhishek Dubey, and Xenofon Koutsoukos. Optimal detection of faulty traffic sensors used in route planning. In *Second International Workshop on Science of Smart City Operations and Platforms Engineering (SCOPE)*, 2017.

[48] Annarita Giani, Eilyan Bitar, Manuel Garcia, Miles McQueen, Pramod Khargonekar, and Kameshwar Poolla. Smart grid data integrity attacks: characterizations and countermeasures π. In *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 232–237. IEEE, 2011.

[49] Robert D Gibbons. Use of combined shewhart-cusum control charts for ground water monitoring applications. *Ground Water*, 37(5):682–691, 1999.

[50] Peter H Gleick. Water and terrorism. *Water Policy*, 8(6):481–503, 2006.

[51] Amir Globerson and Sam Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine learning*, pages 353–360. ACM, 2006.

[52] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[53] Abhishek Gupta, Cédric Langbort, and Tamer Başar. Optimal control in the presence of an intelligent jammer with limited actions. In *2010 49th IEEE Conference on Decision and Control (CDC)*, pages 1096–1101. IEEE, 2010.

[54] Rachana A Gupta and Mo-Yuen Chow. Performance assessment and compensation for secure networked control systems. In *2008 34th Annual Conference of IEEE Industrial Electronics*, pages 2929–2934. IEEE, 2008.

[55] John Hall, Alan D Zaffiro, Randall B Marx, Paul C Kefauver, E Radha Krishnan, Roy C Haught, and Jonathan G Herrmann. On-line water quality parameters as indicators of distribution system contamination. *Journal (American Water Works Association)*, 99(1):66–77, 2007.

[56] David Hart, Sean A McKenna, Katherine Klise, Victoria Cruz, and Mark Wilson. CANARY: A water quality event detection algorithm development tool. In *Proceedings of the World Environmental and Water Resources Congress*, pages 1–9, 2007.

[57] Xiali Hei, Xiaojiang Du, Shan Lin, and Insup Lee. Pipac: Patient infusion pattern based access control scheme for wireless insulin pump system. In *2013 Proceedings IEEE INFOCOM*, pages 3030–3038. IEEE, 2013.

[58] Nathan Henry, Nathanael Paul, and Nicole McFarlane. Using bowel sounds to create a forensically-aware insulin pump system. In *HealthTech*, 2013.

[59] Erik Hollnagel. Resilience engineering. *PSYKOLOGIA*, 42(6):493, 2007.

[60] Austin Jones, Zhaodan Kong, and Calin Belta. Anomaly detection in cyber-physical systems: A formal methods approach. In *2014 IEEE 53rd Annual Conference on Decision and Control (CDC)*, pages 848–853. IEEE, 2014.

[61] Khurum Nazir Junejo and Jonathan Goh. Behaviour-based attack detection and classification in cyber physical systems using machine learning. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pages 34–43. ACM, 2016.

[62] Yiannis Kamarianakis, Angelos Kanas, and Poulicos Prastacos. Modeling traffic volatility dynamics in an urban network. *Transportation Research Record: Journal of the Transportation Research Board*, pages 18–27, 2005.

[63] Tung T Kim and H Vincent Poor. Strategic protection against data injection attacks on power grids. *IEEE Transactions on Smart Grid*, 2(2):326–333, 2011.

[64] István Kiss, Béla Genge, and Piroska Haller. A clustering-based approach to detect cyber attacks in process control systems. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 142–148. IEEE, 2015.

[65] Katherine A Klise and Sean A McKenna. 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, 2006.

[66] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

[67] Oliver Kosut, Liyan Jia, Robert J Thomas, and Lang Tong. Malicious data attacks on smart grid state estimation: Attack strategies and countermeasures. In *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 220–225. IEEE, 2010.

[68] Oliver Kosut, Liyan Jia, Robert J Thomas, and Lang Tong. Malicious data attacks on the smart grid. *IEEE Transactions on Smart Grid*, 2(4):645–658, 2011.

[69] Sudha Krishnamurthy, Soumik Sarkar, and Ashutosh Tewari. Scalable anomaly detection and isolation in cyber-physical systems using bayesian networks. In *ASME 2014 Dynamic Systems and Control Conference*, pages V002T26A006–V002T26A006. American Society of Mechanical Engineers, 2014.

[70] Marina Krotofil, Jason Larsen, and Dieter Gollmann. The process matters: Ensuring data veracity in cyber-physical systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 133–144. ACM, 2015.

[71] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[72] Pavel Laskov et al. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE Symposium on Security and Privacy (SP)*, pages 197–211. IEEE, 2014.

[73] Aron Laszka, Benjamin Johnson, and Jens Grossklags. 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)*, pages 319–332, December 2013.

[74] Aron Laszka, Waseem Abbas, S Shankar Sastry, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Optimal thresholds for intrusion detection systems. In *Proceedings of the 3rd Annual Symposium and Bootcamp on the Science of Security (HotSoS)*, pages 72–81, 2016.

[75] Aron Laszka, Jian Lou, and Yevgeniy Vorobeychik. Multi-defender strategic filtering against spear-phishing attacks. In *AAAI*, pages 537–543, 2016.

[76] Aron Laszka, Bradley Potteiger, Yevgeniy Vorobeychik, Saurabh Amin, and Xenofon Koutsoukos. Vulnerability of transportation networks to traffic-signal tampering. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.

[77] Bo Li and Yevgeniy Vorobeychik. Feature cross-substitution in adversarial classification. In *Advances in Neural Information Processing Systems*, pages 2087–2095, 2014.

[78] Bo Li, Yevgeniy Vorobeychik, and Xinyun Chen. A general retraining framework for scalable adversarial classification. *arXiv preprint arXiv:1604.02606*, 2016.

[79] Jingwen Liang, Oliver Kosut, and Lalitha Sankar. Cyber attacks on ac state estimation: Unobservability and physical consequences. In *2014 IEEE PES General Meeting— Conference & Exposition*, pages 1–5. IEEE, 2014.

[80] Thomas Liebig, Nico Piatkowski, Christian Bockermann, and Katharina Morik. Dynamic route planning with real-time traffic predictions. *Information Systems*, 64:258–265, 2017.

[81] Ondrej Linda, Todd Vollmer, and Milos Manic. Neural network based intrusion detection system for critical infrastructures. In *International Joint Conference on Neural Networks, 2009. IJCNN 2009.*, pages 1827–1834. IEEE, 2009.

[82] Viliam Lisỳ, Robert Kessl, and Tomáš Pevnỳ. Randomized operating point selection in adversarial classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 240–255. Springer, 2014.

[83] Chao Liu, Sambuddha Ghosal, Zhanhong Jiang, and Soumik Sarkar. An unsupervised spatiotemporal graphical modeling approach to anomaly detection in distributed cps. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.

[84] Wei Liu and Sanjay Chawla. Mining adversarial patterns via regularized loss minimization. *Machine learning*, 81(1):69–83, 2010.

[85] Yao Liu, Peng Ning, and Michael K Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14 (1):13, 2011.

[86] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge Discovery in Data Mining*, pages 641–647. ACM, 2005.

[87] Xiao-Yun Lu, Pravin Varaiya, Roberto Horowitz, and Joe Palen. Faulty loop data analysis/-correction and loop fault detection. In *15th World Congress on Intelligent Transport Systems*, 2008.

[88] Yunzhao Luo, Zhonghua Li, and Zhaojun Wang. Adaptive cusum control chart with variable sampling intervals. *Computational Statistics & Data Analysis*, 2009.

[89] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[90] Ralph Mac Nally and BT Hart. Use of cusum methods for water-quality monitoring in storages. *Environmental Science & Technology*, 31(7):2114–2119, 1997.

[91] Keith Marzullo. Tolerating failures of continuous-valued sensors. *ACM Transactions on Computer Systems (TOCS)*, 8(4):284–304, 1990.

[92] Peter W Mayer, William B DeOreo, Eva M Opitz, Jack C Kiefer, William Y Davis, Benedykt Dziegielewski, and John Olaf Nelson. Residential end uses of water, 1999.

[93] B McHugh. The opentripplanner project. *The OpenTripPlanner Project*, pages 12–16, 2011.

[94] Sean A McKenna, Mark Wilson, and Katherine A Klise. Detecting changes in water quality data. *Journal – American Water Works Association*, 100(1):74, 2008.

[95] Fei Miao, Miroslav Pajic, and George J Pappas. Stochastic game approach for replay attack detection. In *2013 IEEE 52nd Annual Conference on Decision and Control (CDC)*, pages 1854–1859. IEEE, 2013.

[96] Wanli Min and Laura Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606–616, 2011.

[97] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)*, 46(4):55, 2014.

[98] Y. Mo and B. Sinopoli. Secure control against replay attacks. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 911–918, Sept 2009.

[99] Yilin Mo, Rohan Chabukswar, and Bruno Sinopoli. Detecting integrity attacks on scada systems. *IEEE Transactions on Control Systems Technology*, 22(4):1396–1407, 2014.

[100] Yilin Mo, Sean Weerakkody, and Bruno Sinopoli. Physical authentication of control systems: designing watermarked control inputs to detect counterfeit sensor outputs. *IEEE Control Systems*, 35(1):93–109, 2015.

[101] Kate L Morrow, Erich Heine, Katherine M Rogers, Rakesh B Bobba, and Thomas J Overbye. Topology perturbation for detecting malicious data injection. In *2012 45th Hawaii International Conference on System Science (HICSS)*, pages 2104–2113. IEEE, 2012.

[102] Patric Nader, Paul Honeine, and Pierre Beauseroy. $\{l_p\}$-norms in one-class classification for intrusion detection in scada systems. *IEEE Transactions on Industrial Informatics*, 10(4): 2308–2317, 2014.

[103] Patric Nader, Paul Honeine, and Pierre Beauseroy. Mahalanobis-based one-class classification. In *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2014.

[104] Kien C Nguyen, Tansu Alpcan, and Tamer Basar. Stochastic games for security in networks with interdependent nodes. In *Game Theory for Networks, 2009. GameNets' 09. International Conference on*, pages 697–703. IEEE, 2009.

[105] ES Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[106] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[107] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design. In *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 2195–2201. IEEE, 2011.

[108] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, 58(11):2715–2729, 2013.

[109] Animesh Patcha and J-M Park. A game theoretic approach to modeling intrusion detection in mobile ad hoc networks. In *Proceedings of the 5th Annual IEEE SMC Information Assurance Workshop*, pages 280–284. IEEE, 2004.

[110] Jeffrey Pawlick, Sadegh Farhang, and Quanyan Zhu. 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)*, pages 289–308. Springer, 2015.

[111] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[112] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. *arXiv preprint arXiv:1705.06640*, 2017.

[113] Lina Perelman, Jonathan Arad, Mashor Housh, and Avi Ostfeld. Event detection in water distribution systems from multivariate water quality time series. *Environmental Science & Ttechnology*, 2012.

[114] Mohammad Ashiqur Rahman, Ehab Al-Shaer, and Md Ashfaqur Rahman. A formal model for verifying stealthy attacks on state estimation in power grids. In *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 414–419. IEEE, 2013.

[115] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[116] N Lawrence Ricker. Decentralized control of the tennessee eastman challenge process. *Journal of Process Control*, 6(4):205–221, 1996.

[117] Stephen Peter Robinson. *The development and application of an urban link travel time model using data derived from inductive loop detectors*. PhD thesis, University of London, 2006.

[118] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and JD Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, pages 1–14. ACM, 2009.

[119] Henrik Sandberg, André Teixeira, and Karl H Johansson. On security indices for state estimators in power networks. In *First Workshop on Secure Control Systems (SCS), Stockholm, 2010*, 2010.

[120] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

[121] Shigen Shen, Yuanjie Li, Hongyun Xu, and Qiying Cao. Signaling game based strategy of intrusion detection in wireless sensor networks. *Computers & Mathematics with Applications*, 62(6):2404–2416, 2011.

[122] Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. Pycra: Physical challenge-response authentication for active sensors under spoofing attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1004–1015. ACM, 2015.

[123] Roy S Smith. A decoupled feedback structure for covertly appropriating networked control systems. *IFAC Proceedings Volumes*, 44(1):90–95, 2011.

[124] Roy S Smith. Covert misappropriation of networked control systems: Presenting a feedback structure. *IEEE Control Systems*, 35(1):82–92, 2015.

[125] Siddharth Sridhar and Manimaran Govindarasu. Model-based attack detection and mitigation for automatic generation control. *IEEE Transactions on Smart Grid*, 5(2):580–591, 2014.

[126] Shiliang Sun, Rongqing Huang, and Ya Gao. Network-scale traffic modeling and forecasting with graphical lasso and neural networks. *Journal of Transportation Engineering*, 138(11): 1358–1367, 2012.

[127] André Teixeira, György Dán, Henrik Sandberg, and Karl H Johansson. A cyber security study of a scada energy management system: Stealthy deception attacks on the state estimator. *IFAC Proceedings Volumes*, 44(1):11271–11277, 2011.

[128] André Teixeira, Daniel Pérez, Henrik Sandberg, and Karl Henrik Johansson. Attack models and scenarios for networked control systems. In *Proceedings of the 1st International Conference on High Confidence Networked Systems*, HiCoNS '12, pages 55–64, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1263-9.

[129] André Teixeira, Henrik Sandberg, György Dán, and Karl H Johansson. Optimal power flow: Closing the loop over corrupted data. In *American Control Conference (ACC), 2012*, pages 3534–3540. IEEE, 2012.

[130] André Teixeira, Iman Shames, Henrik Sandberg, and Karl H Johansson. Revealing stealthy attacks in control systems. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1806–1813. IEEE, 2012.

[131] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, 2009.

[132] David I Urbina, Jairo A Giraldo, Alvaro A Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1092–1105. ACM, 2016.

[133] Junia Valente and Alvaro A Cárdenas. Using visual challenges to verify the integrity of security cameras. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 141–150. ACM, 2015.

[134] Marten Van Dijk, Ari Juels, Alina Oprea, and Ronald L Rivest. Flipit: The game of stealthy takeover. *Journal of Cryptology*, 26(4):655–713, 2013.

[135] Pravin Varaiya. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies*, 36:177–195, 2013.

[136] Ghislain Verdier, Nadine Hilgert, and Jean-Pierre Vila. Adaptive threshold computation for cusum-type procedures in change detection and isolation problems. *Computational Statistics & Data Analysis*, 52(9):4161–4174, 2008.

[137] Ognjen Vukovic, Kin Cheong Sou, Gyorgy Dan, and Henrik Sandberg. Network-aware mitigation of data integrity attacks on power system state estimation. *IEEE Journal on Selected Areas in Communications*, 30(6):1108–1118, 2012.

[138] Wenye Wang and Zhuo Lu. Cyber security in the smart grid: Survey and challenges. *Computer Networks*, 57(5):1344–1371, 2013.

[139] Peter Widhalm, Hannes Koller, and Wolfgang Ponweiser. Identifying faulty traffic detectors with floating car data. In *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*. IEEE, 2011.

[140] Le Xie, Yilin Mo, and Bruno Sinopoli. False data injection attacks in electricity markets. In *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 226–231. IEEE, 2010.

[141] Yuanchang Xie, Kaiguang Zhao, Ying Sun, and Dawei Chen. Gaussian processes for short-term traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, 2010.

[142] Wente Zeng and Mo-Yuen Chow. Optimal tradeoff between performance and security in networked control systems based on coevolutionary algorithms. *IEEE Transactions on Industrial Electronics*, 59(7):3016–3025, 2012.

[143] Yan Zhou and Murat Kantarcioglu. Adversarial learning with bayesian hierarchical mixtures of experts. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 929–937. SIAM, 2014.

[144] Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Bowei Xi. Adversarial support vector machine learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1059–1067. ACM, 2012.

[145] Quanyan Zhu and Tamer Başar. Indices of power in optimal ids default configuration: theory and examples. In *International Conference on Decision and Game Theory for Security*, pages 7–21. Springer, 2011.

[146] Nikolaos Zygouras, Nikolaos Panagiotou, Nikos Zacheilas, Ioannis Boutsis, Vana Kalogeraki, Ioannis Katakis, and Dimitrios Gunopulos. Towards detection of faulty traffic sensors in real-time. In *MUD@ ICML*, pages 53–62, 2015.